Bridging the conceptual gap between biological and artificial learning systems

Arna Ghosh

Doctor of Philosophy

School of Computer Science
McGill University
Montreal, Quebec, Canada

February 24, 2025

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Doctor of Philosophy

Abstract

The quest to understand intelligence and its origins has taken various forms, with one of the most recent frontiers emerging at the intersection of neuroscience and artificial intelligence (AI). This convergence has been fueled by the remarkable achievements of deep neural networks in AI and advancements in large-scale recording and analysis techniques in neuroscience. These developments have opened up exciting opportunities for synergistic interdisciplinary research, where artificial neural networks (ANNs) provide a testbed for exploring theories of learning in the brain, while neuroscience findings inspire the design of more sophisticated AI models. In this work, we present a framework aimed at unraveling the distinctive properties of individual learning systems. We focus on two key axes of comparison: the learning rule and the learned representations, in the context of studying learning in brains and ANNs.

First, we showcase how observations from computational neuroscience can serve as inspiration for developing metrics that assess the quality of learned representations in unsupervised representation learning, specifically within the realm of self-supervised learning (SSL) loss functions. Second, we develop a theoretical framework to understand the dynamics of SSL feature learning and investigate the computational advantages of specific biological mechanisms in aiding this process, thereby leading to practical recommendations that improve the sample and computational complexity of SSL. Lastly, we leverage insights by analyzing neural networks to gain a deeper understanding of how the brain may employ gradient approximations for learning and thereby, shed light on the role of certain design principles in the neuronal learning process. Taken together, our work contributes to the ongoing synergy between neuroscience and AI, bridging the conceptual gap between learning in the brain and in machines.

Abrégé

La quête pour comprendre l'intelligence et ses origines a pris de multiples formes, l'une des plus récentes se situant à l'intersection des neurosciences et de l'intelligence artificielle (IA). Cette convergence a été alimentée par les avancées remarquables des réseaux neuronaux profonds en IA et les progrès des techniques d'enregistrement et d'analyse à grande échelle en neurosciences. Ces développements ont ouvert de passionnantes perspectives pour une recherche interdisciplinaire synergique, où les réseaux neuronaux artificiels (RNA) servent de banc d'essai pour explorer les théories de l'apprentissage dans le cerveau, tandis que les découvertes en neurosciences inspirent la conception de modèles d'IA plus sophistiqués. Dans ce travail, nous présentons un cadre visant à élucider les propriétés distinctives des systèmes d'apprentissage individuels. Nous nous concentrons sur deux axes de comparaison: les mécanismes d'apprentissage et les représentations apprises, dans le contexte des neurosciences et des RNA.

Premièrement, nous montrons comment les neurosciences computationnelles peuvent inspirer le développement de métriques pour évaluer la qualité des représentations apprises dans l'apprentissage non supervisé, particulièrement dans le domaine des fonctions de perte en apprentissage auto-supervisé (AAS). Deuxièmement, nous développons un cadre théorique pour comprendre la dynamique de l'AAS et explorons les avantages computationnels de mécanism-es biologiques spécifiques pour faciliter ce processus, ce qui conduit à des recommandations pratiques pour améliorer la complexité en échantillons et en calcul de l'AAS. Enfin, nous exploitons des analyses des réseaux neuronaux artificiels pour approfondir notre compréhension de la manière dont le cerveau pourrait utiliser des approximations du gradient pour l'apprentissage, éclairant ainsi le rôle de certains principes dans le processus d'apprentissage neuronal. Notre travail contribue à la synergie croissante entre les neurosciences et IA, comblant le fossé conceptuel entre l'apprentissage dans le cerveau et dans les machines.

Contribution

Summary of Contributions

This thesis consists of original contributions towards demonstrating the synergies between computations underlying intelligent behavior in brains and Artificial Neural Networks (ANNs). First, I present a framework to compare and understand intelligent systems, and thereafter, I present evidence of shared computational motifs under specific pillars of this framework. Specifically, I make the following original contributions:

- **Representations in brains and ANNs:** A brain-inspired metric of representation geometry that is indicative of a shared structure of representations across brains and large-scale AI systems and is connected to downstream task performance, along with a pipeline to assess quality of representations learned by self-supervised trained ANNs (Part II).

- **Dynamics of feature learning:** Theoretical and empirical analysis of learning dynamics of high-dimensional representation when optimizing a self-supervised loss function, along with insights into computational advantages of neuroscience-inspired mechanisms (like orthogonality constraints and multi-view learning) and practical recommendations to improve the sample and compute-efficiency of self-supervised learning pipelines (Part III).

- **Learning with approximate gradients:** Characterization of the implications of gradient approximation, such as in brains, on learning and generalization properties using ANNs as a model, along with investigations into the role of architectural design choices, e.g. size of the network or non-linearity properties, on mitigating the effects of gradient approximation errors (Part IV).

Contribution of authors

- Chapters 1 and 2 provide introduction and background material for this thesis, which I wrote, drawing inspiration from various published sources and informal discussions throughout the course of my PhD.

- Part II is based on [Agrawal et al., 2022], which is a published paper at Neural Information Processing Systems (NeurIPS) 2022. In this project, I co-led the project ideation and design along with Kumar Krishna Agrawal, and co-led the experiments with Arnab Kumar Mondal. I was also responsible for the theoretical analysis presented in this paper, and co-developed the code[1] with Kumar Krishna Agrawal and Arnab Kumar Mondal. Arnab Kumar Mondal also led the efforts to scale up experiments on the compute cluster. Blake Richards was instrumental in shaping the narrative of the paper and provided overall supervision for the project. All authors contributed to writing the manuscript.

- Part III is based on [Ghosh et al., 2024], which is a published paper at Neural Information Processing Systems (NeurIPS) 2024. In this project, I co-led the project ideation with Kumar Krishna Agrawal, and led the experiment design of the project. Adam Oberman developed some of the key theoretical results presented in this paper, and provided overall supervision on research direction and manuscript writing. I collaborated with Adam on the theoretical results about learning dynamics. Shagun Sodhani helped run the large-scale experiments and validate the practical recommendations on large-scale datasets, specifically Imagenet. Blake Richards provided overall guidance and supervision on the project design, which shaped the narrative of the project. All authors contributed to writing the manuscript, with Kumar Krishna Agrawal, Blake Richards, and I co-leading the initial manuscript writing efforts.

- Part IV is based on [Ghosh et al., 2023], which is a published paper at International Conference on Learning Representations (ICLR) 2023. In this project, Blake Richards and Konrad Körding co-led the project ideation. I led the design of the analytical framework and experimental

[1]FastSSL library

setup to validate our theoretical results. Yuhan Helena Liu and Guillaume Lajoie helped analyze results, and contributed to discussions that led to extending the theoretical results to incorporate the results on generalization. Blake Richards provided the overall guidance and supervision for this work, and also played a pivotal role in the manuscript writing process. Konrad Körding contributed to the manuscript writing as well as the reviewer rebuttal process. All authors contributed to writing the manuscript.

- Chapter 9 consists of discussion and broader impact of the results presented in this thesis which is written by me, with inspiration from discussions I have had with Blake Richards, Shahab Bakhtiari, and Zahraa Chorghay during the final year of my PhD.

Acknowledgements

Completing this thesis has been an exciting journey and an extremely enriching experience, and I am incredibly grateful to several people who supported me, both intellectually and emotionally, along the way. Science is done by people, and people are embedded in communities. Unsurprisingly, this work has been possible because of the social support of my family and friends, both in Montréal and around the world.

First and foremost, I would like to extend my heartfelt gratitude to my PhD supervisor, Blake Richards, for his unwavering support, encouragement, and mentorship throughout my PhD. Discussions with Blake shaped my thinking about the fields of neuroscience and AI, as well as my approach to science in general. Moreover, I am very grateful to Blake for giving me numerous opportunities to present my research at conferences and scientific meets, which have helped improve my science communication skills and engage with the broader scientific community. I would also like to thank the generous support of my funding sources, particularly the Healthy Brains for Healthy Lives Graduate Fellowship and the Vanier Canada Graduate Scholarship, for their financial support throughout my PhD.

I would like to take this opportunity to thank my incredible mentors who were instrumental in shaping my research direction and projects (in alphabetical order): Adam Oberman, Anthony (Tony) Zador, Edward Ruthazer, Eric Shea-Brown, Guillaume Lajoie, Konrad Körding, Shagun Sodhani, and Stuart Trenholm. I am also incredibly grateful to my MSc supervisors, Marie-Hélène Boudrias and Georgios Mitsis, who believed in me, including at times when I myself was short on self-belief, and consistently encouraged me to advance my scientific career, including pursuing my PhD. My scientific journey would have definitely been cut short if not for the support of all my mentors. I would also like to thank my PhD advisory committee comprising of Aaron Courville and Stuart Trenholm for their guidance and helpful discussions that have shaped the overall structure of this thesis. I am also grateful to my

PhD thesis examiners, Adriana Romero-Soriano and Rui Ponte Costa, as well as my PhD defence committee pro-dean, Adrien Peyrache, and chair, David Ifeoluwa Adelani, for their invaluable feedback and thoughtful questions that have significantly improved the quality and overall presentation of the thesis. Furthermore, I would like to thank my professors at McGill and Mila who have helped me foster my research interests and have always been open for a friendly chat about science, research, and beyond: Adrien Peyrache, David Rolnick, Doina Precup, Gauthier Gidel, Ioannis Mitliagkas, Joëlle Pineau, Kaleem Siddiqi, Karim Jerbi, Michael Langer, and Siamak Ravanbakhsh.

Research is a collaborative effort, and I have been extremely fortunate to have worked with some great researchers and thinkers as my collaborators on projects: Ann Huang, Arnab Kumar Mondal, Augustine Mavor-Parker, Kumar Krishna Agrawal, Divyansha Lachi, Dongyan Lin, Jonathan Cornford, Mandana Samiei, Matteo Gamba, Melody (Zixuan) Li, Olivier Codol, Pingsheng Li, Raymond Chua, Roman Pogodin, Ronan da Silva, Rudi Tong, Sonia Joseph, Yuhan Helena Liu, and Zahraa Chorghay. I would like to take a moment to especially extend my gratitude to Kumar Krishna Agrawal, with whom I have had the pleasure of discussing scientific projects, politics, and more. I would also like to extend my heartfelt gratitude to Olivier Codol for translating the abstract of this thesis to French.

I consider myself to be extremely fortunate to have been a member of Blake's lab, and learn from Blake and past and current members of the LiNC lab: Aidan Sirbu, Aleksei Efremov, Annik Yalnizyan-Carson, Anthony Chen, Charlotte Volk, Chen Sun, Colleen Gillon, Colin Bredenberg, Damjan Kalajdzievski, Dane Carnegie Malefant, Daniel Levenstein, Dongyan Lin, Jenny Zheng, Jordan Guerguiev, Jonathan Cornford, Joshua Tindall, Katharina Wilmes, Kirthana Sathiyakumar, Krystal Xuejing Pan, Luke Prince, Mandana Samiei, Mashbayar Tugsbayar, Melody Zixuan Li, Pingsheng Li, Raymond Chua, Roman Pogodin, Roy Henha Eyono, Shahab Bakhtiari, Soheila Samiee, Sonia Joseph, and Surya Penmetsa. I will deeply miss our discussions during lab meetings, journal clubs, coffee sessions, and casual hangouts, which have largely shaped my philosophy of science. Moreover, being a member of the Mila Quebec AI Institute has been an integral part of my PhD experience. I would like to extend my gratitude to my peers at Mila for creating a collaborative and friendly atmosphere, especially the following friends: Arnav Kumar Jain, Arushi Jain, Avery Ryoo, Ezekiel Williams, Gabrielle Hurtubise-Radet, Geneviève Couture, Jose Gallego-Posada, Julien Besset, Ludovic Soucisse, Manuela Girotti, Martin Weiss, Niki Howe, Olivier Codol, Rim Assouel, Sangnie Bhardwaj, Sarthak Mittal, Sékou-Oumar Kaba, Simon Galarneau, Sumana Basu, Tegan Maharaj, and Victor Schmidt. I would also like to thank Ann Jack and Annie Le Bire from McGill University and Julien Wiredu from Mila, for helping me with administrative processing tasks, and Olexa Bianuik from the Mila IDT team for helping solve numerous server-level issues on the Mila cluster.

During my PhD, I have been fortunate to work with several amazing researchers at Meta and Google, who helped me explore the overlap between my research interests and research projects in industry: Alexander Miller, Blaine Lewis, Blaise Aguera y Arcas, Carlos Xavier Hernández, Dan Wetmore, David Schwab, Esteban Real, Florian Bordes, Joao Sacramento, Katarzyna Szymaniak, Koustuv Sinha, Manuel del Verme, Mohammad Pezeshki, Reyhane Askari Hemmat, Ricardo Monti, Shagun Sodhani, Suman Mulumudi, Yiğit Demirağ, and Zöe Steine-Hanson. My internship experiences have been pivotal in broadening my horizons and position my research interests to better align with consumer product requirements to better serve the community.

My life in Canada would be incomplete without the social support of my friends here. I am extremely grateful to all of them, not only for the fun moments, but for lending an ear whenever I wanted to complain about work or life, making my life that much easier: Ajin Jiji Tom, Alekhya Munagela, Aman Joshi, Amar Kumar, Aniket Raj, Anirban Kundu, Arnab Mondal, Aryann Gupta, Danielle Benesch, Dishank Bansal, Irum Chorghay, Ishita Hiremath, Jagannadh Sripada, Komal K Teru, Koustuv Sinha, Lalit Singh, Lokha Ranjani Alagar Boopathy, Mehak Pannu, Naman Sharma, Neil Banik, Nitish Chandrashekharaiah, Pampa Banik, Pratap Singh Markam, Renaud Legoux, Senthilkumar Kailasam, Sethu Kovendhan Boopathy Jegathambal, Sivakumar Chidambaram, Tapabrata Mukherjee, Tejas Vaidhya, Tuhin (Bob) Banik , Upasana Dasgupta, and Vinaya Hari. I would also like to thank friends all over the world who have consistently been there both virtually (thanks to voice and video calls) and in-person (the occasional yearly get-together): Abhishek Sultania, Alba Xifra-Porxas, Atal Agarwal, Amrit Ashu, Anwesha Sur Dey, Aratrika Sengupta, Ayantika Sengupta, Ayushi Chakraborty, Bodhisattya Bhattacharya, Chandreyee Bhaumik, Korok Bhattacharjee, Michalis Kassinopoulos, Mitesh Gulati, Rohan Ghoshal, Sayantika Ray, Shubhayan Ghosh, Siddharth, Soham Saha, Sourav Das, Sravya Kovvali, Srinjoy Ghosh, Suryapratim Chakrabarti, Vikram Mohanty, Vishal Verma, and Yashdeep Singh.

Finally, I would like to thank my parents for being patient with my research pursuits and supporting me through this journey. And lastly, none of this would have been possible without the support of Zahraa Chorghay, who kept me sane through the process, and often had to be on the receiving end of my half-baked ideas and shower thoughts. I am sincerely grateful for their support and belief in me, which helped me push through and overcome obstacles, both scientific and personal.

Contents

Part I

Introduction

1 Introduction

Throughout its history, humanity has embarked on the quest to understand intelligence and its origins. This quest has taken many forms, from philosophy to psychology, and from biology to computer science. In recent decades, two fields have emerged as particularly prominent in this endeavor: artificial intelligence (AI) and neuroscience. Both AI and neuroscience aim to shed light on the nature of intelligent systems, one from an in silico perspective and the other from a biological perspective. While AI researchers direct most of their efforts to create systems that can solve computationally challenging tasks, such as image recognition, natural language processing, or decision-making, neuroscientists aim to deconstruct the biological basis of intelligence and behavior, seeking to uncover how the brain processes information, learns, and adapts to changes in the agent's environment. Despite the difference in their approaches, AI and neuroscience share a fundamental interest in understanding intelligence, and their findings can and have informed and inspired each other over the last few decades. In this thesis, I will explore the intersection of AI and neuroscience by highlighting key axes along which progress in one field can be leveraged to advance the other.

A key ingredient of intelligent systems, identified by both AI and neuroscience, is *learning*. Learning can be defined as changes in the system's internal parameters to better adapt to its environment. AI, powered by deep learning (DL), has made large strides towards building human-like intelligent systems, by leveraging neural network models (ANNs) and the backpropagation algorithm. The backpropagation algorithm provides a mechanism to compute the gradient of some scalar objective function with respect to each parameter in the ANN. The gradient vector indicates the direction of steepest ascent in the parameter space; following this will result in the largest possible change in the objective function for the smallest possible change in parameters. In practice, learning in ANNs uses a loss function (instead of objective function) and takes a step in the direction of steepest descent (instead of ascent), therefore being termed as gradient descent. The backpropagation algorithm, followed by a step of gradient descent underlies, in large part, the success of ANNs.

Given this recipe for success in ANNs for learning complex tasks, it is reasonable to ask whether the brain follows a similar approach. Specifically, does the brain use gradient-based methods for synaptic weight changes? That is, even if the brain doesn't use backpropagation, might it have an alternative that achieves something similar? Furthermore, are the emergent system properties, i.e. the representations, similar in the brain and ANNs? These questions raise an even broader question: can our understanding of learning in biological brains inspire the development of next-generation AI systems? While there are undoubtedly differences between biological and artificial intelligent systems, exploring their potential convergence can reveal the underlying principles governing intelligence more broadly. The search for these common principles forms the foundation of the field of neuroscience-inspired AI, an area of AI research that aims to uncover the general laws underlying learning in intelligent systems — applicable to both biological and artificial brains.

1.1 Marr's levels of analysis

To answer these questions, we need a framework for comparing two learning systems that are, at face value, very different. For this, we turn to David Marr's levels of analysis for understanding information-processing systems [Marr, 1982]. Briefly, Marr proposed that a system can be understood at three levels – computational, algorithmic and implementation.

- *Computational level*: This level describes the problem to be solved, the available inputs to the system and the desired solution, i.e. the desired output for specific inputs.

- *Algorithmic level:* This level specifies the steps or procedures involved in solving the problem, including the algorithms and data structures required to implement the desired solution.

- *Implementation level:* This level details the physical realization of the algorithm. It describes the hardware or neural circuitry that carries out the computations.

Marr's levels of analysis are useful to distill the inter-level relationships of a complex system, i.e. this framework allows us to gain a mechanistic understanding of how a system uses its constituent components to perform a particular set of computations in order to solve a task. Take, for example, the human behavior of catching a ball. The system, in this case a human, needs to solve the *computational level* of the problem by computing or estimating the ball's trajectory and positioning their body at the drop point of the ball's arc. The corresponding *algorithmic level* of the problem could be to process the image, identify the ball, track it, then calculate the direction required to run in order to keep the ball's position relative to the body constant or moving at a constant vertical velocity. Finally, the *implementation level* could be far more complicated, involving the neural substrates that are involved in the visual processing of the image of the ball, as well as those involved in calculating the spatial trajectory and controlling the muscle movement of the body to ensure it matches the motion of the ball. These three levels, taken together, provide a mechanistic explanation of a human's behavior of catching a ball.

1.2 Beyond Marr's levels: Towards a Normative Framework

Understanding learning systems, however, requires one to consider how the information processing behavior changes with experience. This requires a more fine-grained consideration of the factors that impact the functions performed by the system, at each level of Marr's framework. To this end, what we need is a *normative framework that encapsulates the various facets of learning and can be used to compare intelligent systems.*

With this goal in mind, in this thesis, I present a more granular framework (see Figure 1.1) that is inspired by Richards *et al.* [Richards et al., 2019].

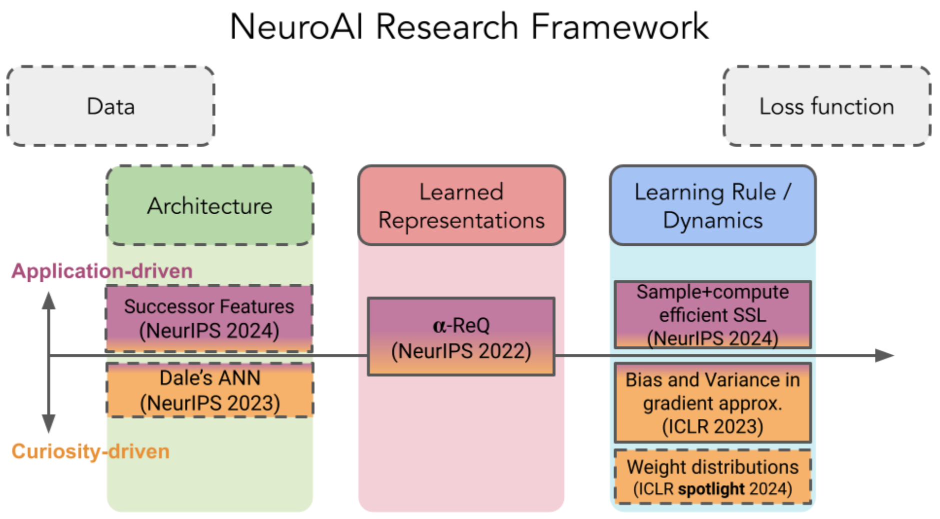

NeuroAI Research Framework
Data
Loss function
Architecture
Learned Representations
Learning Rule / Dynamics
Application-driven
Curiosity-driven
Successor Features (NeurIPS 2024)
Dale's ANN (NeurIPS 2023)
α-ReQ (NeurIPS 2022)
Sample+compute efficient SSL (NeurIPS 2024)
Bias and Variance in gradient approx. (ICLR 2023)
Weight distributions (ICLR **spotlight** 2024)

Figure 1.1: General framework for comparing learning systems and corresponding projects binned into respective pillars. Subsequent chapters will focus on projects under the Representations and Learning Rule pillars, specifically those highlighted with solid-border boxes. In contrast, dashed-border boxes indicate collaborative works that are not covered in this thesis. While the pillars of data and loss Function are fundamental to learning systems, they were not the focus of this doctoral research.

Note that, while Richards *et al.* elucidate the core pillars of *designing* artificial learning systems that are in turn useful for studying biological learning systems, the framework presented here describes the key factors that can be used to study, understand and compare the behavior of learning systems – both biological and artificial.

The framework consists of five key pillars that govern the behavior of learning systems, as follows:

- *Data obtained from environment:* The data available plays a crucial role, following Marr's computational level of analysis, as it determines the input provided to the system for solving a task. Furthermore, the quality and characteristics of the available data are essential for the system to learn a certain behavior.

- *Architecture:* The system's architecture for information processing imposes specific inductive biases, narrowing down the space of possible functions it can compute.

- *Learned representations:* The learned representations provide insights into the intermediate computations utilized by the system's algorithms to solve a (compositional) task. Studying the learned representations is essential while comparing different learning systems that exhibit similar behavior at the computational level, but might differ at the algorithmic level.

- *Learning rule:* The learning strategy, governed by a learning rule, introduces dynamics that shape the trajectory of learned functions and ultimately determine the computations involved in supporting the system's learned behavior.

- *Loss/Objective Function:* The loss function represents the objective that the system aims to optimize, typically measuring how closely the system's current behavior matches the desired behavior while satisfying specific operational constraints. Different formulations of the objective function can reflect differences in the operational constraints that the system must satisfy, eventually leading to distinct solutions and computational motifs that the system employs to achieve the desired outcome.

Let us again consider the example of catching a ball. For a system to learn how to catch a ball from experience, it needs to solve the task of tracking the ball's position and positioning the body accordingly. To understand how the system accomplishes this learning, we need to inspect the experiential data it learned the behavior from. For instance, what was the color of the ball? What were its other physical attributes, e.g. weight or material, of the ball? It is possible that if the color of the ball changes, the system can no longer perceive or detect the ball. If the weight or material of the ball changes, will its aerodynamics be altered to the extent that the system can no longer estimate its trajectory? Along similar lines, the architecture of the system imposes certain inductive biases over what computations the system can perform. For instance, if the system has an architectural bias for detecting round-shaped objects, tracking the ball's position might be a simpler problem than without this inductive bias. Similarly, having an inductive bias over translation-invariant computations might help the system better track the ball's position.

While the data and architecture pillars provide insights into the constraints that the system operates under, studying the learned latent representations provides insights into the intermediate algorithmic-level processing stages used by the system while exhibiting the behavior in question. For instance, the latent representations could indicate whether the underlying algorithm decomposes the task into a combination of perception (ball tracking) and motor (body positioning) problems. Investigating how these latent representations emerge over the course of learning provides insight into the system's internal mechanisms that enable learning. For instance, the system's internal representations might not be initially specialized for perception or motor skills related to ball catching, but over the course of learning, such a specialization might emerge. Understanding the learning rule used in shaping these representations (and the emergent behavior) could provide insight into how the system uses the available input and output signals to change the computations it performs over the course of learning. For instance, the system could rely on external global feedback signals (how well was it able to catch the ball) to alter its computations or the system could rely on intermediate self-generated feedback signals (how well it could track the ball, or how close it was able to get to the ball) to alter computations that eventually lead to an improved behavior. Each strategy would lead likely to a different set of computations that the system could perform, thereby leading to differences in the algorithmic and implementation levels.

Finally, a better understanding of the loss functions that the system attempts to optimize in order to solve the task provides an insight into the constraints that the system operates under as well as the potential solutions that the system can converge to. For instance, if the system wants to reduce any sudden movements, it would attempt to reduce any last minute motor movements that are required to adjust to the ball's trajectory. Consequently, it would devote more resources towards better tracking the ball's trajectory such that it is able to plan the sequence of motor actions *a priori*, with minimal last minute adjustments. In contrast, a system with stronger constraints over perception might learn fine-grained motor skills that are more useful for last-minute adjustments. Understanding the loss function(s) that the system optimizes might also provide insights into the development of specialized subsystems for perception and motor skills. For instance, a system might use a loss function for object detection and tracking that considers only how to learn the computations underlying perception. The result of these computations could then provide input to a distinct motor subsystem to output the motor commands required for self-positioning and catching the ball.

Although the factors are discussed in isolation, in reality, these factors often interact extensively. For instance, a learning system could use learning rules that are better suited to its architecture over another. Furthermore, the behavior of an active learning system would impact the data experienced by it. For instance, if the system is able to track the ball from far away and is able to position itself roughly along the trajectory of the ball, it will experience more data streams where the system's behavior needs to be adjusted when the ball is close. Indeed, such biases in data streams can be exploited when designing learning curricula for artificial learning systems, wherein the system is presented with data that is easier to learn from early in the learning process.

By analyzing and comparing information processing systems using the five pillars of the framework I have presented, we can attain a comprehensive understanding of learning systems. It is worth emphasizing that this framework offers a normative view of the computations performed by the system, i.e. it provides insight into how the five factors impact the computations that are learned to successfully exhibit a behavior according to some norm.

1.3 Why Artificial Neural Networks?

To search for shared principles of intelligence between artificial and biological systems, we can turn to ANNs. Not only can ANNS solve computationally challenging tasks like the brain [LeCun et al., 2015], but ANNs provide a cheaper, more observable test-bed to study the impact of different design factors on the computations underlying a particular behavior. Therefore, they allow us, scientists concerned with understanding intelligence, to stick to Richard Feynman's ideology "*What I can not create, I can not understand*"[1]. Moreover, despite their differences at the implementation level (i.e. silicon-based chips compared to carbon-based networks of cells), ANNs allow us to study computations that roughly match the brain's at an algorithmic level, enabling a designer to increase the level of biological relevance of their model and study the impact of its algorithmic properties on the emergent computations.

A second, perhaps more opportunistic reason, is that connectionist models enabling intelligent behavior are increasingly lauded by the scientific com-

[1]https://www.goodreads.com/quotes/7306651-what-i-cannot-build-i-do-not-understand

munity, most recently being validated in the form of a Nobel prize in Physics[2]. The immense success of connectionist models in learning intelligent behavior by changing the internal connection weights signifies the position of these models as our best existing bet to build intelligent systems that can both solve human-level tasks and perform similar computations, thereby allowing us to discover the core principles that underlie intelligent behavior more generally [Doerig et al., 2023].

In this thesis, I will present evidence of algorithmic level similarity between brains and ANNs and demonstrate three key examples of how such similarities can be leveraged to improve several design choices of AI systems. The three examples can be grouped under the pillars of learned representations and learning rules. My principal hope is that I may convince the reader of the utility of using ANNs to understand and leverage shared motifs of intelligence between biological and artificial systems. Over the rest of this chapter, I will present the specific research questions that I sought to answer. I will then proceed to describe a brief overview of the biological motivation and corresponding results from ANN experiments.

1.4 Key Research Questions

This thesis will investigate the following research questions in each of the subsequent chapters:

- Do brains and deep neural networks learn similar representations, in terms of the geometry of the representations? Can we use the geometry of the learned representations to identify ANN models with good performance?

- Does representation geometry change similarly in brains and deep neural networks during learning? Can we leverage these similarities to design sample and compute-efficient learning strategies for ANNs?

- Given that the brain cannot compute the exact gradient of a loss function with respect to each of its parameters, can we understand the impact of using some approximation to the gradient signal on the learning and generalization properties using ANNs?

[2]https://www.nobelprize.org/prizes/physics/2024/press-release/

1.5 Preview of Results

The remaining chapters of the thesis aim to systematically answer these questions. In each chapter, we will draw inspiration from prior studies uncovering certain properties of learning or neural activity in the brain, and use ANNs to better understand the role of such properties in enabling the algorithms and computations underlying intelligent behavior. As elaborated above, ANNs provide a more observable framework than biological systems, thereby making it easier to study the contribution of individual properties in isolation. One (*Ipcha Mistabra*) might argue that such an endeavor may be unreasonable, especially given that the five pillars often interact with each other. But as we often adopt a reductionist view in science, our hope is to simplify the system into its individual constituents, study and understand them better, and then put them back together to try and reconstruct the system. I hope to convince the reader that the results presented in this thesis can indeed provide a better combined understanding of both biological and intelligent systems.

This thesis first presents evidence for shared computational motifs in brains and ANNs by investigating their learned representations. In Part II, we draw inspiration from recent findings in the mammalian visual cortex (V1) that demonstrate scale-free high-dimensional representations for naturalistic images [Stringer et al., 2019, Kong et al., 2022]. These results are indicative of the representation geometry in the mammalian brain, specifically, the manifold of natural image representations in the brain's internal high-dimensional space. Motivated by these findings, we study the geometry of learned representations in ANNs, specifically utilizing a measure developed from systems neuroscience. We formally connect this metric of representation geometry to downstream task performance under certain simplistic assumptions. Moreover, we demonstrate that not only do representations in deep neural networks of vision exhibit a scale-free nature like brains, but also, that this measure of representation geometry can be used to assess the quality of ANN models, especially those trained using self-supervised learning (SSL) frameworks. Taken together, this chapter demonstrates that advances in systems neuroscience could lead to efficient design principles for ANNs, and in turn, ANNs could provide a testbed for normative explanations of observations in the brain.

In the next chapter, we build upon these results to study the learning dynamics, i.e. how these high-dimensional representations are shaped over the course of learning. This chapter draws motivation from two key results in neuroscience. First, behavioral neuroscience studies have demonstrated that animals tend to spend more time investigating novel rather than familiar objects: i.e. they spend more time observing the novel object from several views or interacting with it [Antunes and Biala, 2012]. Second, systems neuroscience studies have found that representations in mouse V1 become more orthogonal over the course of learning [Failor et al., 2021]. In other words, neural representations tend to increase in dimensionality as the animal learns a behavior. Inspired by these two findings, we aimed to better understand the learning dynamics of ANNs trained using a SSL loss function, i.e. training without explicit human supervision [LeCun, 2022]. To do so, in Part III, we provide a theoretical characterization of the explicit biases imposed by common SSL losses and the implicit biases imposed by gradient-based optimization on the learning dynamics; specifically, we study how these biases shape the learned feature space. We leverage this theoretical understanding to propose a normative role for using orthogonalization constraints on the feature space and for using multiple views of an object during learning. Furthermore, we demonstrate that incorporating these strategies in non-contrastive SSL methods lead to compute and sample-efficient learning pipelines, yielding better representations earlier in training. Taken together, this chapter demonstrates that advances in behavioral and systems neuroscience can act as guiding principles for designing efficient learning strategies for ANNs, and in turn, advances in theoretical deep learning could provide normative explanations for experimental observations in biological systems that might seem phenomenological at first.

For the next chapter, the reader's attention is directed towards learning mechanisms in the brain. Notably, learning in ANNs is achieved using gradient descent, i.e. updating parameters to best follow the gradient of the loss function, thereby ensuring each small update step in the parameter space leads to the greatest reduction possible in the loss. Computing the gradients for an ANN with hierarchical structure leverages the backpropagation algorithm [Rumelhart et al., 1986]. Brains, however, cannot compute the gradient of the loss using the backpropagation algorithm, owing to their physical constraints [Lillicrap et al., 2020]. Therefore, they must have mechanisms to approximate the gradient if they are to rely on small updates to their synaptic weights to learn a task [Richards and Kording, 2023]. In Part IV, we investigate the impact of approximating the gradient and its implications for learning and generalization. Once again, we leverage ANNs, owing to their observability and controllability in experimental settings, using tools from deep learning theory to characterize the impact of gradient approximation errors on learning and generalization. Furthermore, we study the role of different architecture design choices in mitigating the potential negative impacts of such approximations. Specifically, we analytically and empirically demonstrate that increasing the size of the network and adding a sparsity promoting non-linearity on the activations helps to mitigate the negative impacts of approximation errors during training while also benefiting from improved generalization owing to noise in the optimization process.

Finally, in Chapter 9, I will present a broader perspective of how the fields of neuroscience and AI can leverage the existing synergies to accelerate the advances in their respective domains. While operating at seemingly different levels of systems understanding, I believe that each field has a wealth of knowledge to offer the other, as evidenced by the results presented here. Furthermore, I believe the emerging field of NeuroAI will help shape these directions of convergence, eventually leading to a better understanding of the shared physical principles that underlie both biological and synthetic intelligence.

2

Background

This chapter will introduce fundamental concepts that broadly shape the remainder of this thesis. In addition to this, at the beginning of each chapter, I will also discuss relevant literature that is specific to the contributions in the chapter.)

We will explore three key topics in this chapter: artificial neural network architectures, backpropagation, and self-supervised learning (SSL) loss functions. Notably, these sections correspond to three (of five) pillars of the NeuroAI framework: architecture, learning rule, and loss function, respectively. Moreover, this chapter also presents a short section on eigenvalue decomposition, a linear algebra technique that is used in the methodology of all subsequent chapters. Note that this chapter is not intended to be comprehensive reading material for either of these topics, but rather a concise introduction that helps the reader grasp the core contributions of the following chapters. Readers seeking a more in-depth understanding of any of these topics are encouraged to consult relevant literature that specifically deal with these topics.

2.1 Artificial Neural Networks (ANNs)

Artificial neural networks (ANNs) have revolutionized the field of artificial intelligence, enabling significant advancements in solving computationally challenging problems across scientific disciplines, such as object recognition [Krizhevsky et al., 2012], language modeling [Vaswani et al., 2017], and protein folding [Jumper et al., 2021]. It is worth mentioning that the structure of ANNs is inspired by our understanding of the human brain, specifically following the ideology of connectionist models [Doerig et al., 2023]. Traditionally, to understand learning and different cognitive phenomena, connectionist modeling leveraged a network of interconnected nodes, or units, that processed information in parallel. Similarly, ANNs are composed of multiple layers of interconnected nodes, or artificial neurons, that process and transform input data into meaningful output. The connections between nodes represent synapses, and the strength of each synapse, called its weight, determines the influence that one (upstream) neuron has on the activity of another (downstream) neuron. These weights act as adjustable parameters that can be changed through a learning process in order to solve a task or exhibit desired behavior. On the other hand, the connectivity pattern among individual units imposes implicit biases over the functions that can be represented by the ANN. As a result, different architectures that are explored in this thesis have biases toward learning specific types of functions. In this section, we will delve into some of the most popular ANN architectures and their key components.

2.1.1 Fully-Connected Networks (FCNs)

Fully-connected networks (FCNs), also known as multi-layer perceptrons (MLPs), are a fundamental network architecture that apply a sequence of linear-nonlinear transformations over its inputs [Rosenblatt, 1958]. Each layer of an FCN can be thought to map an input vector in $\mathbb{R}^{d_{in}}$ to an output vector in $\mathbb{R}^{d_{out}}$. Each neuron in a layer is connected to every neuron in the subsequent layer, thereby allowing every input feature to influence every dimension in the output vector. While this connectivity profile allows the network to represent highly expressive functions in high-dimensions, it also leads to a large number of parameters in each layer, specifically $\mathcal{O}(d_{in} \times d_{out})$. Therefore, the number of parameters scales linearly with both the dimensionality of inputs and outputs.

As noted before, each layer in an FCN applies a linear transformation followed by a non-linear activation function. Mathematically, the transformation at some layer l can be written as:

$$\mathbf{h}^{(l)} = \mathcal{F}\left(\mathbf{h}^{(l-1)}\mathbf{W}^{(l)} + \mathbf{b}^{(l)}\right) \tag{2.1}$$

where:

- $\mathbf{h}^{(l)} \in \mathbb{R}^{B\times d_l}$, represents the activations in layer l and B is the number of input vectors,
- $\mathbf{W}^{(l)} \in \mathbb{R}^{d_{l-1}\times d_l}$, is the weight matrix connecting layer $l-1$ to layer l,
- $\mathbf{b}^{(l)} \in \mathbb{R}^{d_l}$ is the bias vector for layer l, and
- $\mathcal{F}(\cdot)$ is an activation function, such as ReLU, tanh, or sigmoid.

The output of layer (l), $\mathbf{h}^{(l)}$, serves as the input to the next layer $(l+1)$, propagating information through the network until the final layer, which produces the output.

The subsequent chapters in this thesis will present several settings where the objective is to classify inputs into certain discrete categories. For this specific purpose, the desired output of the ANN is a probability distribution that indicates the likelihood of the input to be in each category. For this purpose, the final layer's output is often passed through a softmax function:

$$\text{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{2.2}$$

where $\mathbf{z}$ represents the input to the softmax layer, and the resulting vector represents the probabilities for each class, with each element lying in the range $[0, 1]$ and the total summing to 1.

Despite their simple formulation and ease of implementation, FCNs are less popular for processing high-dimensional naturalistic inputs such as images or text sequences, primarily because they do not exploit local structures. Instead, FCNs have been widely used for demonstrating the utility of ANNs in performing complex computations, and continue to serve as platforms for theoretical analysis. For instance, FCNs have been used to demonstrate that ANNs are universal function approximators [Hornik et al., 1989], a result that has served as motivation for decades of research eventually leading to advanced ANNs architectures that have driven the modern AI revolution.

2.1.2 Convolutional Neural Networks (CNNs)

Convolutional neural networks (CNNs) are designed to efficiently process spatial data, such as images. Images are known to possess locally consistent structures, i.e. nearby pixels have similar correlated information [LeCun and Bengio, 1995]. Furthermore, there also exists a notion of translation equivariance in low-level image structures. For instance, the left and right edges of a black box on a white background are represented similarly in the pixel space of the image, and therefore an edge detector should respond similarly at the two different locations. CNNs are designed to detect such translation-equivariant structures, by applying filters (used for convolution operation) that detect local patterns. Notably, the convolution layer uses a local connectivity pattern whereby each unit in the output is influenced only by a set of spatially-connected input units, i.e. input units in a local region on the input. Coupled with a weight sharing strategy, where different units in the output layer apply the same filter at different spatial locations in the input, CNNs can detect translation-equivariant local structures in an input. Moreover, CNNs exploit the spatial hierarchy in data through a series of convolutional and pooling layers, allowing them to identify features like edges, textures, and shapes. This local-connectivity approach reduces the number of parameters compared to fully-connected networks, making CNNs more efficient for processing images.

Each convolutional layer applies a set of learnable filters over local regions of the input. The convolution operation can be represented as:

$$\mathbf{y}_{i,j} = \sum_{c=1}^{c_{in}} \sum_{m=1}^{M} \sum_{n=1}^{N} \mathbf{X}_{c,i+m,j+n} \cdot \mathbf{K}_{c,m,n} \tag{2.3}$$

where:

- $\mathbf{X} \in \mathbb{R}^{c_{in} \times H \times W}$ is the input feature map of c_{in} channels, and size $H \times W$,
- $\mathbf{K}$ is the convolutional kernel (or filter) of size $c_{in} \times M \times N$, and
- $\mathbf{y}_{i,j}$ is the output of the convolution at position (i, j).

This convolution operation can be applied multiple times, resulting in an output feature map of c_{out} channels. Therefore, a convolution layer performs a linear transformation on an input tensor of shape $c_{in} \times H \times W$ to yield an output tensor of shape $c_{out} \times H_2 \times W_2$, where H_2, W_2 depends on the positions (i, j) the convolution operator was applied. The stride and padding parameters determine the values of these positions. For more details on these parameters, the reader is deferred to signal processing textbooks which deal with convolution operations. Note that the number of learnable parameters in a convolution layer is $\mathcal{O}(c_{out} \times c_{in} \times M \times N)$, i.e. it scales linearly with number of channels in the input and output tensors but not their spatial dimensions, contrary to FCNs.

The convolution operation is typically followed by an activation function f, such as ReLU, which introduces non-linearity:

$$\mathbf{y}_{i,j} = \mathcal{F}\left(\sum_{c=1}^{c_{in}}\sum_{m=1}^{M}\sum_{n=1}^{N}\mathbf{X}_{c,i+m,j+n} \cdot \mathbf{K}_{c,m,n}\right) \tag{2.4}$$

Pooling Layers

Pooling layers reduce the spatial dimensions of feature maps, thereby decreasing the dimensionality of computations and providing a form of translation invariance. Since pooling layers are often applied after the convolution layers, they perform spatial aggregation over the output of convolution layers to learn translation-invariant features. For instance, while convolution operation with an edge detector filter leads to similar outputs at different locations containing edges, the pooling operation aggregates over a spatially local region and indicates if there is an edge present in that region.

For a pooling operation over a region of size $M \times N$, the output at position (i, j) is given by:

$$\bar{\mathbf{y}}_{i,j} = \mathcal{G}(\mathbf{y}_{i:i+m,j:j+n}) \tag{2.5}$$

One common pooling operation is average pooling, which downsamples the input by taking the mean value of activations over a spatially local region, i.e. $\mathcal{G} = mean(.)$. Other pooling methods, such as max pooling, select the maximum activation value within each region rather than averaging, i.e. $\mathcal{G} = max(.)$. While both average and max pooling operations reduce the dimensionality of the feature tensor and improve the signal-to-noise ratio, they use different spatial aggregation operations. Specifically, average pooling runs a spatial-smoothing operation over the feature tensor while max pooling preserves the most prominent features.

Example Architectures: ResNet-18 and ResNet-50

ResNet (Residual Network) architectures [He et al., 2016], such as ResNet-18 and ResNet-50, are examples of widely used ANN architectures that use

CNNs as their early layers and FCNs as their later layers. Additionally, these architectures employ residual connections to allow the parameters in the network's early layers to influence its output, allowing for stable training of very deep networks. Each residual block in a ResNet adds a shortcut/skip connection that bypasses the transformation within the block, allowing gradients from later layers to flow directly through the skip connections in the network. Mathematically, the output of a residual block can be written as:

$$\mathbf{y} = \mathcal{B}\left(\mathbf{X}\right) + \mathbf{X} \tag{2.6}$$

where:

- $\mathcal{B}(\mathbf{X})$ represents the transformations within the block, often two or more convolutional layers with batch normalization (see section 2.1.4) and activation functions, and
- $\mathbf{X}$ is the input to the block.

ResNet-18 and ResNet-50 differ in the number of layers (18 and 50, respectively) but both leverage the residual structure, making them popular choices for computer vision tasks, particularly object recognition.

2.1.3 Transformer Networks

An ANN architecture with widespread applications in image and language processing tasks is the attention-based [Bahdanau et al., 2014] transformer network [Vaswani et al., 2017]. Like CNNs, transformers apply local computations to every region of the input. However, unlike CNNs, they rely on self-attention mechanisms to perform context-aware information aggregation from other regions, thereby enabling them to capture long-range dependencies. For the rest of this subsection on transformer networks, we will refer to spatially-local regions as tokens in order to better align with the vocabulary of the attention-based networks literature.

Self-Attention Mechanism

The core component of a transformer network is the scaled dot-product attention, which calculates attention scores between elements of the input sequence. Given an input sequence of text/image tokens, we compute three matrices for each element: queries ($\mathbf{Q}$), keys ($\mathbf{K}$), and values ($\mathbf{V}$). The self-attention operation is computed as:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V} \tag{2.7}$$

$$\mathbf{A} = X\mathbf{W}_{\mathbf{A}} \quad , \quad \mathbf{A} \in \{\mathbf{Q}, \mathbf{K}, \mathbf{V}\} \tag{2.8}$$

where:

- $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{T \times d_k}$ are the query, key, and value matrices, respectively, derived from the input, $X \in \mathbb{R}^{T \times d_{in}}$ by applying linear transformations, and T is the number of tokens/elements in the input sequence, and
- d_k is the dimensionality of the keys, used to scale the dot products for stable gradients.

In this formulation, each element of the sequence calculates an attention score with every other element in the sequence, capturing long-range dependencies across all positions. The attention score can be thought of as the dot-product distance between different elements of the sequence, with different dimensions weighted differently. The softmax function, similar to Equation (2.2), normalizes these scores to sum to 1, producing a weighted average of the values, $\mathbf{V}$.

Note that the dot product inside the softmax operator can be written as follows: $\mathbf{Q}\mathbf{K}^T = \mathbf{X}(\mathbf{W}_{\mathbf{Q}}{\mathbf{W}_{\mathbf{K}}}^T)\mathbf{X}^T = \mathbf{X}\mathbf{R}\mathbf{X}^T$. Here, $\mathbf{R}$ represents a $d_{in} \times d_{in}$ affine transformation matrix that weighs the dimensions of X while computing the generalized dot product across all elements of the input sequence, i.e. $\left[\mathbf{Q}\mathbf{K}^T\right]_{ij} = \sum_{p,q} \mathbf{X}_{ip}\mathbf{R}_{pq}\mathbf{X}_{jq}$. Having a parameterized measure of distance, instead of the standard vector dot-product, enables the network to learn context-aware distance measures across elements of the input sequence. This property of learning context-aware distance measures can be further extended by adding multiple attention heads in the attention layer to learn different contexts in different heads.

Multi-Head Attention

As noted above, to compute different context-aware distance measures for aggregating long-range information, transformers employ multi-head attention. Instead of a single set of $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ matrices, multiple sets are learned, and their outputs are concatenated. The multi-head attention operation is given by:

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)\mathbf{W}_O \tag{2.9}$$

$$\text{head}_i = \text{Attention}(\mathbf{X}\mathbf{W}_Q^i, \mathbf{X}\mathbf{W}_K^i, \mathbf{X}\mathbf{W}_V^i) \tag{2.10}$$

where:

- $\mathbf{W}_Q^i, \mathbf{W}_K^i, \mathbf{W}_V^i$ are the learned query, key, and value matrices, respectively, and
- $\mathbf{W}_O$ as an additional learned projection matrix to transform the output of the attention operations.

Note that the number of learnable parameters in a multi-head attention layer is $\mathcal{O}(d_{in} \times d_k \times h)$, i.e. it scales linearly with the input embedding dimensionality, query/key/value embedding dimensionality, and the number of heads, but not with the length of the sequence. While this parameter complexity is similar to CNNs, the space complexity of the underlying computations is much higher since storing the attention matrix, i.e. the output of the softmax, requires $\mathcal{O}(T^2)$ space.

Token-Wise Feedforward Networks and Layer Normalization

Each layer in a transformer model includes a token-wise feedforward network (FFN) applied independently to each token in the sequence. The FFN is a MLP consisting of two linear transformations with a ReLU activation in between:

$$\text{FFN}(\mathbf{x}) = \max(0, \mathbf{x}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2 \tag{2.11}$$

where $\mathbf{W}_1$, $\mathbf{W}_2$, $\mathbf{b}_1$, and $\mathbf{b}_2$ are learned parameters. Additionally, transformers use layer normalization (discussed below) and residual connections (similar to ResNets) to stabilize training.

Positional Encoding

While the attention mechanism is adept at aggregating long-range dependencies, its output is generally order-invariant, i.e. the output of the attention module does not change if the tokens are permuted in the sequence. To make sure transformers are sensitive to the position of tokens in the sequence, positional encodings are added to the input embeddings. A common approach is to use sine and cosine functions of different frequencies:

$$\begin{aligned} \mathrm{PE}_{(pos,2i)} &= \sin\left(\frac{pos}{10000^{2i/d_{in}}}\right) \\ \mathrm{PE}_{(pos,2i+1)} &= \cos\left(\frac{pos}{10000^{2i/d_{in}}}\right) \end{aligned} \tag{2.12}$$

where:

- pos is the position in the sequence,
- i is the dimension, and
- d_{in} is the embedding dimensionality of the input.

For 2D inputs, like images, pos for each token can be assigned using different strategies, although there is little difference between using strategies that involve treating the image as a 1D or 2D sequence [Dosovitskiy et al., 2021, Yuan et al., 2021].

2.1.4 Additional Layers and Techniques

Popular ANNs leverage the aforementioned architectural motifs or a combination of them to build a hierarchical structure that sequentially processes an input vector to return an output vector. However, training these hierarchical stack of layers, commonly referred to as deep networks, requires additional layers that normalize the intermediate vectors to ensure that feedforward activities (or backpropagated gradients) do not blow up and lead to numerical overflows. We describe two such normalization layers below.

Batch Normalization

Batch normalization (BN) is used to stabilize and accelerate training in deep neural networks [Ioffe and Szegedy, 2015] by normalizing the input to each layer . This normalization is performed over a mini-batch of data, ensuring that each feature dimension has a mean of 0 and a variance of 1 within the batch. For a mini-batch $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$ of size m, the batch normalization process for each feature j is defined as follows:

1. Compute the mean and variance for each feature j in the mini-batch:

$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_{i,j} \qquad \sigma_j^2 = \frac{1}{m}\sum_{i=1}^{m}(x_{i,j} - \mu_j)^2 \tag{2.13}$$

2. Normalize each feature using the batch statistics:

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \epsilon}} \tag{2.14}$$

where ϵ is a small constant added for numerical stability.

3. Scale and shift the normalized features using learned parameters γ_j and β_j:

$$y_{i,j} = \gamma_j \hat{x}_{i,j} + \beta_j \tag{2.15}$$

Here, γ_j and β_j are learned parameters. Batch normalization allows the network to control the first and second-order moments of each dimension of the activations as they are sequentially processed by different layers in a network.

Layer Normalization

Layer normalization (LN) normalizes the inputs across all features for each individual data point [Ba et al., 2016]. This is particularly useful in tasks where batch statistics are less informative, such as in transformers. For an input vector $\mathbf{x} = \{x_1, x_2, \ldots, x_d\}$ of dimensionality d, layer normalization is defined as:

1. Compute the mean and variance across all features for each individual data point:

$$\mu = \frac{1}{d}\sum_{j=1}^{d} x_j \qquad \sigma^2 = \frac{1}{d}\sum_{j=1}^{d}(x_j - \mu)^2 \tag{2.16}$$

2. Normalize each feature using the computed statistics:

$$\hat{x}_j = \frac{x_j - \mu}{\sqrt{\sigma^2 + \epsilon}} \tag{2.17}$$

3. Scale and shift using learned parameters γ and β:

$$y_j = \gamma \hat{x}_j + \beta \tag{2.18}$$

As for batch normalization, γ and β are learned parameters. Layer normalization is commonly used in transformer architectures to stabilize training and improve convergence.

For more details on architectural motifs in deep learning pipelines, the reader is deferred to the Deep Learning textbook [Goodfellow et al., 2016].

2.2 Gradient Descent and Backpropagation

Having provided sufficient background on the architecture pillar of the NeuroAI framework, we now move on to the learning rule that is used to update the ANN parameters. Parameter updates in ANNs require solving the credit assignment problem, i.e. determining the influence of each parameter on a measure of the system's performance. This measure of performance, often referred to as the objective or loss function, is typically a mapping from the parameter space to a scalar value. Optimizing this objective or loss function includes two key steps: (1) assigning credit to each parameter based on its influence on the network output, then (2) updating each parameter in order to improve the loss function. These steps fit naturally into an optimization framework, wherein we want to make small changes in the parameter space that yield the greatest improvements in the loss function. In the rest of this section, we first present gradient descent as the overarching optimization framework, followed by the backpropagation algorithm that is widely used to compute parameter space gradients efficiently in deep neural networks.

2.2.1 Gradient Descent as an Optimization Framework

The desiderata of updating parameters is as follows: How can we make small changes in the parameter space that yield the greatest improvements in the loss function? This question is similar to the standard constrained optimization problem, and can be addressed with the mirror descent framework [Bubeck et al., 2015, Nemirovskij and Yudin, 1983]. Let us indicate the loss function as $\mathcal{L}(w)$, where w is a point in the parameter space. Further, let $w^{(t)}$ indicate the parameter values at t^{th} step of the optimization process. The desired optimization framework can be formulated as follows:

$$w^{(t+1)} = argmin_w \left[\mathcal{L}(w) + \frac{1}{\eta}\mathbf{D}(w \| w^{(t)}) \right] \tag{2.19}$$

where:

- $\mathbf{D}(w \| w^{(t)})$ indicates the distance function between w and $w^{(t)}$, and
- η determines the weight of the constraint, equivalent to the Lagrangian in a constrained optimization problem.

If we set the distance function to be 2-norm or Euclidean, i.e. impose a Euclidean geometry assumption on the parameter space, we can write

$\mathbf{D}(w \| w^{(t)}) = \|w - w^{(t)}\|^2$. It is possible to use other distance measures, that impose different geometry assumptions on the parameter space. For a more detailed discussion on possible distance measures and the corresponding parameter update rules, the reader is deferred to literature on the mirror descent framework.

Now, one more trick is required in order to solve the minimization problem in Equation (2.19): Taylor series expansion of a multivariate function. Namely, using a first-order Taylor Series expansion, we can expand $\mathcal{L}(w)$ around $w^{(t)}$ as follows:

$$\mathcal{L}(w) \approx \mathcal{L}(w^{(t)}) + \nabla_w \mathcal{L}(w^{(t)})(w - w^{(t)}) \tag{2.20}$$

Note that by using Equation (2.20), we are explicitly assuming that higher order terms in the Taylor expansion are less significant, compared to the zeroth and first order terms. This assumption is further justified because of the distance penalty that is imposed to ensure that $w^{(t+1)}$ is reasonably close to $w^{(t)}$, in turn ensuring that $\|w - w^{(t)}\|^p \to 0 \quad \forall p > 1$. This step can also be seen as a linear approximation of the loss function in the local neighborhood of $w^{(t)}$.

Incorporating Equation (2.20) in the optimization problem presented in Equation (2.19), we get the following:

$$w^{(t+1)} = argmin_w \left[\mathcal{L}(w^{(t)}) + \nabla_w \mathcal{L}(w^{(t)})(w - w^{(t)}) + \frac{1}{\eta}\|w - w^{(t)}\|^2 \right] \tag{2.21}$$

The resulting function is *convex* with respect to w. Therefore, we can now use the convex optimization approach to find maxima/minima of a function using derivatives, i.e. a smoothly changing convex function achieves its minimum value where its slope/derivative is zero [Boyd and Vandenberghe, 2004]. Doing so yields the standard gradient descent update that is popular in modern deep learning:

$$w^{(t+1)} = w^{(t)} - \eta \nabla_w \mathcal{L}(w^{(t)}) \tag{2.22}$$

In practice, it is important to keep in mind that the learning rate η corresponds to the weight of the distance constraint, i.e. the Lagrangian factor, in the constrained optimization framework. Using a very large learning rate implies a lower weight to the constraint, and thereby could break the first-order Taylor series approximation assumption. This, in turn, could cause the parameter update to overshoot the minimum and lead to instabilities in the learning process. On the other hand, choosing a very small η can make the optimization process slow. As a practitioner, one needs to tune the η as a hyperparameter of the optimization framework.

Gradient Descent in modern Deep Learning frameworks

In the context of modern deep learning, gradient descent is used to minimize a loss function $\mathcal{L}(\mathbf{w})$, which typically measures the discrepancy between the model's output and the desired output. We will discuss the loss function formulation in detail in the next section. The model's parameters $\mathbf{w}$ are adjusted iteratively using gradient descent (or its variants) to reduce this loss. The overall goal of the training process is to find the parameter configuration that minimizes the loss across all training samples. However, computing the loss and gradients over all training samples can be computationally expensive, especially for large networks processing high-dimensional inputs. As a result, a variant of gradient descent, called stochastic gradient descent (SGD) is used in practice. SGD computes the loss and parameter space gradients for one or few examples (randomly chosen subsets from the entire training set) at a time, and updates the parameters using this gradient information.

It is also worth mentioning that gradient descent is a first-order optimization method. In other words, it only uses the first-order derivative of the loss function with respect to the parameters to make updates in the parameter space. It is possible to use the Hessian information, i.e. second-order derivative of the loss function with respect to the parameters, to improve the convex optimization process. However, calculating the Hessian is computationally expensive, especially for networks with large number of parameters. The more popular variants use of gradient descent, e.g. Adam [Kingma and Ba, 2014], leverage information about gradients from the past optimization steps to accelerate the optimization process. For more details about these optimization strategies, the reader is deferred to literature on deep learning optimization.

2.2.2 Backpropagation

Thus far, we have assumed that we have access to the quantity $\nabla_w \mathcal{L}(.)$, i.e. the gradient of the loss function with respect to the parameters. In practice, computing the loss function itself might be intractable, making gradient computations even harder. For instance, the true performance measure is a network's mean performance on all possible input stimuli. Computing this performance requires computing a mean over the input distribution, which itself might be unknown. As a result, most machine learning setups use some notion of empirical risk as a way to measure the system's performance. In other words, the true measure of performance is often replaced with a mathematically tractable function that is typically indicative of the true performance and is easy to compute given a subset of all possible input stimuli. We will refer to this mathematically tractable version of the performance measure as $\tilde{\mathcal{L}}(.)$, which will also be referred to as the loss function that needs to be optimized.

Computing the gradient of $\tilde{\mathcal{L}}(.)$ is still a different challenge, especially in deep hierarchical networks. This problem is akin to the aforementioned credit assignment problem, where we want to compute the influence of each parameter on the system's performance. The problem of credit assignment in deep neural networks is solved using the *backpropagation*, or backward propagation of errors, algorithm.

Let us define a deep neural network as a composition of functions. For a given input $\mathbf{x}$, the output of the network can be represented as:

$$\hat{\mathbf{y}} = f_L(f_{L-1}(\ldots f_1(\mathbf{x}; \mathbf{w}_1) \ldots ; \mathbf{w}_{L-1}); \mathbf{w}_L) \tag{2.23}$$

where:

- L is the number of layers,
- f_i represents the function in the i-th layer, and
- $\mathbf{w}_i$ represents the parameters of that layer.

Backpropagation [LeCun et al., 2015] computes the gradient of the loss, $\tilde{\mathcal{L}}(w; \hat{\mathbf{y}}, \mathbf{y}_{true})$, with respect to each parameter by using the chain rule of differentiation. Here, $\mathbf{y}_{true}$ represents the desired output for input $\mathbf{x}$ and w represents all parameters of the network, $\{\mathbf{w}_1, \mathbf{w}_2 \ldots \mathbf{w}_L\}$, written as a flattened vector. Being able to compute the gradient allows for the application of gradient descent to adjust parameters and minimize the loss function. Specifically, for each layer i, we compute:

$$\frac{\partial \tilde{\mathcal{L}}}{\partial \mathbf{w}_i} = \frac{\partial \tilde{\mathcal{L}}}{\partial \hat{\mathbf{y}}} \cdot \frac{\partial \hat{\mathbf{y}}}{\partial f_L} \cdot \frac{\partial f_L}{\partial f_{L-1}} \cdots \frac{\partial f_{i+1}}{\partial \mathbf{w}_i} \tag{2.24}$$

The backpropagation algorithm proceeds as follows:

- Forward Pass: Compute output $\hat{\mathbf{y}}$ by processing the input $\mathbf{x}$ and compute empirical loss $\tilde{\mathcal{L}}(w; \hat{\mathbf{y}}, \mathbf{y}_{true})$.
- Backward Pass: Starting from the output layer, propagate the error backward through the network using the chain rule of differentiation, computing the gradients with respect to each parameter.
- Parameter Update: Use the gradients obtained to update the parameters according to gradient descent.

Efficient Gradient Calculation: Chain Rule and Layer-Wise Computation

The advantage of using backpropagation is the efficient computation of gradients through the chain rule of differentiation, allowing each layer's gradient to be calculated using the gradient of the following layer. This layer-wise approach reduces the computational complexity of calculating gradients, especially in deep networks with many layers.

For example, if we denote the output of an intermediate layer i as $\mathbf{h}_i = f_i(\mathbf{h}_{i-1}; \mathbf{w}_i)$, the gradient of the loss $\tilde{\mathcal{L}}$ with respect to $\mathbf{h}_i$ is computed as:

$$\frac{\partial \tilde{\mathcal{L}}}{\partial \mathbf{h}_i} = \frac{\partial \tilde{\mathcal{L}}}{\partial \mathbf{h}_{i+1}} \cdot \frac{\partial \mathbf{h}_{i+1}}{\partial \mathbf{h}_i} \tag{2.25}$$

Then, the gradient with respect to $\mathbf{w}_i$ can be calculated as:

$$\frac{\partial \tilde{\mathcal{L}}}{\partial \mathbf{w}_i} = \frac{\partial \tilde{\mathcal{L}}}{\partial \mathbf{h}_i} \cdot \frac{\partial \mathbf{h}_i}{\partial \mathbf{w}_i} \tag{2.26}$$

By iteratively applying the chain rule of differentiation from the output layer to the input layer, backpropagation enables credit assignment in deep neural networks. Coupled with gradient descent and its variants, backpropagation forms the workhorse of the modern deep learning revolution.

For more details on popular optimizers used in deep learning pipelines, the reader is deferred to the Deep Learning textbook [Goodfellow et al., 2016].

2.3 Self-supervised Learning (SSL)

Having covered the background for the architecture and learning rule pillars of the NeuroAI framework, we now move on to describing the loss function used to train neural networks. The loss function plays a crucial role in normatively defining the behavior that is desired from the network when processing an input. Typically, the loss function is a mathematically tractable mapping from the network's outputs to a scalar value that is indicative of the discrepancy between the current and desired outputs. This notion of discrepancy is used to learn the parameter configurations that yield the desired network behavior. In other words, by minimizing the loss function, the network adjusts its parameters to satisfy the conditions imposed by the task at hand. The most common definition of a loss function is the supervised loss, i.e. a loss function that compares the network's output for an input stimuli to a human-annotated label for the input. By learning to reduce the discrepancy between the network's predictions and human-annotated labels, the network learns representations of an input that help it have a similar response as a human to that specific input. Although supervised learning setups have led to significant progress in computer vision tasks, e.g. object recognition, they require large amounts of high-quality human-annotated data, which is expensive and challenging to acquire. An alternate paradigm of learning is *unsupervised learning*.

As opposed to supervised learning, which is limited by the availability of high-quality human-annotated data, unsupervised representation learning can be performed on vast sources of unlabeled data. The goal of unsupervised representation learning is to learn a representation space that is similar to the structure of the input stimuli space, and thereby can be leveraged to learn specific downstream behavior from few labeled samples. While early unsupervised learning approaches in deep learning relied on learning representations that can be used to reconstruct the input, recent approaches have demonstrated success by leveraging existing symmetries in the stimulus space to generate intrinsic targets, instead of requiring extrinsic human-generated targets. These approaches underpin the recent success of deep learning models in natural language processing as well as computer vision tasks, and are grouped under the *self-supervised learning (SSL)* umbrella [Balestriero et al., 2023].

In vision, SSL approaches build on the core insight that similar images should map to nearby points in the learned feature space, which is often termed the *invariance criterion*. The notion of similarity among images is defined implicitly using transformations in the pixel space that do not change the semantic content of the image. For instance, an image of a dog or cat remains as such even if it is rotated by a few degrees or translated by a few pixels or even if it is blurred slightly. This invariance to affine transformations in the pixel space or the blurring operation defines a space of operations under which representations of a deep neural network should not change. This invariance criterion serves as the desiderata of SSL methods for computer vision, which have been able to match or in some cases surpass models trained on labeled data [Balestriero et al., 2023, Chen et al., 2020]. Note that just optimizing this invariance criterion has a trivial solution: mapping the entire input space to the same representation. While this case of trivial solution, termed as representation collapse, does satisfy the invariance criterion, it is unfortunately not useful for any downstream tasks. In order to prevent representation collapse while still learning representations that are invariant to semantics-preserving transformations, SSL methods use a combination of strategies. Based on these collapse prevention strategies, SSL methods can be broadly categorized into four main categories – deep metric learning family, self-distillation family, canonical correlation analysis family, and masked image modeling.

2.3.1 Deep metric learning family

The deep metric learning family of SSL algorithms is based on the idea of encouraging representations of semantically transformed images to be more similar, compared to those of different images. In other words, these methods formulate the loss as a prediction problem, wherein a network is tasked with predicting whether two inputs are semantically similar (or not) by learning representations that are close (or far from each other). This loss, often termed as the contrastive loss, uses positive (semantically similar) and negative (semantically dissimilar) pairs as training examples. Positive pairs are obtained by applying semantics-preserving pixel-level transformations, whereas negative pairs are obtained by sampling two different images from the complete pool of unlabeled data. One common examples of an SSL algorithm that falls under the deep metric learning family is SimCLR (Simple framework for Contrastive Learning) [Chen et al., 2020]. For a set of unlabeled examples, denoted as $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ and their corresponding semantics-preserving pixel-level transformed versions, denoted as $\tilde{\mathcal{D}} = \{\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, ..., \tilde{\mathbf{x}}_n\}$, the SimCLR loss can be written as follows:

$$\begin{aligned}\mathcal{L}_{SimCLR} &= -\frac{1}{n}\sum_i log\frac{exp(\tilde{s}_{ii}/\tau)}{\sum_{j\neq i} exp(s_{ij}/\tau) + \sum_{j\neq i} exp(\tilde{s}_{ij}/\tau)} \\ &\quad -\frac{1}{n}\sum_i log\frac{exp(\tilde{s}_{ii}/\tau)}{\sum_{j\neq i} exp(\bar{s}_{ij}/\tau) + \sum_{j\neq i} exp(\tilde{s}_{ij}/\tau)} \\ s_{ij} &= f(\mathbf{x}_i)^T f(\mathbf{x}_j)/(\|f(\mathbf{x}_i)\|\|f(\mathbf{x}_j)\|) \\ \tilde{s}_{ij} &= f(\mathbf{x}_i)^T f(\tilde{\mathbf{x}}_j)/(\|f(\mathbf{x}_i)\|\|f(\tilde{\mathbf{x}}_j)\|) \\ \bar{s}_{ij} &= f(\tilde{\mathbf{x}}_i)^T f(\tilde{\mathbf{x}}_j)/(\|f(\tilde{\mathbf{x}}_i)\|\|f(\tilde{\mathbf{x}}_j)\|) \end{aligned} \tag{2.27}$$

where:

- $f(.)$ denotes the function applied by the feature encoder network on the input, and
- τ denotes the temperature hyperparameter, that determines the sensitivity of the loss to the representation similarity values (s_{ij}, $\tilde{s}_{ij}$, $\bar{s}_{ij}$).

While the deep metric learning family of algorithms has shown promising results in learning representations of images that are suitable for downstream tasks, e.g. object recognition, a major design challenge is choosing the negative samples. SimCLR relies on using other elements in a batch of inputs as negative examples, but more elaborate sampling strategies are required for learning good representations from natural videos.

2.3.2 Self-distillation family

While the deep metric family of algorithms use negative samples to avoid representation collapse, the self-distillation family of algorithms leverage an asymmetry between processing the image and its transformed version. These approaches use an online network (that processes the image) whose parameters are updated using the gradient of the loss function and a target network (that processes the transformed image) whose parameters are a running average of the online network's parameters. One common example of an SSL algorithm that falls under the self-distillation family is BYOL (Bootstrap Your Own Latent) [Grill et al., 2020], whose loss is as follows:

$$\mathcal{L}_{BYOL} = \frac{1}{n}\sum_i \|g(f(\mathbf{x}_i)) - \bar{f}(\tilde{\mathbf{x}}_i)\|^2 \tag{2.28}$$

where:

- $f(.)$ denotes the function applied by the feature encoder network on the input, and $g(.)$ is an additional predictor network applied on the outputs of the online network, and
- $\bar{f}(.)$ denotes the function applied by the target network, i.e. the exponential moving average version of the online network.

By having an asymmetry in the feature processing of the two inputs, self-distillation methods avoid representation collapse, without requiring negative examples. However, this asymmetry in the feature encoder pipeline requires extra compute and space, thereby increasing the overall hardware resources requirement of using these methods in practice.

2.3.3 Canonical Correlation Analysis (CCA) family

Another family of algorithms that refrains from using negative examples is the Canonical Correlation Analysis family of SSL algorithms. Inspired by the Canonical Correlation Analysis (CCA) framework, the core idea of this family of algorithms is to infer the relationship between two variables by analyzing their cross-covariance matrices. Two popular and closely related examples of this family of algorithms are BarlowTwins [Zbontar et al., 2021] and VICReg [Bardes et al., 2022]. BarlowTwins, which was inspired by the information-encoding ideas of the neuroscientist, Horace Barlow [Barlow et al., 1961], aims to optimize the cross-covariance structure of representations obtained from the two semantically-similar views of a set of images in order to reduce redundancies in the feature space. Variance Invarance Covariance Regularization (VICReg) is a modification of BarlowTwins that added a variance term in the loss function in order to ensure that every feature dimension has a finite variance [Bardes et al., 2022].

The BarlowTwins loss function is as follows:

$$\begin{aligned}
\mathcal{L}_{BT} &= \sum_i (C_{ii} - 1)^2 + \beta \sum_i \sum_{j \neq i} C_{ij}^2 \\
C &= \frac{1}{n-1} \sum_{i=1}^{n} (f(\mathbf{x}_i) - \overline{f(\mathbf{x})})(f(\tilde{\mathbf{x}}_i) - \overline{f(\tilde{\mathbf{x}})})^T \\
\overline{f(\mathbf{x})} &= \frac{1}{n} \sum_{i=1}^{n} f(\mathbf{x}_i) \quad , \quad \overline{f(\tilde{\mathbf{x}})} = \frac{1}{n} \sum_{i=1}^{n} f(\tilde{\mathbf{x}}_i) \qquad (2.29)
\end{aligned}$$

where:

- C_{ij} is the element of C at row i, column j, and
- n is the batch size.

The VICReg loss function is as follows:

$$\begin{aligned}
\mathcal{L}_{VIC} &= \frac{1}{n}\mu \sum_{i=1}^{n} \|f(\mathbf{x}_i) - f(\tilde{\mathbf{x}}_i)\|^2 + \frac{1}{2}\mu \left[v(f(\mathbf{x}_:)) + v(f(\tilde{\mathbf{x}}_:))\right] \\
&\quad + \frac{1}{2}\left[c(f(\mathbf{x}_:)) + c(f(\tilde{\mathbf{x}}_:))\right] \\
v(f(\mathbf{x}_:)) &= \frac{1}{d}\sum_{i=1}^{d} max(0, 1 - Stdev(f(\mathbf{x})_{:,i})) \\
c(f(\mathbf{x}_:)) &= \frac{1}{d}\sum_{i}\sum_{j\neq i}\left[C(f(\mathbf{x}_:))_{ij}\right]^2 \\
C(f(\mathbf{x}_:)) &= \frac{1}{n-1}\sum_{i=1}^{n}(f(\mathbf{x}_i) - \overline{f(\mathbf{x})})(f(\mathbf{x}_i) - \overline{f(\mathbf{x})})^T
\end{aligned} \tag{2.30}$$

The BarlowTwins and VICReg loss formulations ensure that the two semantically-similar views are represented similarly but also avoid collapse by encouraging different feature dimensions to capture different attributes of the input. While the CCA family of algorithms refrains from using negative examples or asymmetrical feature pipelines, they generally require large enough batch sizes to ensure reliably estimates of the covariance (or correlation) matrices.

2.3.4 Masked Image modeling

Inspired by recent progress in masked language modeling, more recent SSL algorithms have proposed reconstructing a masked image patch from the rest of the image. These approaches, which are modern variants of the prominent early unsupervised learning approaches of reconstructing an image from its noisy or corrupted version, heavily leverage the transformer architecture to demonstrate promising performance at large-scale pretraining tasks [He et al., 2016]. Recent successful SSL methods, often referred to as Joint Embedding Predictive Architectures (JEPA), have used a mix of masked image modeling and self-distillation strategies to achieve competitive performance on a wide variety of vision tasks [Assran et al., 2023, Oquab et al., 2024]. For more details on the historical origins of each family of SSL algorithm as well as practical deployment tips, the reader is deferred to the SSL cookbook [Balestriero et al., 2023].

2.4 Eigen and Singular Value Decomposition

This chapter has so far discussed key background material for the key pillars of the NeuroAI framework. Before diving into the next chapters, we present a short note on two linear algebra techniques that are essential in understanding the learned representations and learning dynamics under gradient-based optimization in deep neural networks [Strang, 2022]. These two techniques, specifically focusing on matrix decompositions, are *eigendecomposition* and *singular value decomposition* (SVD). These decompositions will frequently be used in the subsequent chapters to characterize the spectral properties of weight matrices, learned representations, optimization dynamics or generalization properties of ANNs.

2.4.1 Eigendecomposition

Eigendecomposition is a matrix factorization method that applies specifically to square matrices. Given a square matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, the eigendecomposition represents $\mathbf{A}$ in terms of its eigenvalues λ_i and eigenvectors $\mathbf{v}_i$ as follows:

$$\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1} \tag{2.31}$$

where:

- $\mathbf{V}$ is a matrix with eigenvectors $\mathbf{v}_i$ as columns, and
- $\mathbf{\Lambda}$ is a diagonal matrix containing the eigenvalues λ_i.

Note that the eigenvectors form an orthonormal basis set, i.e. $\mathbf{v}_i^T\mathbf{v}_i = 1 \ \forall i$ and $\mathbf{v}_i^T\mathbf{v}_j = 0 \ \forall i \neq j$

Eigenvalues indicate the magnitude of scaling in the direction of each eigenvector, such that when the matrix $\mathbf{A}$ is multiplied with a vector that is directed along eigenvector $\mathbf{v}_i$, it is scaled by a factor of λ_i. When applied to a covariance matrix of representations, the eigenvectors corresponding to the largest eigenvalues are the orthogonal directions in the representation space that capture most of the variance.

While analyzing learning dynamics under gradient-based optimization, eigendecomposition of the Hessian matrix (second-order derivative of the loss with respect to the parameters) indicates the curvature of the loss landscape. Large eigenvalues in the Hessian indicate directions of high curvature, i.e. small updates along those directions in the parameter space lead to large changes in the loss function, whereas small or zero eigenvalues reveal flat directions, i.e. updates along those directions in the parameter space have little effect on the loss function. Eigendecomposition is a fundamental tool in understanding the manifold structure of representations and gradient-based optimization properties, both of which are discussed in detail in the subsequent chapters.

2.4.2 Singular Value Decomposition (SVD)

Singular Value Decomposition (SVD) generalizes eigendecomposition to any $m \times n$ matrix, which need not be square. Given a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, the SVD represents $\mathbf{A}$ as a product of three matrices:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \tag{2.32}$$

where:

- $\mathbf{U} \in \mathbb{R}^{m \times m}$ and $\mathbf{V} \in \mathbb{R}^{n \times n}$ are orthogonal matrices containing the left and right singular vectors of $\mathbf{A}$, and
- $\mathbf{\Sigma} \in \mathbb{R}^{m \times n}$ is a diagonal matrix with non-negative singular values σ_i on the diagonal.

Similar to the eigenvalues, these singular values indicate the extent to which a vector directed along a particular singular vector is scaled when multiplied by $\mathbf{A}$.

In ANNs, SVD is often used to analyze weight matrices and dynamical systems analysis of gradient descent. For instance, SVD can be used to decouple the multivariate partial differential equations obtained when analyzing the dynamics of gradient descent in a simple fully-connected ANN. This decoupling effect enables tractable analysis of the emergent weight matrices over training, thereby providing insights into the emergent representations learned by the network as well as potential pathologies that might occur over the course of training. These applications are investigated in detail in the subsequent chapters, wherein the learning dynamics of supervised learning and SSL is analyzed.

Both eigendecomposition and SVD thus serve as powerful tools in theoretical analyses of ANNs, offering insights into the learning trajectory and potential pathologies along with factors affecting them. These theoretical insights are key to designing practical interventions that avoid pathologies in training pipelines, and lead to efficient learning strategies for ANNs.

Part II

Representations in brains and ANNs

3 Preface to Part II

Beginning with the first pillar of the NeuroAI framework, learned representations, Part II will investigate the characteristics of representations that are shared across brains and ANNs. As discussed earlier, intermediate representations offer an observable window into the system's underlying mechanisms, shedding light on how an input stimulus is processed by different stages of the system eventually leading to behavior. To this end, similarities in the representation characteristics between brains and ANNs can indicate potential similarities in the computational motifs employed by the two systems. Therefore, in Part II, we ask, *Do brains and ANNs learn representations that have similar geometry?*

At this point, it might be helpful to clarify what is meant by representations in an ANN and how representation geometry can help reveal shared computational motifs: First, we define representations of an input in a particular layer of an ANN as the set of activations of all units in that layer, written in a vectorized form. So, representations of each input are denoted as a vector if $\mathbb{R}^d$ space, where d is the number of units in the layer. Second, representation geometry is a key quantity in determining how information about the stimuli is distributed in this high-dimensional $\mathbb{R}^d$ space.

Representation geometry is key to understanding the tradeoff between two fundamental properties essential for systems that map inputs to behaviors – capacity and robustness. Capacity of a representation space indicates the ability to discriminate between any two inputs, and a higher capacity implies a larger discriminability between similar inputs, thereby allowing more complex behavior. Robustness of the representation space, however, indicates how drastically the representation of an input changes when it is slightly corrupted. Higher degree of robustness implies smaller changes to representations, thereby allowing reliable decision making. Higher capacity could often lead to overfitting to few examples, while higher robustness could lead to underfitting due to reduced sensitivity to differences in the input stimuli.

Part II of this thesis will address this tradeoff between these two properties in detail, as well as how representation geometry serves as an indicator of this balance. Since this tradeoff is fundamental to any intelligent system (see Appendix Section 2 of [Stringer et al., 2019] for a detailed discussion), similarities in the representation geometries in brains and ANNs are indicative of similar computational principles being employed by the two systems while processing and mapping the structure of the naturalistic stimuli space.

* * *

This work was published at NeurIPS 2022 and can be cited as Agrawal, K.K.[*], Mondal, A.K.[*], Ghosh, A.[*], and Richards, B.A. α-ReQ: Assessing Representation Quality in Self-Supervised Learning by measuring eigenspectrum decay. NeurIPS 2022.

4

α-ReQ : Assessing representation quality by measuring eigenspectrum decay

α-ReQ : Assessing representation quality by measuring eigenspectrum decay

Kumar Krishna Agrawal[†]
UC Berkeley
CA, USA

Arnab Kumar Mondal[†]
Mila & McGill University
Montréal, QC, Canada

Arna Ghosh[†]
Mila & McGill University
Montréal, QC, Canada

Blake A. Richards
Mila, Montreal Neurological Institute & McGill University
Montréal, QC, Canada
Learning in Machines and Brains Program, CIFAR
Toronto, ON, Canada

Abstract

Self-Supervised Learning (SSL) with large-scale unlabelled datasets enables learning useful representations for multiple downstream tasks. However, efficiently assessing the quality of such representations poses nontrivial challenges. Existing approaches train linear probes (with frozen features) to evaluate performance on a given task. This is expensive both *computationally*, since it requires retraining a new prediction head for each downstream task, and *statistically*, which requires task-specific labels for multiple tasks. This poses a natural question, *how do we efficiently determine the "goodness" of representations learned with SSL across a wide range of potential downstream tasks?* In particular, a task-agnostic statistical measure of representation quality that predicts generalization without explicit downstream task evaluation would be highly desirable.

In this work, we analyze characteristics of learned representations $\mathbf{f}_\theta$ in well-trained neural networks with canonical architectures & across SSL objectives. We observe that the eigenspectrum of the empirical feature covariance $\mathrm{Cov}(\mathbf{f}_\theta)$ can be well approximated with the family of a power-law distribution. We analytically and empirically (using multiple datasets, e.g. CIFAR, STL10, MIT67, ImageNet) demonstrate that the decay coefficient α serves as a measure of representation quality for tasks that are solvable with a linear readout, that is, there exist well-defined intervals for α where models exhibit excellent downstream generalization. Furthermore, our experiments suggest that key design parameters in SSL algorithms, such as BarlowTwins [1], implicitly modulate the decay coefficient of the eigenspectrum (α). As α depends only on the features themselves, this measure can be used for compute-efficient model selection with hyperparameter tuning for BarlowTwins.

1 Introduction

The recent success of self-supervised learning (SSL) has changed the landscape of deep learning significantly. With well-engineered architectures and training objectives, SSL models learn useful representations from large datasets without relying on any labels [1, 2, 3]. Despite this progress, quantifying the *representation quality* for models trained with SSL is still an open problem. The most obvious (and common) solution is to assess performance of models using these representations

[†]These authors contributed equally to this work

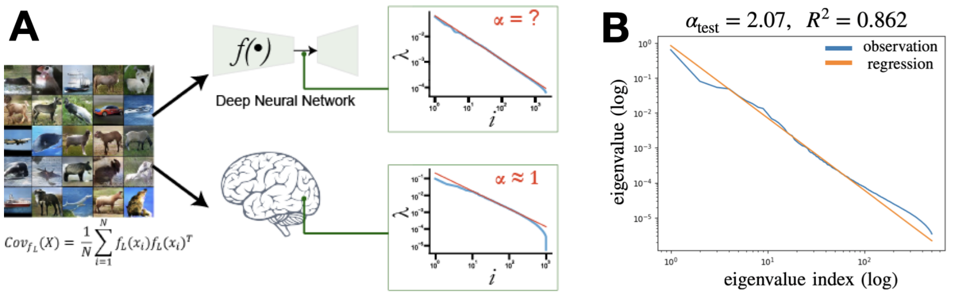


Figure 1: **(A)** One approach to evaluate representation quality is tracking the eigenspace of feature covariance matrix $\Sigma_n(\mathbf{f}) = 1/n \sum_{i=1}^{n} \mathbf{f}(x_i)\mathbf{f}(x_i)^\top$. Analysing populations of neural activation [6, 7], suggests that the eigenspectrum for $\Sigma_n(\mathbf{f})$ can be approximated by power-law, where $\lambda_i \propto i^{-\alpha}$. **(B)** For a ResNet-50 model pretrained on ImageNet, we extract activations from an intermediate layer. We plot the eigenvalue spectrum $\{\lambda_1, ..., \lambda_d\}$ for covariance matrix Σ_n`(resnet50(feats='block4'))` in `log-log` scale. Further, fitting a linear regressor we estimate the decay coefficient $\hat{\alpha}$.

on downstream tasks. However, if the goal is general representations that can be used across many domains, then this either requires a large investment of time and energy to be done well (in order to assess performance on many tasks and datasets). Alternatively, if models are assessed on only a small number of datasets and tasks, then it is hard to be confident in the assessment. Thus, we are left with a question: Can we assess the quality of learned representations without explicitly evaluating the performance on downstream tasks? To answer this question, we must formally define the "quality" of representations and subsequently examine statistical estimators that measure this property without needing downstream evaluation. Beyond theoretical interest, such a metric would be highly desirable for model selection and also useful for designing new SSL algorithms.

In search of such a metric, we turn our attention to one of the more efficient learning machines in existence – the mammalian brain. The hierarchical and distributed organization of neural circuits, especially in the cortex, provides neural representations that support a wide array of behaviours across many domains. For example, representations in primary visual cortex (V1) of mammalian brains are used by animals to support downstream behaviours ranging from object categorization to movement detection and motor control [4, 5]. This berth of downstream uses of V1 representations is desirable for artificial vision systems trained with SSL. Thus, understanding the properties of representations in V1 is a reasonable starting point for seeking a general metric of representation quality.

Recent breakthroughs in systems neuroscience enable large-scale recordings of neural activity. By recording and analyzing the response to visual stimuli, [6, 7] find that activations in the mouse and macaque monkey V1 exhibit a characteristic information geometric structure. In particular, these representations are high-dimensional, yet the amount of information encoded along the different principal directions varies significantly. Notably, this variance (computed by measuring the eigenspectrum of the empirical covariance matrix) is well-approximated by a power-law distribution with decay coefficient ≈ 1, i.e., the n^{th} eigenvalue of the covariance matrix scales as $1/n$.

Motivated by these results, we explore the use of the decay rate of the empirical eigenspectrum to characterize representation *quality* in neural networks trained with SSL (Fig. 1a). Across diverse model architectures, pretraining objectives, and downstream classification tasks, we empirically observe an eigenspectrum decay in the representation covariance matrix that roughly follows a power-law distribution (see e.g. Fig. 1b). We also find that the coefficient of this decay, denoted by α, is informative of downstream generalization performance. Importantly, α can efficiently be calculated without any labels, and thereby, could be incorporated into existing SSL pipelines for efficient model selection on fixed compute budget. Our core contributions in this paper are:

1. α **as a potential metric** We observe that canonical pretrained architectures have representations which loosely conform to a power-law distribution in their covariance eigenspectrum. Under an assumption of a power-law distribution, we prove that the convergence rate and upper bound on generalization error are related to decay-coefficient (α) of the corresponding power-law distribution, with best values of α being neither too large nor too small.

2. α **as a label-free measure for representation quality**: We empirically validate our theoretical results by demonstrating a relationship between α and downstream generalization. In particular, we find that either too high or too low an α value implies poor generalization, both in-distribution and out-of-distribution. Generally, the best representations are those where α is in a range that is close to 1, as observed in V1 of the real brain. Furthermore, these results hold irrespective of the choice of network architecture or pretraining objective.
3. α **for model selection in SSL**: We establish α as a reliable metric for model selection in a specific case of SSL, Barlow Twins [1]. Notably, we show that α allows us to identify model hyper-parameters that lead to representations that generalize well without any labels, more so than the actual loss function used to train the network.

Altogether, our results show that α is a promising task/architecture/data agnostic metric for assessing representation quality in SSL. We publicly release our results and code [†].

2 Theoretical Framework to Assess Representations in SSL

We are interested in evaluating the quality of high dimensional representations learned by neural networks. Formally, we consider the overparameterized setting, with datasets $\mathcal{X} \subset \mathbb{R}^d$, and learned mappings $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^D$ such that $\mathbf{f}(\mathbf{x})$ is a vector of D-dimensional features.

We consider DNNs as our function approximators, where each architecture implicitly defines a function class $\mathcal{F} = \{\mathbf{f}_\theta : \theta \in \Theta\}$ where $\Theta \subset \mathbb{R}^p$ is the feasible set of model parameters (e.g bounded Θ, $[-B, B]^p$ for some $B \in \mathbb{R}$). The search for good representations poses the following optimization problem: with a dataset $\mathcal{D}_{\text{pretrain}}$ from some data distribution $\mathbb{P}_{\text{pretrain}}$ (potentially with labels), search for *optimal* parameters θ^* such that with pretrain objective $\mathcal{L}_{\text{pretrain}}(\mathbf{f}, \mathcal{D})$

$$\theta^* = \arg\min_{\mathbf{f}_\theta \in \mathcal{F}} \mathcal{L}_{pretrain}(\mathbf{f}_\theta, D_{\text{pretrain}}) \tag{1}$$

The above optimization problem is usually non-convex, and often uses gradient based optimizer to find an approximate solution $\hat{\theta}$. Typically, to measure the quality of $f_{\hat{\theta}}$, researchers evaluate the quality of representations on a downstream task $\mathcal{D}_{\text{downstream}}$ with a linear readout using some metric $\mathcal{R}(\texttt{linear}(\mathbf{f}_{\hat{\theta}}), D_{\text{downstream}})$. A concrete example is pretraining a VGG-16 model ($\mathcal{F}$) on ImageNet dataset (D_{pretrain}) and evaluating the learned representations using classification accuracy ($\mathcal{R}$) by learning a linear classifier on the MIT67 dataset ($D_{\text{downstream}}$). For the rest of the paper, we denote feature maps as $\mathbf{f}_\theta(x) \in \mathbb{R}^D$ where $\mathbf{f}_\theta : \mathcal{X} \to \mathbb{R}^D$, and the readout network as $\mathbf{g}_\phi : \mathbb{R}^D \to \mathbb{R}^k$, where k is the target dimensionality. For simplicity of analysis, we consider linear readouts unless explicitly mentioned, i.e. $\mathbf{g}_\phi(x) = x^T\phi$.

2.1 Covariance estimation and eigenspectrum

For a parameterized function $\mathbf{f}_\theta : \mathcal{X} \to \mathbb{R}^D$ (assume centered), the (n-sample) empirical covariance matrix is defined as:

$$\Sigma_n(\mathbf{f}_\theta) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{f}_\theta(x_i)\mathbf{f}_\theta(x_i)^T$$

where x_i is the i^{th} sample. The eigenspectrum of $\Sigma_n(\mathbf{f}_\theta)$ informs us about the variance explained by each principal component of the space spanned by representations $\mathbf{f}_\theta(\mathbf{x})$. Using the spectral decomposition theorem on symmetric matrices, $\Sigma = U\Lambda U^T$, where Λ is a diagonal matrix with nonnegative entries, and U is a matrix whose columns are the eigenvectors of Σ. Without loss of generality we assume that $\lambda_1 \geq \lambda_2 ... \geq \lambda_m$, where $m = \min(n, D)$ is the rank of $\Sigma_n(\mathbf{f}_\theta)$.

2.2 Eigenspectrum Decay in Deep Representation Learning

Recent work in characterizing representation structure in canonical DNNs has demonstrated that the covariance eigenspectrum roughly follows a power-law [8, 9, 10]. Specifically, the eigenspectrum

[†] https://github.com/kumarkrishna/fastssl

of a covariance matrix follows a power-law distribution $PL(\alpha)$, or *zeta distribution* if for $\lambda_j \in [\lambda_{min}, \lambda_{max}]$, the eigenvalues λ_j are all nonnegative, and

$$\lambda_j \propto j^{-\alpha}$$

for some $\alpha > 0$. Here, α is the *slope* of the power law, and is referred to as the *coefficient of decay* of the eigenspectrum. Intuitively, small α (typically $\alpha \leq 1$) suggests a dense encoding, while a high α (rapid decay) corresponds to a sparse encoding.

Insights from High-Dimensional Linear Regression : Recent work in theoretical machine learning has connected bounds on generalization error for a linear regression problem to the eigenspectrum of the feature covariance matrix. Specifically, in the infinite-dimensional setting with $D \to \infty$, [11] studied the linear regression setting with Gaussian features, and proved that if the eigenspectrum follows a power-law distribution (up to polylogarithm factors), the min-norm solution provides good generalization performance iff $\alpha = 1$. The asymptotic regime of infinite width is an excellent framework to study theoretical properties of DNN representations. However, despite having a large number of parameters, practical DNNs always possess finite dimensional representations, making it important to investigate the implications of such results in finite width models. In particular, for the finite dimensional setting we try to answer: *Does α sufficiently larger or smaller than 1, still allow efficient learning and strong generalizability?*

To answer this question, we narrow our focus to gradient-based optimization techniques usually used to train DNNs. In particular, from the optimization perspective, Advani *et al.* showed that for deep linear regression in high dimensions, the time required for training and the steady-state generalization error are both $\mathcal{O}(\frac{1}{\lambda_{min}})$ [12]. A key difference from our work is that they assume the inputs are drawn from an isotropic Gaussian distribution. Instead, we investigate the generalization of linear regression on $\mathbf{f}_\theta(x)$, which has a power law structure in its covariates. We show that the training convergence time grows exponentially with α using the following theorem:

Theorem 2.1. *Let $\hat{y} = \mathbf{f}_\theta(x)^T\psi$ be an overparameterized linear regression problem where ψ is learned using gradient descent in order to optimize the training error, $\mathbb{E}_{x,y}[(y - \mathbf{f}_\theta(x)^T\psi)^2]$, where $(x, y) \sim \mathcal{D}_{train}$. If we assume a power-law distribution in the eigenspectrum of representations at $\mathbf{f}_\theta$, i.e. $\lambda_n = \frac{c}{n^\alpha} \quad \forall n \geq n^*$, where $n^* \in \{1, 2...N\}$, then the time required by gradient descent to minimize the training error, $T_{convergence} = \mathcal{O}(N^\alpha)$ where N is number of training samples.*

The outline of the proof builds on key results relating to gradient descent dynamics from [13]. We show that gradient descent updates, when ψ is initialized to 0, yield a recursive relation for $\psi(k)$, i.e. ψ after k update steps. Plugging this relation in the gradient formulation, we show that the update step length along the n^{th} principal direction of $\mathbf{f}_\theta(x)$ shrinks exponentially with a decay rate proportional to λ_n. Therefore, the time to convergence in training is controlled by the smallest eigenvalue which, by design, follows the power law. In sum, Theorem 2.1 provides an explanation against arbitrarily large values of α.

Next we show that α can't be arbitrarily small as the upper bound on generalization error is higher for smaller values of α.

Theorem 2.2. *Let $\hat{y} = \mathbf{f}_\theta(x)^T\psi$ be a linear regression problem as before. Let us further assume that $\mathbf{f}_\theta(x)\ \forall x \sim \mathcal{D}_{train}$ is a representative subset of the inputs from true data distribution: $(x, y) \sim \mathcal{D}$. Assuming a power-law distribution in the eigenspectrum of representations at $\mathbf{f}_\theta$, i.e. $\lambda_n = \frac{c}{n^\alpha} \quad \forall n \geq n^*$, where $n^* \in \{1, 2...N\}$, the generalization error after T weight update steps, $\mathcal{G}(T)$ is:*

$$\mathcal{G}(T) := \mathbb{E}_{x,y\sim\mathcal{D}}[(y - \mathbf{f}_\theta(x)^T\psi)^2] \quad \leq \mathcal{O}\left(r_{\hat{d}}(\Sigma(\mathbf{f}_\theta))\right)$$
$$\textit{where} \qquad r_{\hat{d}}(\Sigma(\mathbf{f}_\theta)) = \frac{\sum_{i=\hat{d}}^m \lambda_i}{\sum_{i=1}^m \lambda_i} = \frac{\sum_{i=\hat{d}}^m i^{-\alpha}}{\sum_{i=1}^m i^{-\alpha}} \qquad (2)$$

Here, $m = min(n, D)$ is the rank of $\Sigma_n(\mathbf{f}_\theta)$.

Notably, $r_{\hat{d}}(\Sigma(\mathbf{f}_\theta))$ can be thought of as a measure of effective rank of the covariance matrix and grows with decreasing values of α. Therefore, the upper bound for generalization error is higher for lower α. Taken together, Theorems 2.1 and 2.2 suggest that α can neither be too high nor too low and there exists a tradeoff region which results in efficient learning and strong generalizability, where these representations form a good basis for gradient-based optimization on downstream task performance.

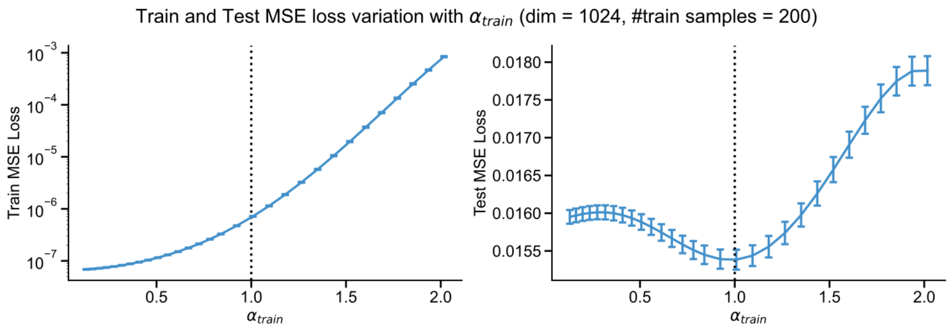


Figure 2: Overparameterized linear regression, with inputs drawn from Gaussian distribution with power-law in the covariance matrix. Models are trained with gradient descent with initialization $\psi_0 = \mathbf{0}$. Note that $\alpha > 1$ particularly suffers from high train, test MSE loss, with low generalization error with $\alpha \approx 1$.

Another motivating example Before exploring the link between α and generalization performance of deep networks, we validate our theorems in a simple linear regression setting.

We first consider its relationship to the finite dimensional regression setting as in [11]. Specifically, Theorem 6 of their paper states that the conditions for "benign overfitting", wherein a model can perfectly fit noisy training data without any subsequent loss of performance on testing data, may be looser in finite dimensions as opposed to the necessary and sufficient condition of $\alpha = 1$ in infinite dimensions. To empirically test this in the finite high dimensional setting, we examine linear least squares regression using different covariate structures for the input data. Formally, we consider covariates $\{x_i\}_{i=1}^N$, such that $x_i \in \mathbb{R}^d$ is sampled from a Gaussian distribution with covariance structure $\Sigma = \text{diag}\{\lambda_1, ...\lambda_d\}$ where $\lambda_j \sim PL(\alpha)$, i.e $\lambda_j \propto cj^{-\alpha}$. We assume access to the corresponding labels $\{y_i\}_{i=1}^N$ generated under a teacher function θ^*, such that $y_i = x_i^T\theta^* + \epsilon_i$. We find that in this scenario there is a clear relationship between the proximity of α to 1 and the presence of benign overfitting. As shown in Fig. 2, when α is close to 1, the training loss is low, but the validation loss is also low. Thus, when α is close to 1, the generalization properties are at their best. This example thus provides another hint that α is a potential measure for how well a model will be able to generalize.

3 Experimental Setup

Our theoretical results suggest the following: for representations with power-law characteristics, the decay-coefficient (α) effects the *adaptivity*, where the convergence rate of linear regression with gradient descent scales exponentially $\mathcal{O}(n^\alpha)$ with α. On the other hand, if trained sufficiently long, the upper bound for generalization error improves with larger α. Together, these items imply that

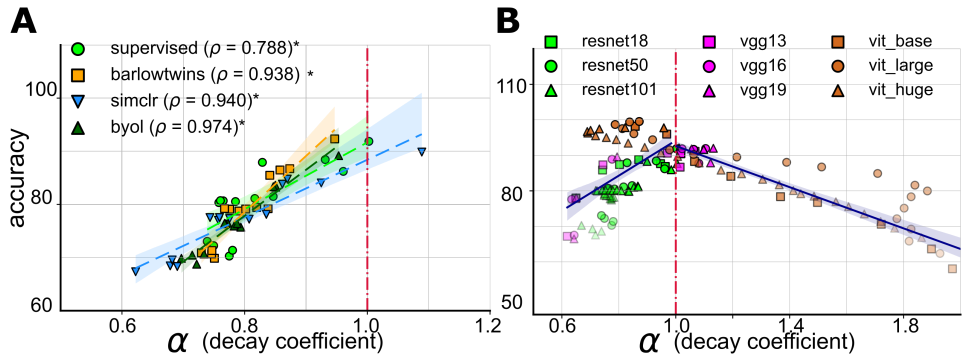


Figure 3: α is predictive of out-of-distribution object recognition performance. α is strongly correlated to object recognition performance on STL10 across different architectures and pretraining loss functions. More transparent (opaque) points indicate representations obtained from earlier (later) layers of a network.

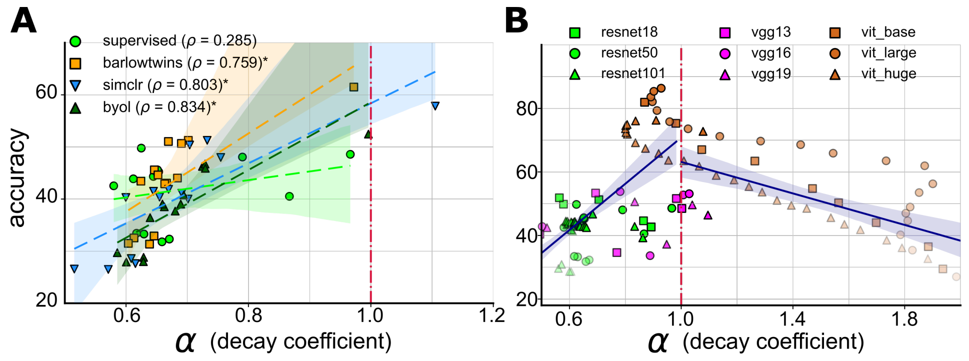


Figure 4: α is predictive of out-of-distribution object recognition performance. α is strongly correlated to object recognition performance on MIT67 across different architectures and pretraining loss functions. More transparent (opaque) points indicate representations obtained from earlier (later) layers of a network.

if one wants to use gradient descent to train a downstream task then α for the representations used should be neither too large nor too small, i.e. high quality representations are those with α in a "Goldilocks" zone. Building on these insights, we now design experiments to empirically study the potential for α to be used as a representation quality metric. We do so by measuring the relationship between α and the standard metric of representation quality, i.e. downstream task generalization performance. In this work, we restrict our scope to vision tasks, specifically object classification and scene recognition. In particular, our experiments:

- **measure the eigenspectrum decay & generalization** : We systematically evaluate *in-distribution* and *out-of-distribution* generalization, and simultaneously, we also measure α, across diverse choices for network architectures, pretraining learning objectives, and across multiple datasets. We then look at whether α values near some neighbourhood around 1 are indeed predictive of representations that generalize well to downstream tasks.
- **explore α as a metric for model-selection in SSL:** Measuring α is label-free and computationally inexpensive, as it requires a single forward-pass on the downstream dataset. We thus ask, is α predictive of generalization capabilities for current SSL algorithms, and if so, can it be used for model selection? We run extensive ablations on design of the BarlowTwins [1] algorithm in an effort to answer this question.

3.1 Measuring eigenspectrum Decay & Generalization

Evaluation Protocol: To measure α for the learned representation $\mathbf{f}(x_i)$, we first extract features from intermediate layers of a DNN pretrained on $\mathcal{D}_{pretrain}$ and estimate the corresponding covariance matrix $\Sigma_n(\mathbf{f})$. Next, we compute the full set of numerical eigenvalues, and estimate α by a fitting a linear model on the eigenspectrum in `log-log` scale. Our pretrained models are taken from PyTorch Hub [14] and `timm` [15]. Given that we observed no significant difference between the observed α values in the train and test sets, we refer to this empirical estimate as the α for the dataset.

To estimate the capacity of intermediate representations in solving the downstream task, we train a linear readout layer, $\mathbf{g}(.)$, from representations to target logits. Intuitively, this comes down to establishing a relationship between the manifold geometry and linear separability of representations [16]. Thereafter, we observe the correlation between estimated α and the linear readout performance. Notably, we also tried non-linear $\mathbf{g}(.)$ and observed a similar trend in results (see Appendix B.1).

Inductive bias of Model Architecture In this section, we investigate the relationship between α of the representation covariance matrix and object/scene recognition performance in DNNs with different backbones and the role of depth. In order to do so, we examine varying depth configurations within network architectures across three generations of models on the STL10 and MIT67 dataset [17]. We choose both object and scene recognition task as they fundamentally differ in the nature of the features that must be extracted to perform well. For scene recognition, the model focuses on global features in the entire image, while for object recognition, the model focuses on local features of the image containing the object[18].

We examine deep Convolutional Neural Networks (CNNs) without any residual connection as our first family of models. Specifically, we choose three different configurations of VGG-Net [19], namely VGG-13, VGG-16 and VGG-19. We inspect representations that are input to the dropout and MaxPool layers during the forward pass of the network. Second, we consider Deep Residual Networks [20] which are widely used in computer vision. We inspect the representations that are input to each of the residual blocks as well as the Adaptive average pool in ResNet-13, ResNet-50 and ResNet-101 during their respective forward passes. Finally, owing to the recent success of transformers in object recognition tasks we consider Vision Transformers (ViT) [21] as the third family of models, namely ViT-Base/8 , ViT-Large/16 and ViT-Huge/14. Unlike VGG and ResNet, we only look at features in the intermediate layers because it summarizes the entire input image and is used in practice for class prediction. For all model architectures, we use the weights obtained from pretraining on ImageNet.

Fig. 3 and Fig. 4 illustrate the relation between performance and α for all the nine architectures across intermediate layer representations, as described above. We found that, while most intermediate representations in CNNs (with or without residual connections) exhibit $\alpha < 1$, representations in ViTs mostly exhibit $\alpha > 1$. Nevertheless, representations extracted from the deepest layers of all the models exhibit α close to 1, irrespective of the total depth of each model (see Appendix B). Furthermore, the performance on downstream task increases with depth. This is unsurprising because all networks were trained to perform object recognition on ImageNet [22] and thereby would have leveraged hierarchical processing to learn features that are tuned towards object recognition. Surprisingly enough, we observe a strong significant correlation between α and performance on the STL10 dataset, i.e. a different data distribution than the training dataset, across layers and model architectures ($\rho = -0.922$, $^{*}p < 0.05$ for representations exhibiting $\alpha > 1$ and $\rho = -0.922$, $^{*}p < 0.05$ for representations exhibiting $\alpha < 1$). It is worth noting here that the correlation was weaker for the earliest layers of each model. We believe that early layers learn more task invariant features that reflect the statistics of natural images [23, 24] and therefore lack task relevant information in their representations. Taken together, this observation confirms our hypothesis that α is a good indicator of out-of-distribution generalization performance when representations possess task relevant information. Thus, α has a *necessary but not sufficient* relationship with generalization (Appendix C).

Learning objective In this section, we first aim to understand how the α value changes across the layers of a fixed architecture DNN when trained with different learning objectives. We take a ResNet-50 model [20] pre-trained using three different SSL algorithms, namely SimCLR [2], BYOL [25] and Barlow Twins [1], and the supervised learning loss objectives on ImageNet-1k[22] dataset. We use a similar procedure as before to extract representations from the network and estimate α.

Similar to results in the previous section, all networks irrespective of the pretraining loss function, exhibit α closer to 1 in the deeper layers in contrast to intermediate layers (see Appendix Fig. 9). This surprising result indicates that although the pre-training loss function was different, representations extracted from deepest layers are reflective of the object/scene semantics in natural images. Furthermore, Fig. 3 and Fig. 4 illustrate the strong correlation between α and generalization performance on STL10 and MIT67 across all pre-training loss functions. Together with results from the previous section, we validate our hypothesis that the representations that demonstrate good out-of-distribution generalization performance are characterized by α in the neighbourhood of 1. This suggests that high quality representations are indeed those with α in this region.

3.2 Label-agnostic metric for Model selection

With empirical evidence of α being a good measure for generalization, we now wish to study its suitability as a metric for identifying the best among models pretrained with SSL algorithms. A label-free metric like α could be beneficial when we don't have access to the downstream task annotations, and the SSL loss is not useful to distinguish models with good generalization performance (see Fig. 5). To investigate this, we exhaustively ablate the relationship between α and model performance across a wide range of hyper-parameters for a representative non-contrastive SSL algorithm.

Current SSL algorithms struggle with dimension collapse (characterized by large α), due to pathologies in training dynamics. To study α across a wide-range of values, for our model selection experiments, we pick the *Barlow Twins*[1], a non-contrastive SSL algorithm that explicitly optimizes

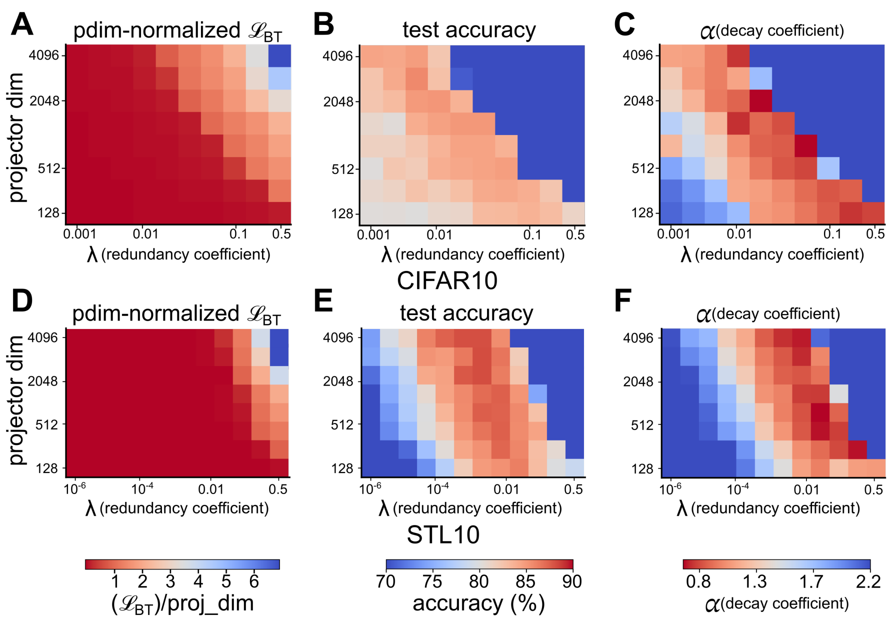


Figure 5: α as a metric to inform model selection. The SSL loss (training for same number of gradient steps) is no longer useful to distinguish models with superior downstream performance. However, decay coefficient α shows strong correspondence to downstream test accuracy over a large hyperparameter ranges. Measuring in-distribution generalization for (A-C) BarlowTwins trained and evaluated on CIFAR10. (D-F) BarlowTwins trained and evaluated on STL10.

to avoid dimension collapse. In particular, the Barlow Twins learning objective ($\mathcal{L}_{\mathrm{BT}}$) proposes imposing a *soft-whitening* constraint :

$$\mathcal{L}_{\mathrm{BT}} = \underbrace{\sum_{i=1}^{d}(1-\mathcal{C}(\mathbf{f}_\theta)_{ii})^2}_{\text{invariance}} + \lambda \underbrace{\sum_{i=1}^{d}\sum_{j\neq i}^{d}\mathcal{C}(\mathbf{f}_\theta)_{ij}^2}_{\text{redundancy-reduction}} \quad \text{s.t. } \mathcal{C}(\mathbf{f}_\theta)_{ij} = \frac{\sum_n \mathbf{f}_\theta(x^A)_i \mathbf{f}_\theta(x^B)_j}{\sqrt{\sum_n \mathbf{f}_\theta(x^A)_i^2 \sum_n \mathbf{f}_\theta(x^B)_j^2}} \tag{3}$$

Notably with sufficient large λ, the model would impose an $\alpha = 0$ constraint on the representations [26]. Despite the intuitive connection between α and λ, it is unclear whether this relationship holds across a wide-range of values for model hyperparameters. To empirically establish this connection, we vary λ (redundancy coefficient), projection head dimensionality and learning rate, and train a ResNet50 encoder using *Barlow Twins* learning objective. We provide the results for CIFAR10 [27] and STL10 [28] in Fig. 5 demonstrating that α is a strong indicator of in-distribution generalization performance across a large range of hyperparameter ranges of $\mathcal{L}_{BT}$. We defer the reader to Appendix Fig. 14 & Fig. 15 for similar trends in optimization hyperparameters. Furthermore, α is predictive of out-of-distribution generalization performance in these settings (see Appendix B.3). The same relation also holds in contrastive SSL frameworks, specifically for SimCLR [2] (see Appendix B.4).

Model Selection on a Compute Budget: With a constrained compute budget, using α for model selection is significantly cheaper than evaluating the representations on a suite of downstream tasks. To illustrate this, consider the standard alternative of training a linear classifier on the representations and evaluating test accuracy. Compared to training a linear probe, which requires multiple epochs of forward & backward passes through the

Table 1: We report the compute time for CIFAR10, STL10, and ImageNet. While CIFAR10 and STL10 are trained for 200 epochs for downstream classification ImageNet is trained for 100 epochs. (tested in 1 A100)

Dataset	w/ eigenspectrum coefficient (α)	w/ linear probe	perf. gains
CIFAR10 (s)	**3**$\pm$ **0.2**	48$\pm$ 7	$\sim 16\times$
STL10 (s)	**5**$\pm$ **0.5**	80$\pm$ 10	$\sim 16\times$
ImageNet (mins)	**4**$\pm$ **0.8**	58$\pm$ 5	$\sim 15\times$

training dataset's features, to achieve
reasonable estimates of downstream accuracy, computing α requires a single PCA step on the validation dataset's features. In Table 1, we contrast the compute times for α and evaluating a linear probe. A detailed algorithm to use α for model selection is provided in Algorithm 1 and its complexity analysis in Appendix B.6.

Algorithm 1 Model selection using α

```
# M: Number of models that can be trained in parallel
# H: Number of sequential steps of model training
# K: Number of top models to log in memory for model selection

$\alpha_{min}$ = 0
$\alpha_{max}$ = $\infty$
models_dict = {}

for iter in range(H):
    # train M models in parallel, each with different hyperparam config
    trained_models = train_SSL_models(num_parallel_models=M)

    # evaluate $\alpha$ for the trained models
    alpha_trained_models = evaluate_alpha(trained_models)
    best_trained_models = {model: alpha for model, alpha in
        alpha_trained_models if alpha  $\in$ [$\alpha_{min}$,$\alpha_{max}$]}
    models_stat = {model: run_linear_eval(model) for model, alpha in
        best_trained_models.items()}

    # Update $\alpha_{min}$ and $\alpha_{max}$
    $\alpha_{min}$ = max({$\alpha_m$ | $\alpha_m > \alpha_j$ & $\mathrm{acc}_m \geq \mathrm{acc}_j$ $\forall$ $m, j \in$ models_stat})
    $\alpha_{max}$ = min({$\alpha_m$ | $\alpha_m < \alpha_j$ & $\mathrm{acc}_m \geq \mathrm{acc}_j$ $\forall$ $m, j \in$ models_stat})

    # Trim models_stat to keep only top K models
    threshold = get_best_models(model_statistics, topk=K)
    models_stat = {model: acc for model, acc in models_stat.items() if acc >
        threshold}

# Return best model performance from the 'selected' models
best_model_acc = max({$\mathrm{acc}_m$ $\forall$ $m \in$ models_stat})
return best_model_acc
```

4 Related Work

Evaluating representations and model quality We note a substantial body of work aiming to empirically characterize the structure of emergent representations in DNN without requiring labels [29, 30]. One such index that quantifies the similarity of representations across layers (of the same or different models) is Centered Kernel Alignment (CKA) [23]. While CKA does not provide explicit guidance for downstream performance, [31] shows that the in-distribution generalization gap can be predicted using a different index based on the model's parameters. In particular, they show that the Empirical Spectral Density (ESD) of weight matrices for many DNNs obey a power-law, with the decay coefficient being predictive of in-distribution performance. In the present work, we explore similar indices that potentially correlate with out-of-distribution generalization by examining the eigenspectrum of *activations*.

Generalization in Overparameterized Models Modern neural networks often have significantly more parameters than the number of training samples, challenging the classical understanding of the bias-variance tradeoff. Overparameterization permits neural networks to overfit to noise in training data without impairing their generalization to unseen data. In recent work, Bubeck *et al.* proved that overparameterization is a necessary condition for smooth interpolation in neural networks [32].

Furthermore, this *benign overfitting* phenomenon in an overparameterized linear regression problem has been linked to the power law coefficient of the input covariance matrix [11]. Specifically, Bartlett *et al.* showed that benign overfitting is possible for an infinite-dimensional linear regression problem iff the eigenspectrum satisfies a power law (up to polylog factors). More recently, Lee *et al.* found that the tail eigenvalues of infinite-width network kernels exhibit a power law decay [33]. Following this, Tripuraneni *et al.* explored high-dimensional random feature regression settings and analytically showed a dependence between the eigenspectrum decay rate of the feature covariance matrix and generalization error [34]. While these characterizations provide a theoretical understanding of generalization error in the asymptotic or random feature settings, corresponding questions in the finite-dimensional DNN trained with gradient descent are open problems.

For deep linear networks trained using gradient descent, the eigenvalues of input covariance determine the generalization error dynamics [12]. Advani *et al.* demonstrated that small eigenvalues determine the convergence of training dynamics as well as the overfitting error at convergence. For overparameterized 2-layer neural networks, Arora *et al.* provided a fine-grained analysis of generalization bounds [35]. In contrast, we study modern DNN architectures and explore the covariance structure of their learned features on visual recognition tasks.

5 Discussion

Summary Our experiments suggest a strong correlation between the decay coefficient for sample eigenspectrum of representations, α, and both in-distribution and out-of-distribution generalization performance on tasks central to computer vision.

Representation Quality in High-Dimensional SSL We demonstrate that assessing the quality of a pretrained model with our label-free metric (α) is consistent with downstream generalization. While being computationally efficient, our procedure removes the dependence on labels, making the model selection more robust and amenable for privacy critical applications.

Necessary, but not sufficient condition Notably, a task-agnostic measure like α is a necessary but not sufficient condition for assessing good performance. To elaborate on this point, let us present a thought experiment. Suppose we have our ideal representation space that satisfies our claims of exhibiting alpha close to the Goldilocks zone. If we perform a set of permutation operations on individual datapoint representations, the structure of the representation space remains the same, but the mapping from representation to label is now destroyed. In doing this set of permutations, we have now arrived at a different set of representations that would exhibit the same (or similar) alpha but demonstrate a lower task performance when measured using a linear readout layer. A similar argument is presented as theorem C.1 in [36]. Extending the conclusions of this thought experiment to our observations, it is clear that we could have models with similar alpha values but distinctly different performances. But an alpha value that is not in the Goldilocks zone would be associated with inferior model performance. This property is the core of our claim and allows us to propose alpha as a measure for model selection in SSL pipelines.

Redundancy in ViT representations Unlike other models that we considered, representations from early layers of ViT had a rapid eigenspectrum decay with $\alpha >> 1$ (see Fig. 3 and Fig. 4). The transformer architecture has a notable difference by design, i.e. early layers possess global receptive field context via self-attention on patch embeddings. Raghu *et al.* found that early ViT layers incorporate both local and global information [30]. Based on these insights, one intuitive interpretation of our results is that the representations have a low effective rank and encode redundant information relevant across multiple scales.

Limitations While the role of α and its relationship to generalization performance is better understood in the asymptotic setting for linear regression, similar questions in finite-dimensional nonlinear models are unanswered. It is also worth noting that the empirical correlation was weaker for the earliest layers in each model. We believe that early layers learn more task invariant features such as corners and edges [23, 24], and thereby lack rich semantic information for the fine-grained downstream task. In this case, distinguishing between poorly-trained models may be inconsistent with α.

Future Directions Learning efficiently at scale from unlabelled datasets poses an exciting open problem in deep representation learning. We hope this work encourages new perspectives into model selection for SSL pipelines and informs the design of learning objectives and model architectures to learn *task-agnostic, adaptive* features. Understanding the behaviour of α (notably, a scale-invariant metric) in high-dimensional representation learning might provide insight into developing a theoretically grounded understanding of generalization in deep neural networks.

Acknowledgements

The authors would like to thank Zahraa Chorghay and Colleen Gillon for their aesthetic contribution to the manuscript and figures. This research was enabled in part by support provided by Mila and Compute Canada. This work was supported by Vanier Canada Graduate scholarship (AG); Healthy Brains, Healthy Lives (AG & BAR); NSERC, Grant No. RGPIN-2020-05105 and RGPAS-2020-00031 (BAR); and CIFAR, Canada AI Chair & Learning in Machines and Brains Fellowship (BAR).

References

[1] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*, pages 12310–12320. PMLR, 2021.

[2] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[3] Yuandong Tian, Lantao Yu, Xinlei Chen, and Surya Ganguli. Understanding self-supervised learning with dual deep networks. *arXiv preprint arXiv:2010.00578*, 2020.

[4] Daniel J Felleman and David C Van Essen. Distributed hierarchical processing in the primate cerebral cortex. *Cerebral cortex (New York, NY: 1991)*, 1(1):1–47, 1991.

[5] Claus C Hilgetag and Alexandros Goulas. 'hierarchy'in the organization of brain networks. *Philosophical Transactions of the Royal Society B*, 375(1796):20190319, 2020.

[6] Carsen Stringer, Marius Pachitariu, Nicholas Steinmetz, Matteo Carandini, and Kenneth D Harris. High-dimensional geometry of population responses in visual cortex. *Nature*, 571(7765):361–365, 2019.

[7] Nathan CL Kong, Eshed Margalit, Justin L Gardner, and Anthony M Norcia. Increasing neural network robustness improves match to macaque v1 eigenspectrum, spatial frequency preference and predictivity. *PLOS Computational Biology*, 18(1):e1009739, 2022.

[8] J Nassar, P Sokol, S Chang, and K Harris. On 1/n neural representation and robustness. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[9] Zijian Jiang, Jianwen Zhou, and Haiping Huang. Relationship between manifold smoothness and adversarial vulnerability in deep learning with local errors. *Chinese Physics B*, 30(4):048702, 2021.

[10] Zeke Xie, Qian-Yuan Tang, Yunfeng Cai, Mingming Sun, and Ping Li. On the power-law spectrum in deep learning: A bridge to protein science. *arXiv preprint arXiv:2201.13011*, 2022.

[11] Peter L Bartlett, Philip M Long, Gábor Lugosi, and Alexander Tsigler. Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences*, 117(48):30063–30070, 2020.

[12] Madhu S Advani, Andrew M Saxe, and Haim Sompolinsky. High-dimensional dynamics of generalization error in neural networks. *Neural Networks*, 132:428–446, 2020.

[13] Vatsal Shah, Anastasios Kyrillidis, and Sujay Sanghavi. Minimum norm solutions do not always generalize well for over-parameterized problems. *stat*, 1050:16, 2018.

[14] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[15] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019.

[16] SueYeon Chung, Daniel D Lee, and Haim Sompolinsky. Classification and geometry of general perceptual manifolds. *Physical Review X*, 8(3):031003, 2018.

[17] Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 413–420, 2009.

[18] A Oliva and A Torralba. Chapter 2 building the gist of a scene: the role of global image features in recognition. *Progress in Brain Research*, pages 23–36, 2006.

[19] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[21] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[22] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[23] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pages 3519–3529. PMLR, 2019.

[24] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[25] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, 33:21271–21284, 2020.

[26] Bobby He and Mete Ozay. Exploring the gap between collapsed & whitened features in self-supervised learning. In *International Conference on Machine Learning*, pages 8613–8634. PMLR, 2022.

[27] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Master's thesis*, 2009.

[28] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223. JMLR Workshop and Conference Proceedings, 2011.

[29] Thao Nguyen, Maithra Raghu, and Simon Kornblith. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. *arXiv preprint arXiv:2010.15327*, 2020.

[30] Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? *Advances in Neural Information Processing Systems*, 34, 2021.

[31] Charles H Martin, Tongsu Serena Peng, and Michael W Mahoney. Predicting trends in the quality of state-of-the-art neural networks without access to training or testing data. *Nature Communications*, 12(1):1–13, 2021.

[32] Sébastien Bubeck and Mark Sellke. A universal law of robustness via isoperimetry. *arXiv preprint arXiv:2105.12806*, 2021.

[33] Jaehoon Lee, Samuel Schoenholz, Jeffrey Pennington, Ben Adlam, Lechao Xiao, Roman Novak, and Jascha Sohl-Dickstein. Finite versus infinite neural networks: an empirical study. *Advances in Neural Information Processing Systems*, 33:15156–15172, 2020.

[34] Nilesh Tripuraneni, Ben Adlam, and Jeffrey Pennington. Covariate shift in high-dimensional random feature regression. *arXiv preprint arXiv:2111.08234*, 2021.

[35] Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning*, pages 322–332. PMLR, 2019.

[36] Nikunj Saunshi, Jordan Ash, Surbhi Goel, Dipendra Misra, Cyril Zhang, Sanjeev Arora, Sham Kakade, and Akshay Krishnamurthy. Understanding contrastive learning requires incorporating inductive biases. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 19250–19286. PMLR, 17–23 Jul 2022.

A Appendix

A.1 Proofs

In this section, we present a formal proof of Theorem 2.1. In order to do so, we will use a lemma pertaining to iterative expression of the linear regression parameters over training epochs. This lemma is inspired by the results presented in [13]

Lemma A.1. *Let $\hat{y} = x^T w$ be a finite dimensional linear regression problem where w is learned using gradient descent in order to optimize the training error,*

$$\mathcal{L} = \mathbb{E}_{x,y}[(y - \hat{y})^2] = \mathbb{E}_{x,y}[(y - x^T w)^2] \tag{4}$$

where $(x, y) \sim \mathcal{D}_{train}$, i.e. the training dataset. Then w_k, i.e. w after training for k epochs can be written as

$$w_k = X^T(XX^T)^{-1}\left[I - (I - \eta XX^T)^k\right] Y \tag{5}$$

where X, Y indicate the entire training dataset, i.e. $X \in \mathbb{R}^{N\times d}$ and $Y \in \mathbb{R}^N$

Proof. We start with the gradient of the regression loss function for a defined training set denoted by (X, Y) in a vectorized notation:

$$\begin{aligned} F(w_k) &= |Y - Xw_k|^2 \\ \implies \nabla F(w_k) &= X^T(Xw_k - Y) \end{aligned} \tag{6}$$

$$\tag{7}$$

We assume that the weights are initialized at 0, i.e. $w_0 = 0$. Using the gradient descent update:

$$\begin{aligned}
w_{k+1} &= w_k - \eta\nabla F = w_k - \eta X^T(Xw_k - Y) = (I - \eta X^T X)w_k + \eta X^T Y \\
w_1 &= (I - \eta X^T X)w_o + \eta X^T Y = \eta X^T Y \\
\text{Let } w_k &= \eta X^T u_k Y \implies u_1 = I \\
w_2 &= \eta X^T u_2 Y \\
&= (I - \eta X^T X)\eta X^T Y + \eta X^T Y = \eta X^T[(I - \eta XX^T) + I]Y \\
\implies u_2 &= (I - \eta XX^T) + I \\
u_k &= (I - \eta XX^T)u_{k-1} + I = \sum_{i=0}^{k}(I - \eta XX^T)^{i-1} \\
&= (I - (I - \eta XX^T))^{-1}(I - (I - \eta XX^T))\sum_{i=0}^{k}(I - \eta XX^T)^{i-1} \\
&= (\eta XX^T))^{-1}(I - (I - \eta XX^T))\sum_{i=0}^{k}(I - \eta XX^T)^{i-1} \\
&= \frac{1}{\eta}(XX^T)^{-1}\sum_{i=0}^{k}[(I - \eta XX^T)^{i-1} - (I - \eta XX^T)^i] \\
&= \frac{1}{\eta}(XX^T)^{-1}[I - (I - \eta XX^T)^k] \\
\implies w_k &= \eta X^T u_k Y = X^T(XX^T)^{-1}[I - (I - \eta XX^T)^k]Y
\end{aligned} \tag{8}$$

□

This proves Lemma A.1. We will now use this lemma to prove Theorem 2.1. From this result, we can also write:

$$\Delta w_k = \eta X^T(I - \eta XX^T)^k Y \tag{9}$$

We restate the theorem from the main text. Note that the notations are simplified here from the theorem statement to improve readability.

Theorem A.2. *Let $\hat{Y} = X^T w$ be a finite dimensional linear regression problem where w is learned using gradient descent. If we assume power law distribution in eigenspectrum of X, i.e. $\lambda_n = \frac{c}{n^\alpha} \forall n \in \{1, 2...N\}$, then the time required by gradient descent to minimize the training error, $T_{convergence} = \mathcal{O}(N^\alpha)$*

Proof. Using result from Lemma A.1, it is clear that the gradient converges to 0 if $\lambda_1 < \frac{1}{\eta}$ where λ_1 is the leading eigenvalue of XX^T.

Thus, $\eta < \frac{1}{\lambda_1}$, i.e. small learning rate setting. So, we set $\eta = \frac{\hat{\eta}}{\lambda_1}$ where $\hat{\eta} < 1$. Plugging this in Eq. (9)

$$\Delta w_k = \frac{\hat{\eta}}{\lambda_1} X^T (I - \frac{\hat{\eta}}{\lambda_1} XX^T)^k Y \tag{10}$$

Let $X = U \wedge^{\frac{1}{2}} V^T$ denote the singular value decomposition (SVD), which implies $XX^T = U \wedge U^T$. Using the SVD, we get $(I - \frac{\hat{\eta}}{\lambda_1} XX^T)^k = (I - \frac{\hat{\eta}}{\lambda_1} U \wedge U^T)^k$. It is worth noting that eigenvalues and eigenvectors of $(I - \frac{\hat{\eta}}{\lambda_1} U \wedge U^T)$ are related to that of XX^T as shown below:

$$\begin{aligned} (I - \frac{\hat{\eta}}{\lambda_1} U \wedge U^T) u_i &= u_i - \frac{\hat{\eta}}{\lambda_1} (U \wedge U^T u_i) \\ &= u_i - \frac{\hat{\eta}}{\lambda_1} \lambda_i u_i \qquad \text{[Using } U^T U = I\text{]} \\ \implies (I - \frac{\hat{\eta}}{\lambda_1} U \wedge U^T) u_i &= (1 - \frac{\hat{\eta}}{\lambda_1} \lambda_i) u_i \end{aligned} \tag{11}$$

Using Eq. (11), we can write $(I - \frac{\hat{\eta}}{\lambda_1} U \wedge U^T)$ in the eigendecomposition form as $U \tilde{\wedge} U^T$ where $\tilde{\lambda_i} = 1 - \hat{\eta} \frac{\lambda_i}{\lambda_1}$. Thus, $\left(I - \frac{\hat{\eta}}{\lambda_1} U \wedge U^T\right)^k = U \tilde{\wedge}^k U^T$. Plugging this in Eq. (10), we get:

$$\Delta w_k = \frac{\hat{\eta}}{\lambda_1} V \wedge^{\frac{1}{2}} U^T U \tilde{\wedge}^k U^T Y = \frac{\hat{\eta}}{\lambda_1} V \wedge^{\frac{1}{2}} \tilde{\wedge}^k S \tag{12}$$

where $S = U^T Y \in \mathbb{R}^N$. For the i^{th} element, we get $\Delta w_k^{(i)} = \frac{\hat{\eta}}{\lambda_1} \sum_j v_{i,j} \sqrt{\lambda_j} \tilde{\lambda_j}^k S_j = \frac{\hat{\eta}}{\lambda_1} \sum_j v_{i,j} \sqrt{\lambda_j} (1 - \hat{\eta} \frac{\lambda_j}{\lambda_1})^k S_j$. Since all other factors remain constant across training, i.e. do not change with k, the convergence of gradient descent depends on the factors $\left(1 - \hat{\eta} \frac{\lambda_j}{\lambda_1}\right)^k$. Note that we define gradient descent to converge when $\Delta w_k^{(i)} \approx 0 \,\forall\, i$. Therefore, the limiting factor that determines rate of convergence is $(1 - \hat{\eta} \frac{\lambda_j}{\lambda_1})^k$, which in turn is limited by the smallest eigenvalue factor: $\frac{\lambda_N}{\lambda_1}$.
Assuming $\frac{\lambda_N}{\lambda_1} \ll 1 \implies \hat{\eta} \frac{\lambda_N}{\lambda_1} \ll 1$ as $\hat{\eta} < 1 \implies (1 - \hat{\eta} \frac{\lambda_N}{\lambda_1})^k \approx 1 - k \hat{\eta} \frac{\lambda_N}{\lambda_1}$.
Hence the convergence time, $k^* = \mathcal{O}(\hat{\eta} \frac{\lambda_1}{\lambda_N}) = \mathcal{O}(\frac{\lambda_1}{\lambda_N})$
If λ_i follows power law, i.e, $\lambda_i = c i^{-\alpha}$ and $\frac{\lambda_N}{\lambda_1} = N^{-\alpha}$ then $k^* = \mathcal{O}(N^\alpha)$ i.e. k^* grows exponentially with α. □

Theorem A.3. *Let $\hat{y} = \mathbf{f}_\theta(x)^T \psi$ be a linear regression problem as before. Let us further assume that $\mathbf{f}_\theta(x) \,\forall x \sim \mathcal{D}_{train}$ is a representative subset of the inputs from true data distribution: $(x, y) \sim \mathcal{D}$. Assuming a power-law distribution in the eigenspectrum of representations at $\mathbf{f}_\theta$, i.e. $\lambda_n = \frac{c}{n^\alpha} \quad \forall n \geq n^*$, where $n^* \in \{1, 2...N\}$, the generalization error after T weight update steps, $\mathcal{G}(T)$ is:*

$$\begin{aligned} \mathcal{G}(T) := \mathbb{E}_{x,y \sim \mathcal{D}} [(y - \mathbf{f}_\theta(x)^T \psi)^2] \quad &\leq \mathcal{O}\left(r_{\hat{d}}(\Sigma(\mathbf{f}_\theta))\right) \\ \textit{where} \qquad r_{\hat{d}}(\Sigma(\mathbf{f}_\theta)) &= \frac{\sum_{i=\hat{d}}^m \lambda_i}{\sum_{i=1}^m \lambda_i} = \frac{\sum_{i=\hat{d}}^m i^{-\alpha}}{\sum_{i=1}^m i^{-\alpha}} \end{aligned} \tag{13}$$

Here, $m = min(N, d)$ is the rank of $\Sigma_N(\mathbf{f}_\theta)$.

Proof. We start with the gradient of the regression loss function for a defined training set denoted by (X_s, Y_s) in a vectorized notation where $X \in \mathbb{R}^{N \times d}$ and $Y \in \mathbb{R}^N$. For the brevity of notation let us denote $\mathbf{f}_\theta(x)$ by x. The loss/error for the regression problem can be given by:

$$E = \frac{1}{2}(Y_s - X_s\phi)^T(Y_s - X_s\phi)$$

This gives gradient of weight vector as:

$$\implies \Delta\phi = -\eta\nabla_\phi E = \eta X^T Y - \eta X^T X\phi \tag{14}$$

where η is the learning rate. Now, let the true labels be generated by $Y_s = X_s\phi^* + \epsilon$ where ϵ is a random variable denoting noise. Also, let $X_s = U_s \wedge^{\frac{1}{2}} V^T$ which gives $X_s^T X_s = V \wedge V^T$ and $X_s^T Y_s = V \wedge^{\frac{1}{2}} U_s^T Y = V \wedge^{\frac{1}{2}} \tilde{S}$ where $\tilde{S} = U_s^T Y = U_s^T(U_s \wedge^{\frac{1}{2}} V^T\phi^* + \epsilon) = \wedge^{\frac{1}{2}} Z^* + \tilde{\epsilon}$, $Z^* = V^T\phi^*$ and $\tilde{\epsilon} = U_s^T\epsilon$. Let $\phi = VZ$, i.e. expressing ϕ using the eigenbasis V of X^TX and Z as the new parameters. This gives:

$$\Delta Z = V^T\Delta\phi = \eta V^T(V \wedge^{\frac{1}{2}} \tilde{S} + V \wedge V^T\phi) = \eta(\wedge^{\frac{1}{2}}\tilde{S} - \wedge Z) \tag{15}$$

$$\implies \Delta Z = \eta \wedge (Z^* - Z) + \eta \wedge^{\frac{1}{2}} \tilde{\epsilon} \tag{16}$$

This implies no mixing between the eigenmodes. Solving Eq. (16) we get:

$$Z_i^* - Z_i(t) = (Z_i^* - Z_i(0))e^{-\eta\lambda_i t} - \frac{\tilde{\epsilon_i}}{\sqrt{\lambda_i}}(1 - e^{-\eta\lambda_i t}) \tag{17}$$

Now, let $\phi^* - \phi = \delta$ and $Z^* - Z = \rho = V^T\delta$, which gives $\rho_i(t) = \rho_i(0)e^{-\eta\lambda_i t} - \frac{\tilde{\epsilon_i}}{\sqrt{\lambda_i}}(1 - e^{-\eta\lambda_i t})$. Then the generalization error is:

$$\mathcal{G} =< (\phi^* - \phi)^T X^T X(\phi^* - \phi) >=< \rho^T \wedge \rho >=< \sum_{i=1}^{m} \lambda_i\rho_i^2 >= \sum_{i=1}^{m} \lambda_i < \rho_i^2 > \tag{18}$$

where X is the datapoints from the total dataset for which we care about the generalization error and m is the rank of the covariance matrix, X^TX. Without loss of generality, we can assume $m = min(N, d)$. Note that $\langle . \rangle$ denotes the expectation over different network weight initialization and label noise. We define X_s to be a reliable sample of dataset X if both X^TX and $X_s^TX_s$ have the same eigenvectors, i.e. V. Therefore, we could write $X^TX = X_s^TX_s$ in Eq. 18. Note that the reliability of the sample indicates the inherent structure of the data, i.e. the variance along the principal components in the data space have been captured by X_s. Subsequently, we get the expression for generalization error for a fixed training budget upto time/epoch (T):

$$\mathcal{G}(T) = \sum_{i=1}^{m} \lambda_i\sigma_\rho^2 e^{-2\eta\lambda_i T} + \sigma_\epsilon^2(1 - e^{-\eta\lambda_i T})^2 \tag{19}$$

where σ_ρ indicates the noise due to weight initialization and σ_ϵ indicates the noise in labels. It is clear from Eq. (19) that directions corresponding to larger eigenvalues converge faster and contribute to the generalization error through second term. Let us, therefore, assume that the top $\tilde{d}$ eigenmodes have converged. Therefore, $e^{-\eta\lambda_i T} \leq c_0 \ \forall i \leq \tilde{d} < m$, where c_0 is some small number. Hence,

$$\begin{aligned}
& e^{-\eta\lambda_i T} \leq c_0 \quad \forall i \leq \tilde{d} \quad \text{and} \quad e^{-\eta\lambda_i T} > c_0 \quad \forall i > \tilde{d} \\
\implies & e^{-2\eta\lambda_i T} \leq c_0^2 \quad \forall i \leq \tilde{d} \quad \text{and} \quad (1 - e^{-\eta\lambda_i T})^2 \leq (1 - c_0)^2 \quad \forall i > \tilde{d} \\
\text{Also} \quad & (1 - e^{-\eta\lambda_i T})^2 \leq 1 \quad \forall i \leq \tilde{d} \quad \text{and} \quad e^{-2\eta\lambda_i T} \leq 1 \quad \forall i > \tilde{d}
\end{aligned} \tag{20}$$

Plugging inequalities from Eq. (20) in Eq. (19), we can upper bound the generalization error:

$$\begin{aligned}
\sum_{i=1}^{\tilde{d}} \lambda_i\sigma_\rho^2 e^{-2\eta\lambda_i T} + \sigma_\epsilon^2(1 - e^{-\eta\lambda_i T})^2 &\leq \sigma_\rho^2 c_0^2 \sum_{i=1}^{\tilde{d}} \lambda_i + \sigma_\epsilon^2 \\
\sum_{i=\tilde{d}+1}^{m} \lambda_i\sigma_\rho^2 e^{-2\eta\lambda_i T} + \sigma_\epsilon^2(1 - e^{-\eta\lambda_i T})^2 &\leq \sigma_\rho^2 \sum_{i=\tilde{d}+1}^{m} \lambda_i + \sigma_\epsilon^2(1 - c_0)^2
\end{aligned} \tag{21}$$

Therefore, the generalization error can be upper bounded as follows:

$$\mathcal{G}(T) \leq \sigma_\rho^2 c_0^2 \sum_{i=1}^{\tilde{d}} \lambda_i + \sigma_\epsilon^2 + \sigma_\rho^2 \sum_{i=\tilde{d}+1}^{m} \lambda_i + \sigma_\epsilon^2(1 - c_0)^2 \tag{22}$$

Now, we can choose c_0 to be such that $c_0^2 \sum_{i=1}^{\tilde{d}} \lambda_i = \kappa_0$ where κ_0 is some constant independent of T. Furthermore, we impose the power-law assumption in the eigenvalues, i.e. $\lambda_i = ci^{-\alpha}$. With these assumptions, we can simplify Eq. (22) as follows:

$$\begin{aligned} &\mathcal{G}(T) \leq \sigma_\rho^2 \kappa_0 + \sigma_\epsilon^2 \left(1 + (1 - c_0)^2\right) + \sigma_\rho^2 c \sum_{i=\tilde{d}+1}^{m} i^{-\alpha} \\ \implies &\mathcal{G}(T) \leq \kappa + \sigma_\rho^2 c[var(X)] \frac{\sum_{i=\tilde{d}+1}^{m} i^{-\alpha}}{var(X)} = \kappa + \sigma_\rho^2 [var(X)] \frac{\sum_{i=\tilde{d}+1}^{m} i^{-\alpha}}{\sum_{i=1}^{m} i^{-\alpha}} \end{aligned} \tag{23}$$

where κ is another constant (used for brevity), $\kappa = \sigma_\rho^2 \kappa_0 + \sigma_\epsilon^2 \left(1 + (1 - c_0)^2\right)$. Also, we used the relation: $var(X) = \sum_i \lambda_i = \sum_i ci^{-\alpha}$. Therefore, we can set $\hat{d} = \tilde{d} - 1$ and subsequently prove the statement of the theorem:

$$\mathcal{G}(T) \leq \kappa + \sigma_\rho^2 [var(X)] \frac{\sum_{i=\hat{d}}^{m} i^{-\alpha}}{\sum_{i=1}^{m} i^{-\alpha}} = \mathcal{O}\left(r_{\hat{d}}(\Sigma(\mathbf{f}_\theta)\right) \tag{24}$$

□

B Additional Experimental results

In this section we provide additional results, evidence to support our hypothesis of α being a useful measure of downstream generalization performance.

B.1 Generalization & Eigenspectrum Decay on STL10 and ImageNet

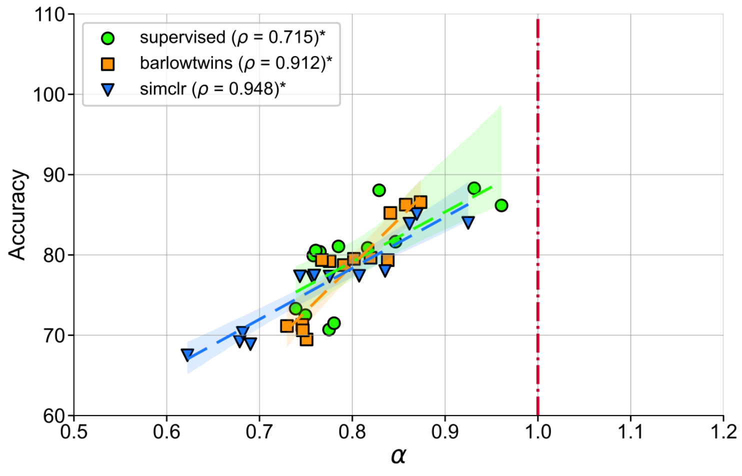


Figure 6: To measure downstream performance, one alternative is using non-linear readouts. With fixed feature-set, the performance of non-linear readout on STL-10 demonstrates correlation with α **(across different pretraining loss functions)** The trends are similar to the performance of a linear readout, as shown in Fig. 3. $^*p < 0.05$.

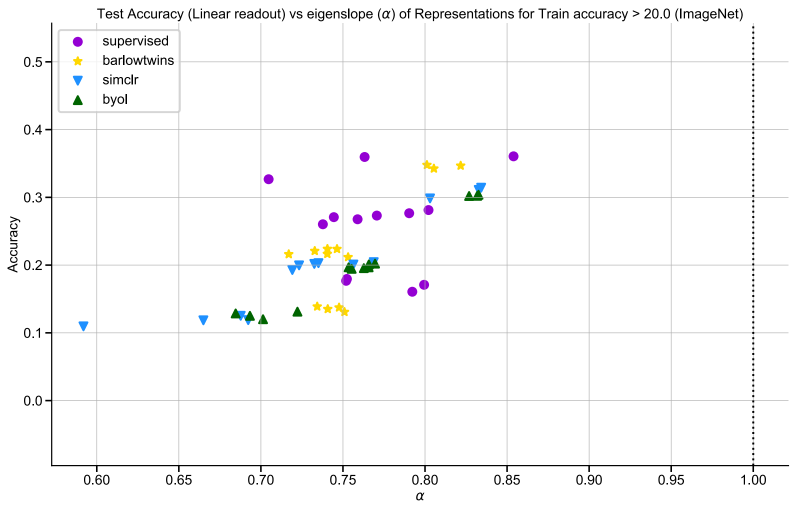


Figure 7: Unless specified explicitly, our models are pretrained on the ImageNet dataset. Here, we evaluate **in-distribution generalization** for ResNet-50 models pretrained with multiple learning objectives. We see α is correlated with the generalization on ImageNet. Note that the input images are downsampled for this experiment to get lower dimension features for intermediate layers, which facilitates the Eigendecomposition of the large covariance matrix for ImageNet. However, this is not a problem when we work with the features extracted from the final layers, which suggests that the model selection algorithm proposed in this paper also works for high-resolution images.

B.2 Layerwise evaluation on STL 10 and MIT 67

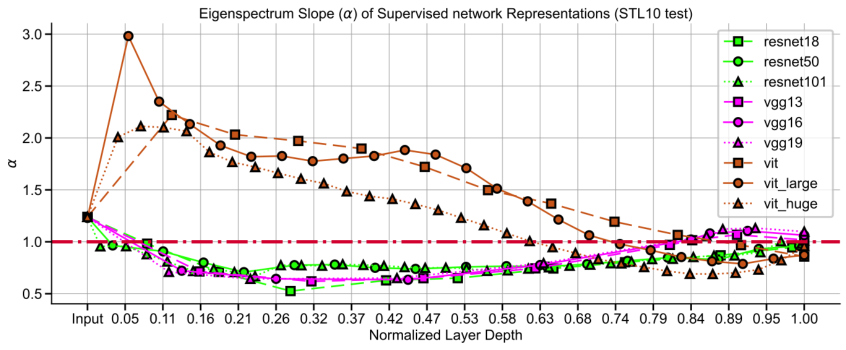


Figure 8: α for intermediate layer representations from different backbone architectures demonstrates the contrasting representations learned by CNNs and ViT.

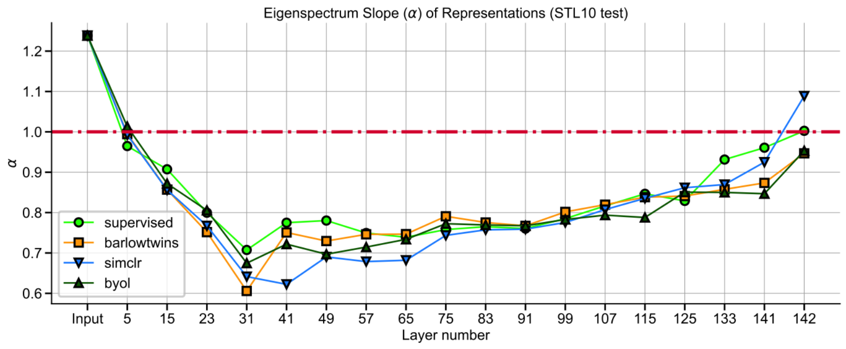


Figure 9: α for intermediate layer representations from networks trained using different loss functions show similar trends. Representations from deeper layers exhibit α closer to 1 as compared to middle layer representations.

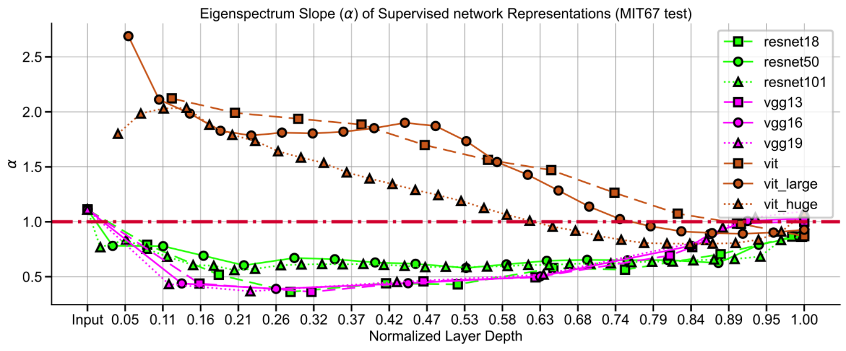


Figure 10: α for intermediate layer representations from different backbone architectures in MIT67. Representations learned by ViT is qualitatively different from those is CNNs both in object and scene recognition datasets.

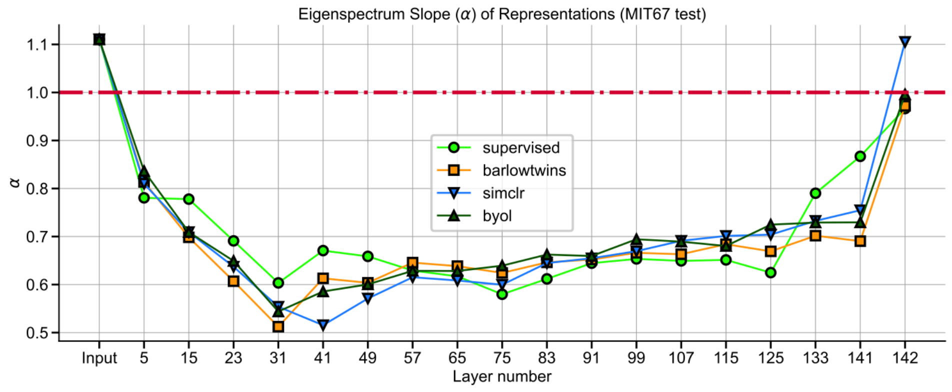


Figure 11: α for intermediate layer representations from networks trained using different loss functions show similar trends in MIT67. Empirical evaluations suggest that representations from deeper layers exhibit α closer to 1 as compared to intermediate layer representations.

B.3 Visualizing design landscape for BarlowTwins

As a diagnostic metric for measuring representation quality, we chart the learning landscape for self-supervised learning algorithms (Barlow Twins here). To this effect, we empirically investigate the role of different critical hyperparameters in learnability and generalization. In particular, we ablate across projection-dimensionality, redundancy coefficient, weight-decay and learning rate.

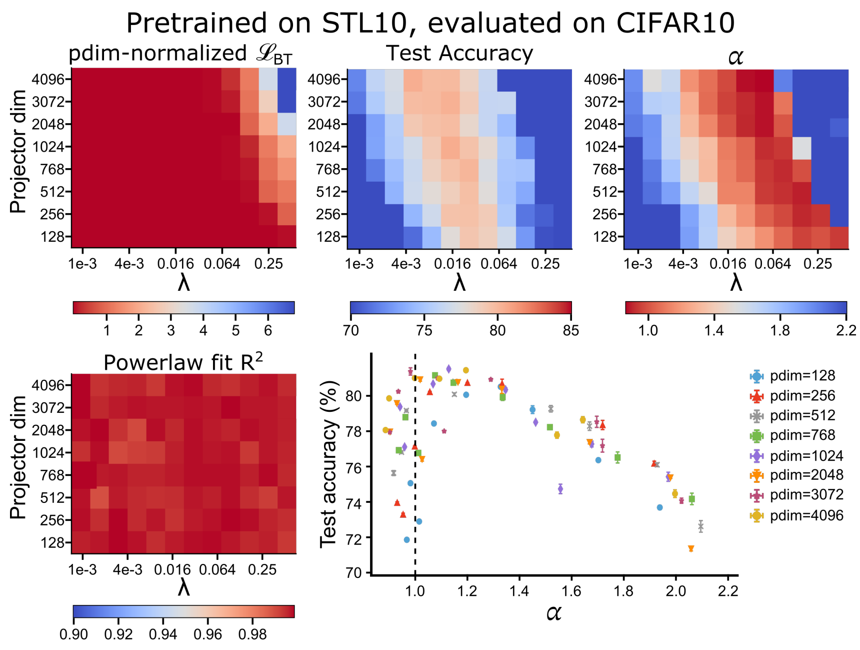


Figure 12: For ResNet-50 model architecture, we measure out-of-distribution generalization for BarlowTwins pretrained on STL10 and evaluated with linear-probes on CIFAR10.

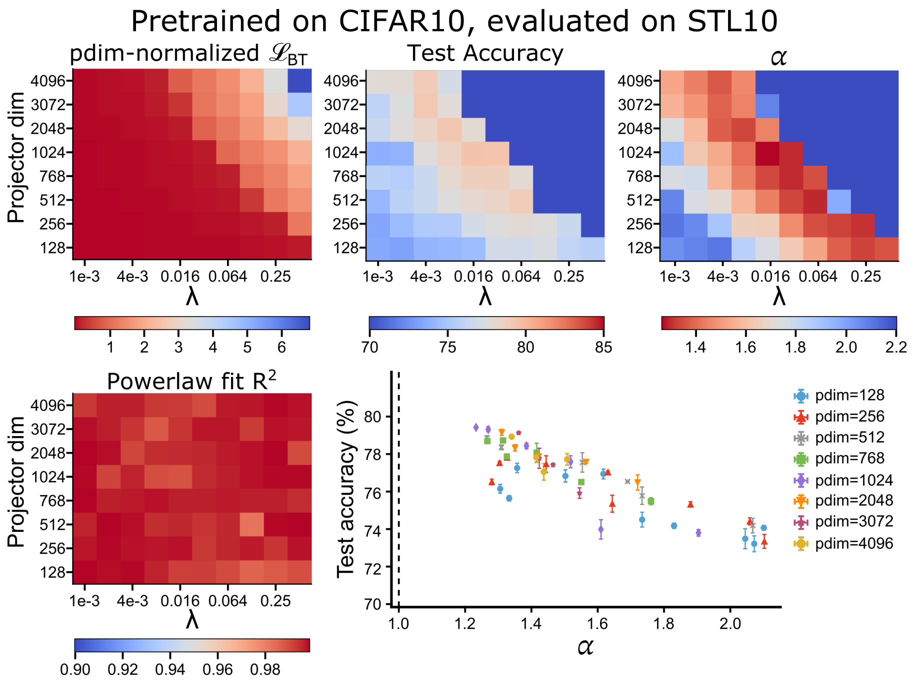


Figure 13: For ResNet-50 model architecture, we measure out-of-distribution generalization for BarlowTwins pretrained on CIFAR10 and evaluated with linear probes on STL10.

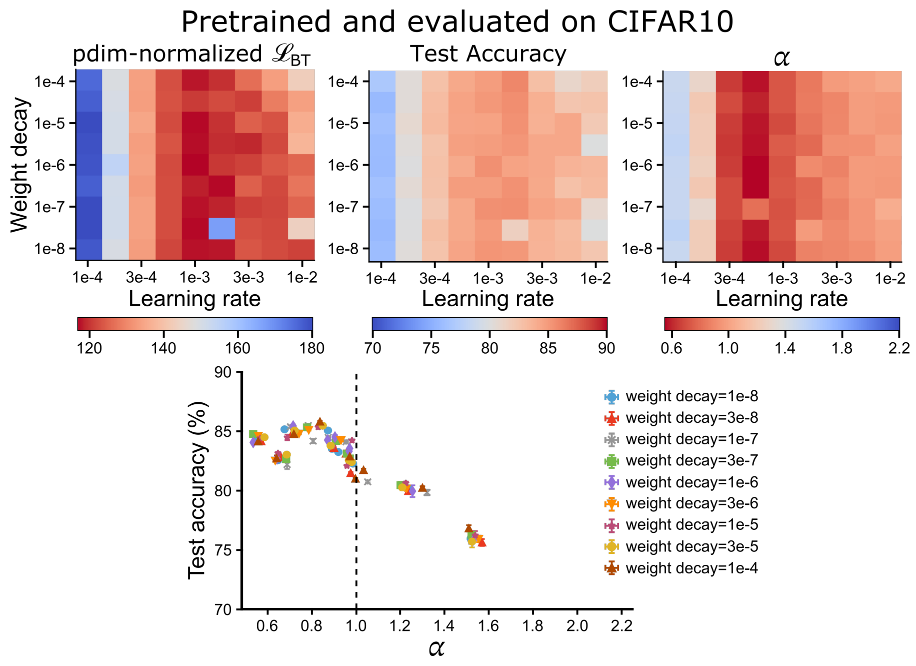


Figure 14: Measuring in-distribution generalization for BarlowTwins trained on CIFAR10 and evaluated on CIFAR10 when sweeping over different optimization hyperaprameters, namely learning rate and weight decay.

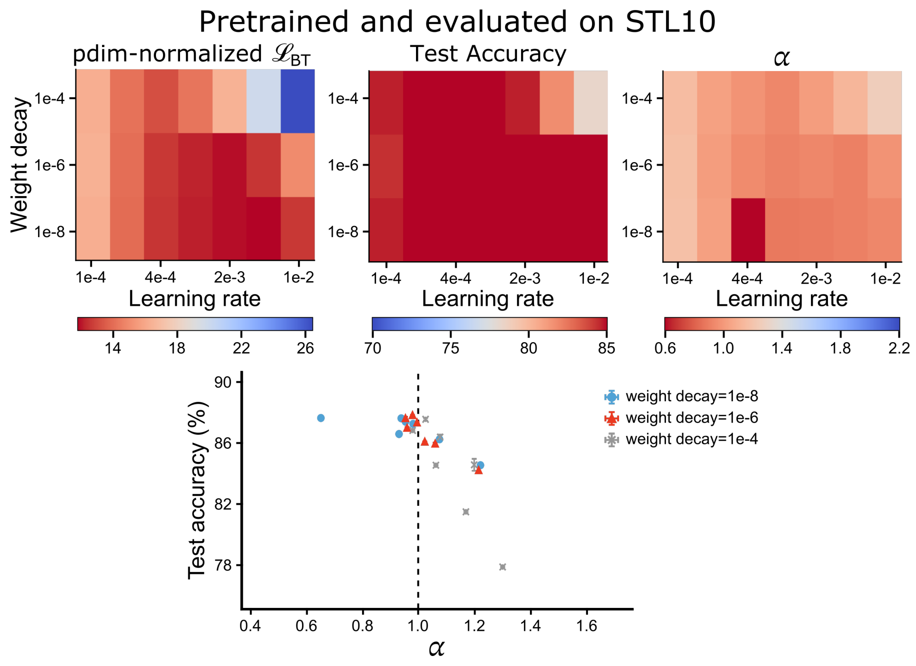


Figure 15: Measuring in-distribution generalization for BarlowTwins trained on STL10 and evaluated on STL10 when sweeping over different optimization hyperaprameters, namely learning rate and weight decay.

B.4 Visualizing design landscape for SimCLR

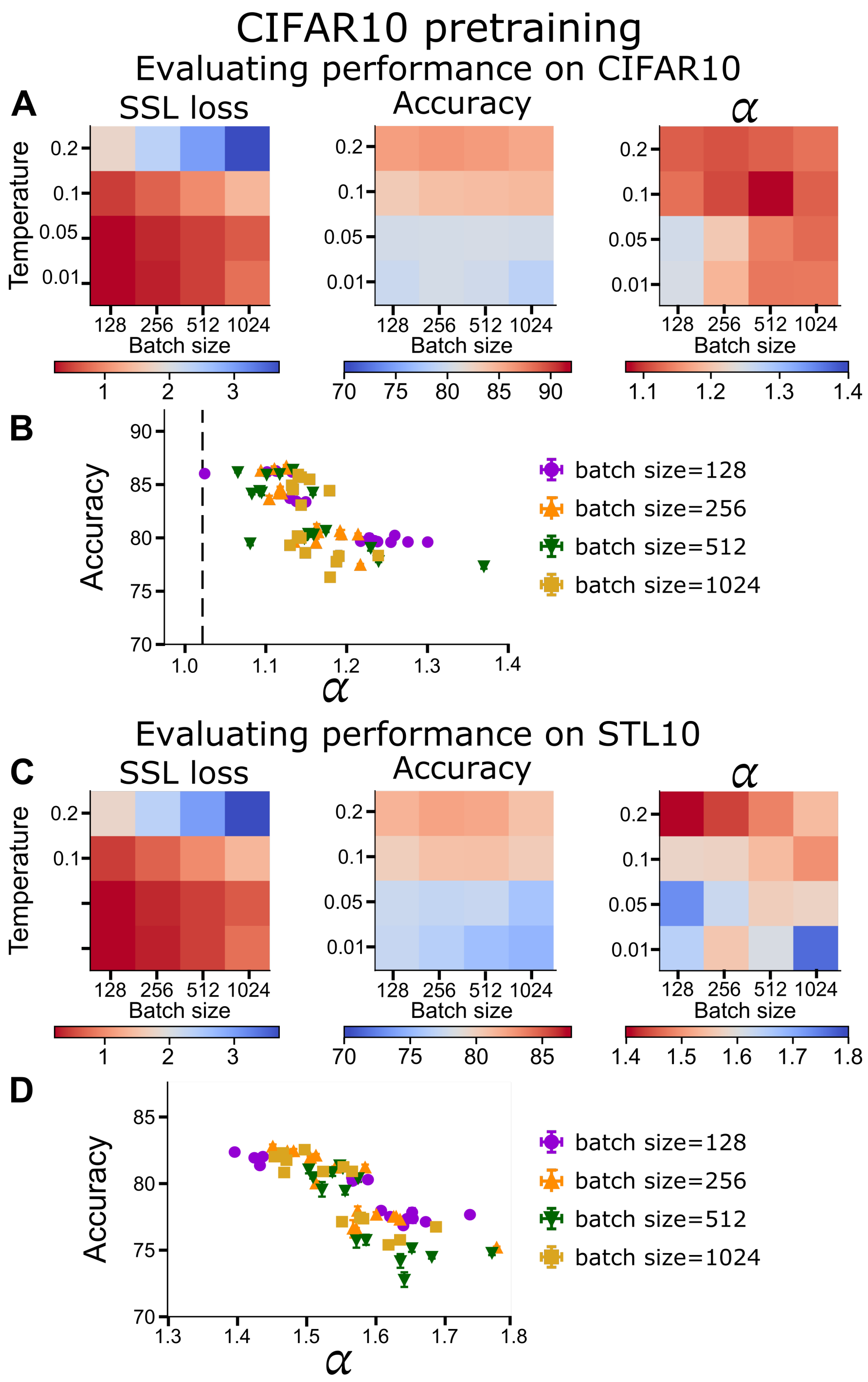


Figure 16: Hyperparameter sweep for SimCLR pretraining on CIFAR10: the correspondence between α and downstream accuracy holds for contrastive SSL methods as well. (A) SimCLR loss, (in-distribution) classification accuracy on CIFAR10 and corresponding α of learned representations when sweeping over different values of temperature and batch size, two key hyperparameters for SimCLR training. (B) Accuracy vs α plot for all models trained with different values of temperature, batch size and projector dimensionality. (C-D) Same as A and B for downstream (out-of-distribution) classification accuracy evaluated on STL10.

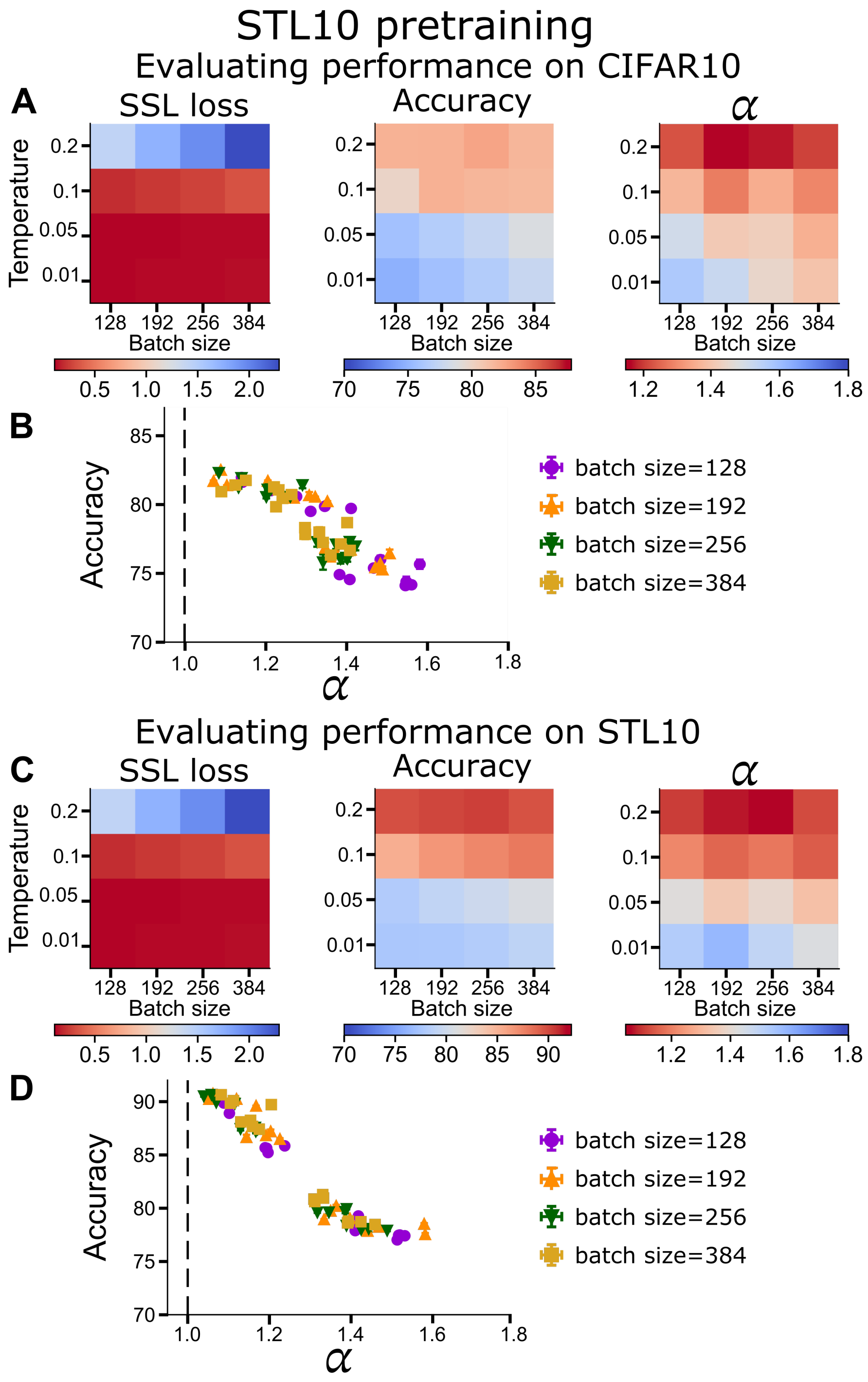


Figure 17: Hyperparameter sweep for SimCLR pretraining on STL10: the correspondence between α and downstream accuracy holds for contrastive SSL methods as well. (A) SimCLR loss, (out-of-distribution) classification accuracy on CIFAR10 and corresponding α of learned representations when sweeping over different values of temperature and batch size. (B) Accuracy vs α plot for all models trained with different values of temperature, batch size and projector dimensionality. (C-D) Same as A and B for downstream (in-distribution) classification accuracy evaluated on STL10.

B.5 Model Selection on Compute Budget

Algorithm 2 Model selection using α

```
# M: Number of models that can be trained in parallel
# H: Number of sequential steps of model training
# K: Number of top models to log in memory for model selection

$\alpha_{min}$ = 0
$\alpha_{max}$ = $\infty$
models_dict = {}

for s in range(H):
    # train M models in parallel, each with different hyperparam config
    trained_models = train_SSL_models(num_parallel_models=M)

    # evaluate $\alpha$ for the trained models
    alpha_trained_models = evaluate_alpha(trained_models)

    # choose models based on alpha to run linear evaluation
    for model in trained_models:
        if alpha_trained_models[model] $\in$ [$\alpha_{min}$,$\alpha_{max}$]:
            model_alpha = alpha_trained_models[model]
            model_performance = run_linear_eval(model)
            models_dict.insert({model:(model_alpha,model_performance)})
        else:
            pass

    # Update $\alpha_{min}$ and $\alpha_{max}$
    $\alpha_{min}$ = max({$\alpha_m$ | $\alpha_m > \alpha_j$ & $performance_m \geq performance_j \ \forall \ m,j \in$ models_dict})
    $\alpha_{max}$ = min({$\alpha_m$ | $\alpha_m < \alpha_j$ & $performance_m \geq performance_j \ \forall \ m,j \in$ models_dict})

    # Trim models_dict to keep only top K models
    performance_thresh = get_top_K_model_performance(models_dict)
    for model in models_dict:
        if model.performance < performance_thresh:
            models_dict.pop(model)

# Return best model performance from the 'selected' models
best_model_performance = max({$performance_m \ \forall \ m \in$ models_dict})
return best_model_performance
```

B.6 Average complexity analysis of Algorithm 1

Let us assume that the compute budget is characterized by M, i.e. the number of SSL models that can be trained parallely on the compute infrastructure. Let us assume H such parallel steps are run sequentially in order to train $M \times H$ different model configurations (e.g. sweeps over different hyperparameter configurations). Let us denote the probability of performing linear evaluation on a model trained in the r^{th} sequential step as ϵ_r. Therefore, expected number of linear evaluations in the r^{th} step will be $M\epsilon_r$. Now, clearly $\epsilon_r \leq 1$, and hence:

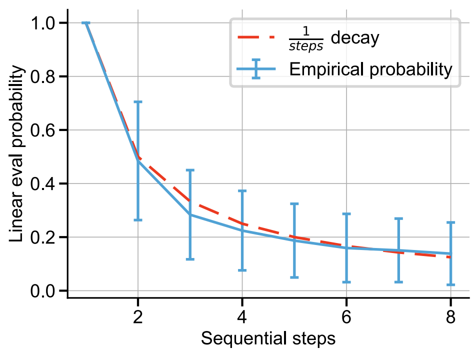
$$\mathbb{E}[\text{linear\_evals}] = \sum_{r=1}^{H} M\epsilon_r \leq \sum_{r=1}^{H} M = MH \tag{25}$$

This relationship provides the worst case complexity of Algorithm 2. We can further assume that the probability of linear evaluation decays with each sequential step. Therefore, the average number of linear evaluations would be less than MH. Assuming $\epsilon_r = \mathcal{O}\left(\frac{1}{r}\right)$ [†], we can write:

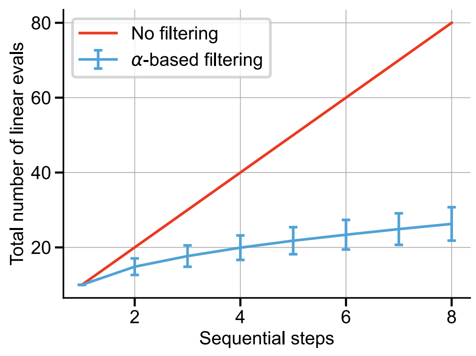
$$\mathbb{E}[\text{linear\_evals}] = \sum_{r=1}^{H} M\epsilon_r = \sum_{r=1}^{H} M\mathcal{O}\left(\frac{1}{r}\right) = \mathcal{O}\left(Mlog(H)\right) \tag{26}$$

Taken together, the above equation shows that the average case complexity of model evaluation using α is less than the standard complexity of model evaluation, i.e. MH linear evaluations.

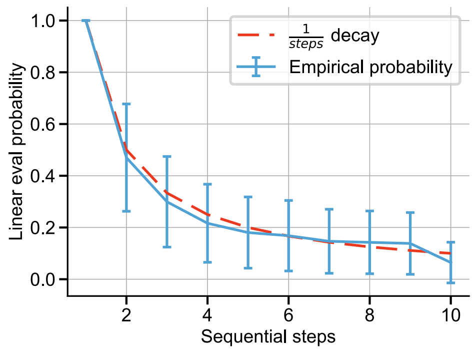
[†] Verified in Fig. 18



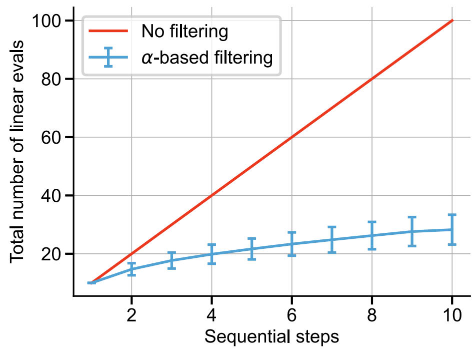
(a) CIFAR10



(b) CIFAR10



(c) STL10



(d) STL10

Figure 18: Our analysis suggests a range $[\alpha_{min}, \alpha_{max}]$ for α which indicates the best model in a set of pretrained models. If this range is known, using α is significantly more compute efficient that standard exhaustive search which requires multiple linear evaluations, as noted in 1. Furthermore, an effective search algorithm using offline-metric α allows us to reduce the number of linear evaluations from $\mathcal{O}(MH)$ to $\mathcal{O}(M \log H)$, assuming that the likelihood of linear evaluation for models decays as $\frac{1}{steps}$. This assumption is verified in (a,c)

C On necessary and sufficiency conditions for predicting generalization

Understanding the correlation between α and *generalization* requires probing the natural question of whether α in a given range is sufficient for good downstream performance. To further elaborate on this question, consider the following: Let us suppose that we have our ideal representation space that exhibit alpha within the Goldilocks zone. If we perform a set of permutation operations on individual datapoint representations, the structure of the representation space remains the same but the mapping from representation to label is now destroyed. In doing this set of permutation operations, we have now arrived at a different set of representations that would exhibit the same (or similar) alpha but demonstrate a lower task performance when measured using a linear readout layer. Extending the conclusions of this thought experiment to our observations, it is clear that we could have models with similar alpha values but distinctly different performance. But an alpha value that is not in the Goldilocks zone would be associated with inferior model performance. This property is the core of our claim and allows us to propose alpha as a measure for model selection in SSL pipelines.

D High-Resolution plots

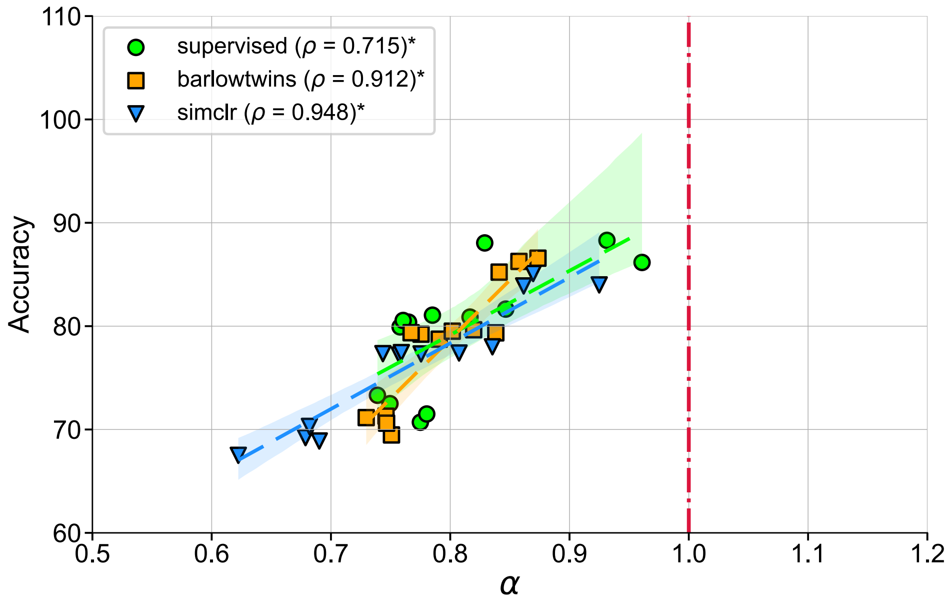


Figure 19: The x-axis represents estimated α, y-axis is evaluating linear-probes on STL-10. Different models correspond to pretraining on ImageNet with different learning objectives (we fix the architecture to ResNet-50).

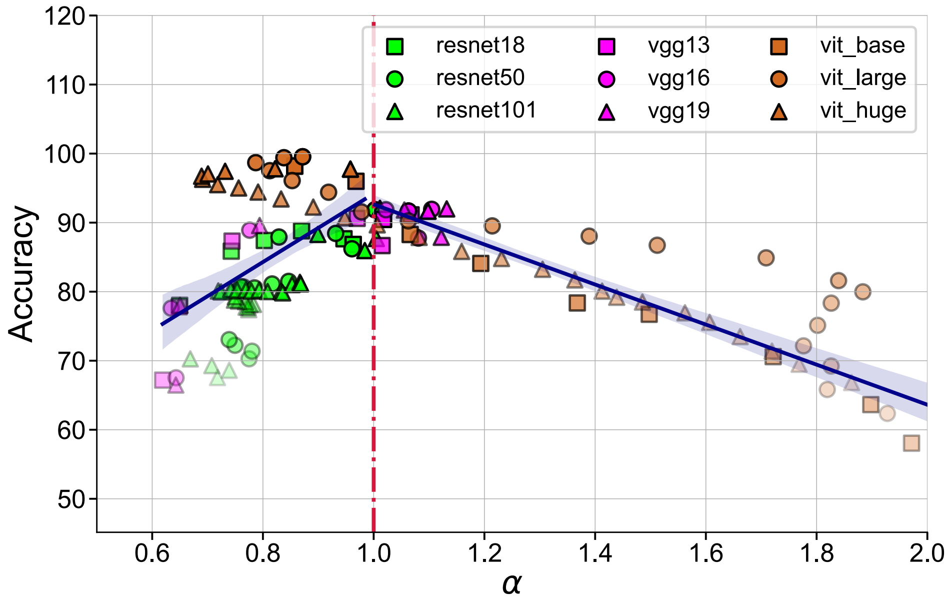


Figure 20: The x-axis represents estimated α, y-axis is evaluating linear-probes across different layers across multiple architectures on STL10. All models are pretrained with supervised learning on ImageNet (we fix learning objective to be cross-entropy).

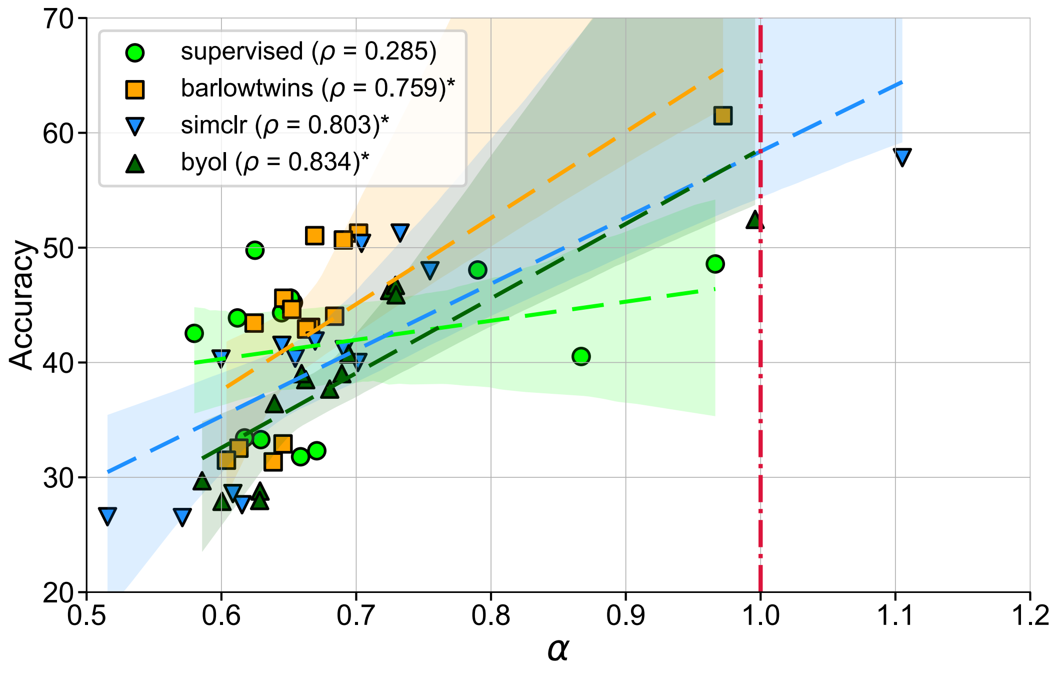


Figure 21: The x-axis represents estimated α, y-axis is evaluating linear-probes on MIT-67 (scene recognition). Different models correspond to pretraining on ImageNet with different learning objectives (we fix the architecture to ResNet-50).

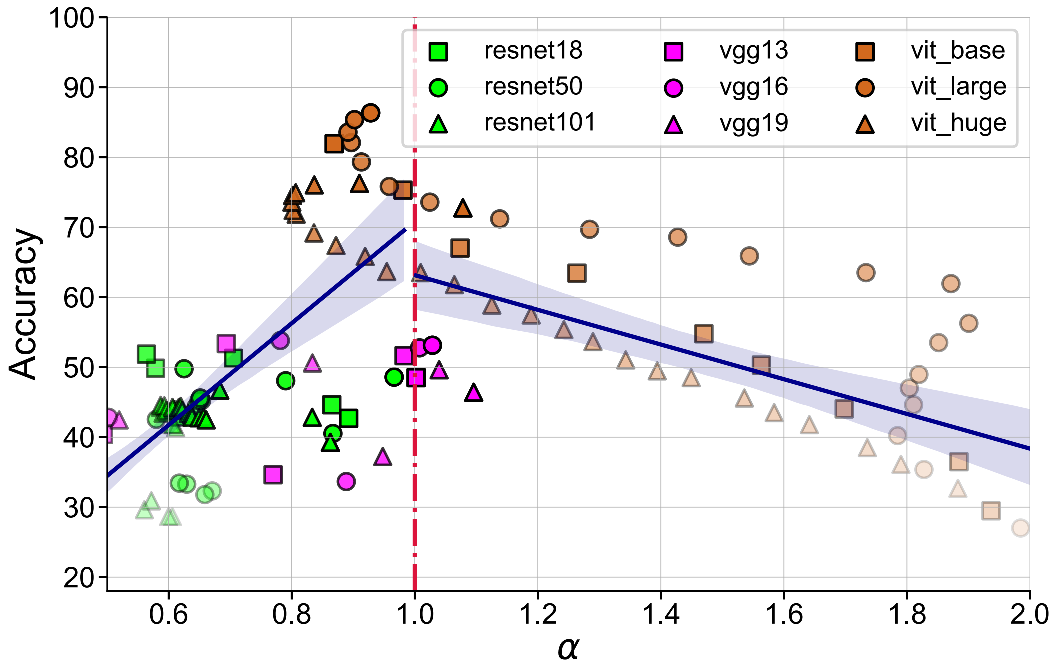


Figure 22: The x-axis represents estimated α, y-axis is evaluating linear-probes across different layers across multiple architectures on MIT-67 (scene recognition). All models are pretrained with supervised learning on ImageNet (we fix learning objective to be cross-entropy).

Part III

Dynamics of feature learning

5 Preface to Part III

In Part III, we move on to the learning dynamics pillar of the NeuroAI framework. Previously, I presented evidence for shared characteristics between learned representations in the early visual cortex and in ANNs trained using SSL. Both systems have high-dimensional scale-free representations that can be characterized with a powerlaw in their eigenspectrum. A natural question that arises, specifically for SSL-trained ANNs, is *What are these representations and how do they emerge over the course of learning?* In Part III, we study the learning desiderata and dynamics of feature learning in these networks. Specifically, we study the core quantity that every SSL objective aims to learn and the influence of gradient descent in shaping the learned features.

In line with the NeuroAI framework, Part III also investigates the normative role of two mechanisms that are thought to aid biological learning in feature learning by studying them in the context of ANNs trained using SSL: (1) orthogonality constraints on neural activities (thought to be implemented by inhibitory neurons), and (2) multiple views of inputs while learning the invariance relationships. It is worth noting that these two mechanisms have been explored in previous literature as heuristics for improving feature learning. Consequently, Part III also aims to leverage this normative understanding to improve the compute and sample-efficiency of SSL pipelines, thereby offering practical benefits to the AI community. Overall, the following chapter extends the utility of representation geometry, not only as a tool for understanding computational motifs of intelligent systems, but also for studying the learning dynamics that yield the observed representations.

* * *

This work is accepted to be published at NeurIPS 2024 and can be cited as Ghosh, A.[*], Agrawal, K.K.[*], Sodhani, S., Oberman, A. and Richards, B.A. Harnessing small projectors and multiple views for efficient vision pretraining. NeurIPS 2024.

6

Harnessing small projectors and multiple views for efficient vision pretraining

Harnessing small projectors and multiple views for efficient vision pretraining

Arna Ghosh [*1] **Kumar Krishna Agrawal** [*2] **Shagun Sodhani** [3]

Adam M. Oberman[† 4] **Blake A. Richards** [† 1 5 6]

Abstract

Recent progress in self-supervised (SSL) visual representation learning has led to the development of several different proposed frameworks that rely on augmentations of images but use different loss functions. However, there are few theoretically grounded principles to guide practice, so practical implementation of each SSL framework requires several heuristics to achieve competitive performance. In this work, we build on recent analytical results to design practical recommendations for competitive and efficient SSL that are grounded in theory. Specifically, recent theory tells us that existing SSL frameworks are actually minimizing the same idealized loss, which is to learn features that best match the data similarity kernel defined by the augmentations used. We show how this idealized loss can be reformulated to a functionally equivalent loss that is more efficient to compute. We study the implicit bias of using gradient descent to minimize our reformulated loss function, and find that using a stronger orthogonalization constraint with a reduced projector dimensionality should yield good representations. Furthermore, the theory tells us that approximating the reformulated loss should be improved by increasing the number of augmentations, and as such using multiple augmentations should lead to improved convergence. We empirically verify our findings on CIFAR, STL and Imagenet datasets, wherein we demonstrate an improved linear readout performance when training a ResNet-backbone using our theoretically grounded recommendations. Remarkably, we also demonstrate that by leveraging these insights, we can reduce the pretraining dataset size by up to $2\times$ while maintaining downstream accuracy simply by using more data augmentations. Taken together, our work provides theoretically grounded recommendations that can be used to improve SSL convergence and efficiency.

1 Introduction

Unsupervised representation learning, i.e., learning features without human-annotated labels, is critical for progress in computer vision. Modern approaches, grouped under the *self-supervised learning (SSL)* umbrella, build on the core insight that similar images should map to nearby points in the learned feature space – often termed as the *invariance criterion*. Current SSL methods can be broadly categorized into contrastive and non-contrastive algorithms, based on whether they formulate their loss functions using negative samples or not, respectively.

[*]Equal Contribution, [†] Co-senior authorship, Correspondence: blake.richards@mcgill.ca

[1]Mila - Quebec AI Institute & Computer Science, McGill University, Montréal, QC, Canada

[2]UC Berkeley, CA, USA, [3] Meta FAIR, Toronto, ON, Canada

[4]Mila - Quebec AI Institute & Mathematics and Statistics, McGill University, Montréal, QC, Canada

[5]Neurology & Neurosurgery and Montreal Neurological Institute, McGill University, Montréal, QC, Canada

[6]CIFAR Learning in Machines & Brains Program, Toronto, ON, Canada

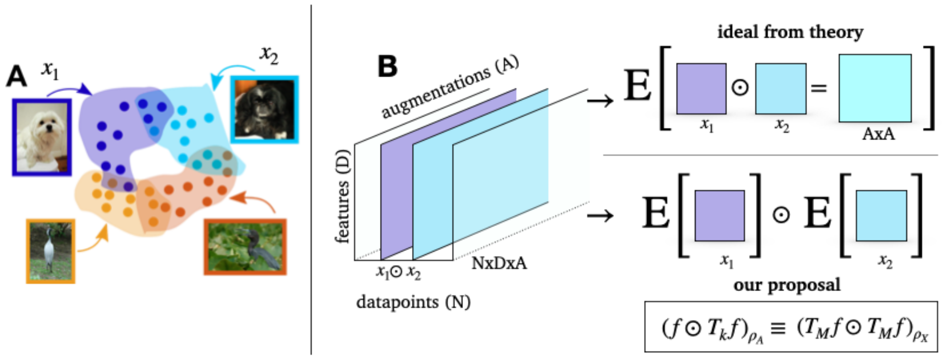

A
x_1
x_2
B
augmentations (A)
features (D)
datapoints (N)
$x_1 \odot x_2$
NxDxA
ideal from theory
$\rightarrow \mathbb{E}\left[\square_{x_1} \odot \square_{x_2} = \square_{AxA} \right]$
$\rightarrow \mathbb{E}\left[\square_{x_1} \right] \odot \mathbb{E}\left[\square_{x_2} \right]$
our proposal
$(f \odot T_k f)_{\rho_A} \equiv (T_M f \odot T_M f)_{\rho_X}$

Figure 1: Design of existing SSL algorithms relies on heuristics. **(A)** Augmentation graphs are common in vision pretraining, providing generalizable features for downstream tasks. (B) We propose an equivalent loss function for SSL pretraining that recovers the same eigenfunctions more efficiently than existing approaches.

Despite this difference in their loss formulations, recent theoretical work has established an equivalence between the contrastive and non-contrastive SSL frameworks [20]. This work shows that these different SSL formulations are ultimately minimizing a loss that encourages the learning of features that best match the data similarity kernel defined by the augmentations used. However, this notion of theoretical equivalence holds only in the limit of ideal pretraining settings, i.e. with access to infinite data and compute budget, and the feature learning behavior of different SSL algorithms in practical scenarios is still not well understood. Therefore, researchers often use empirically driven heuristics that are theoretically ungrounded to design successful applications, such as (i) a high-dimensional projector head for non-contrastive SSL or (ii) the use of two augmentations per image [3]. Moreover, existing SSL algorithms are extremely data-hungry, relying on large-scale datasets [34] or data engines [31] to achieve good representations. While this strategy works exceptionally well in data-rich settings (like training on natural-images), it is not viable in data-constrained settings (like medical imaging), where samples are relatively scarce.

With these challenges in mind, the primary focus of this work is to develop theoretically grounded recommendations for improving the effectiveness and efficiency of feature learning, both with respect to the required compute budget as well as data points. Like any unsupervised representation learning algorithm, features learned through SSL depend on three factors: (i) implicit bias of the architecture, (ii) explicit invariance imposed by data augmentations, (iii) implicit bias of the learning rule. While previous works predominantly studied the role of the model architecture capacity and loss function, and their interplay with data augmentations [10, 45], our approach broadens this perspective by also considering the role of the learning rule (gradient descent) in optimizing these loss functions. Specifically, we extend the previous theoretical findings [45] that unified the desiderata of different SSL algorithms. We reformulate the idealized unifying loss to propose a functionally equivalent loss that is more compute-efficient (see Figure 1). Based on our loss formulation, we provide two practical recommendations that can help improve the efficiency of SSL pipelines while maintaining good performance. First, we show that optimizing the reformulated loss using gradient descent can often reduce the orthogonality among the learned embeddings, thereby leading to an inefficient use of the projector network's capacity. Consequently, we recommend using a stronger orthogonalization constraint to eliminate the requirement of high-dimensional projector heads, thereby significantly reducing the parameter overhead of good feature learning. Second, we show that increasing the number of augmentations leads to a better estimate of the data similarity kernel. Consequently, we recommend using more augmentations to improve optimization convergence and learn better features earlier in training.

We empirically verify our theoretically grounded recommendations using the popular ResNet backbone on benchmark datasets: CIFAR, STL and Imagenet. Strikingly, we show that our multi-augmentation approach can learn good features even with *half* of the samples in the pretraining dataset. Our recommendations provide a path towards making SSL pretraining more data and compute-efficient without harming performance and could unlock massive performance gains in data-constrained setups. In summary, our core contributions are as follows:

- **Efficient SSL loss formulation:** We propose an functionally equivalent and compute-efficient formulation of the SSL desiderata that yields the eigenfunctions of the augmentation-defined data similarity kernel.
- **Role of heuristics:** Based on our loss formulation and the implicit bias of gradient descent in optimizing this loss, we provide a mechanistic explanation for the role of projector dimensionality and the number of data augmentations. Consequently, we empirically demonstrate that low-dimensional projector heads are sufficient and that using more augmentations leads to learning better representations.
- **Data efficient SSL:** Leveraging the convergence benefits of the multi-augmentation SSL framework, we empirically demonstrate that we can learn good features with significantly smaller datasets (up to $2\times$) without harming downstream linear probe performance.

2 Preliminaries

Existing SSL approaches in computer vision In recent years machine learning researchers have developed a number of effective approaches for learning from data without labels. The most popular approaches use augmentations of data points as targets for themselves. One of the first was a Simple framework for Contrastive Learning (SimCLR), which relied on an infoNCE loss with augmentations of an image as positive targets and augmentations of other images as negative samples (to construct the contrastive loss) [13]. Other works have relied on non-contrastive approaches, notably BarlowTwins [44] and VICReg [5]. BarlowTwins, which was inspired by the ideas of the neuroscientist Horace Barlow [6], also uses augmentations of images, but it instead aims to optimize the covariance structure of the representations in order to reduce redundancies in the feature space [44]. Variance Invariance Covariance Regularization (VICReg) was a modification of BarlowTwins that added a variance term in the loss in order to ensure that every feature dimension has a finite variance [5]. In this paper we will focus on non-contrastive methods like BarlowTwins and VICReg, but in line with previous work [45], our results can also be adapted to contrastive methods like SimCLR. Notably, all SSL approaches leverage the use of a multi-layer fully-connected network, termed the projector, during pretraining to transform the features learned by an encoder to some embedding space where the aforementioned properties are imposed through optimization.

Formalizing the self-supervised learning problem Now, we will formalize the unsupervised representation learning problem for computer vision. In particular, we assume access to a dataset $\mathcal{D} = \{x_1, x_2, ..., x_n\}$ with $x_i \in \mathbb{R}^p$ consisting of unlabeled images. The objective is to learn a d-dimensional representation ($d < p$) that is useful across multiple downstream applications. We focus on learning the parameters of a deep neural network $f_\theta \in \mathcal{F}_\Theta$, using the multi-augmentation SSL framework, wherein multiple views of an image are used to optimize the pretraining loss function, $\mathcal{L}_{pretrain}(f_\theta, \mathcal{D})$

Non-Contrastive Self-Supervised Learning (NC-SSL) algorithms impose invariance to data augmentations, while imposing regularization on the geometry of the learned feature space. More generally, $\mathcal{L}_{pretrain}$ can be formulated with two terms (i) $\mathcal{L}_{invariance}$: to learn invariance to data augmentations and (ii) $\mathcal{L}_{collapse}$: regularization to prevent collapsing the feature space to a trivial solution.

$$\mathcal{L}_{pretrain} := \mathcal{L}_{invariance} + \beta\mathcal{L}_{collapse} \tag{1}$$

where β denotes a hyperparameter that controls the importance of the collapse-preventing term relative to the invariance term. This formulation separates features that are invariant to the augmentations from those that are sensitive to them. Intuitively, the ideal feature space is more sensitive to semantic attributes (e.g. “that’s a dog”) and less sensitive to irrelevant attributes (e.g. “direction the dog is facing”), facilitating generalization to new examples.

Data Augmentation graph was introduced by [23] to analyze contrastive losses, like SimCLR [13]. Briefly, we define a graph $\mathcal{G}(\mathcal{A}, \mathcal{W})$ that captures the relationship between images derived from all possible data augmentations. The vertex set $(\mathcal{A}, \rho_A)$ is each augmented sample in a dataset, $\mathcal{X}$, and the adjacency matrix, $\mathcal{W}$, denotes the similarity between pairs of vertices. Let x_0 be an image in $\mathcal{X}$, and let $z = M(x_0) \in \mathcal{A}$ be a random data augmentation of the image, x_0. We define the probability density of reaching z from x_0 via a choice of mapping M:

$$p(z \mid x_0) = \mathbb{P}(z = M(x_0)), \tag{2}$$

Since the mapping is not generally invertible (e.g., cropping), we observe that $p(x_0 \mid z) \neq p(z \mid x_0)$. Using this definition, we now formally define the strength of the edge between nodes $x, z \in \mathcal{A}$ of the augmentation graph as the joint probability of generating augmentations x, z from the same image $x_0 \sim \rho_X$. Notably, the edge strength of the (degree-normalized) augmentation graph is equivalent to the data similarity kernel, defined in [45]. Formally,

$$k^{DAF}(x, z) = w_{xz} := \mathbb{E}_{x_0 \sim \rho_X}\left[\frac{p(x \mid x_0)}{p(x)} \frac{p(z \mid x_0)}{p(z)}\right] \tag{3}$$

The magnitude of w_{xz} captures the augmentation-defined similarity between x and z. A higher value of w_{xz} indicates that both patches are more likely to come from the same image and, thereby, are more similar.

The desiderata of different SSL algorithms can be understood as learning features F that best capture $k^{DAF}(x, z)$, i.e. $F(x)^T F(z) \approx k^{DAF}(x, z)$. Recent theoretical work has shows that different SSL losses can be formulated as special cases of the objective function that recovers the top-d eigenfunctions of $k^{DAF}(x, z)$ [45].

$$\mathcal{L}_{ssl}(F) = \mathbf{E}_{x,z \in \mathcal{A}}\left[(k^{DAF}(x, z) - F(x)^T F(z))^2\right] \tag{4}$$

Note that all rotations of F that don't change its span define an equivalence class of solutions to Equation (4) and make no difference for the downstream generalization of a linear probe. Based on this insight, we define an equivalence among learned feature spaces:

Definition 2.1. Let $F(x) = (f_1(x), \dots f_d(x))$ be a d-dimensional feature vector (a vector of functions). Define the subspace

$$V = V(F) = \{h : X \to \mathbb{R} \mid h(x) = w \cdot F(x), \quad w \in \mathbb{R}^d\} \tag{5}$$

to be the span of the components of F. Given an n-dimensional feature vector, $G(x) = (g_1(x), \dots, g_n(x))$ we say the features G and F are equivalent, if $V(F) = V(G)$.

3 Implicit bias of non-contrastive SSL loss and optimization

We extend the recent theoretical results [45] to propose a compute-efficient reformulation of the loss function of the SSL desiderata that yields equivalent features, i.e. the functions spanning the eigenfunctions of the augmentation-defined data similarity kernel, k^{DAF}. Furthermore, we study the role of gradient descent in optimizing this loss function and uncover a selection and primacy bias in feature learning. Specifically, we find that gradient descent tends to learn the dominant eigenfunctions (eigenfunctions corresponding to larger eigenvalues) earlier during training, and often over-represents these eigenfunctions under weak orthogonalization constraints.

Consequently, we propose employing a stronger orthogonalization constraint during optimization when using a low-dimensional projector to ensure that learned features are equivalent to those learned with a high-dimensional projector. Furthermore, we argue that using more augmentations improves our sample estimate of k^{DAF}, thereby aiding the eigenfunction optimization problem. We dedicate the rest of this section to highlight our key theoretical insights, and practical recommendations that follow them.

3.1 Features in terms of data augmentation kernels

Let us define a kernel operator, T_k, for a positive semi-definite data augmentation kernel, k^{DAF}.

$$T_k f(x) = \mathbb{E}_{z \sim \rho_X}[k(z, x) f(z)] \tag{6}$$

such that Equation (4) can be equivalently written as (Equation 5 of [45])

$$\mathcal{L}_{ssl}(F) = \langle F, (I - T_k)F \rangle_{\rho_{\mathcal{A}}} \tag{7}$$

We can now use Mercer's theorem to factorize k^{DAF} into corresponding spectral features $G : X \to \ell_2$ (where ℓ_2 represents square summable sequences) [16, 17, 32]. However, note that computing k^{DAF} (or T_k) is expensive as it requires computing the overlap among all augmentations of every pair of data points. Instead of computing the eigenfunctions of T_k directly, we propose using an alternative operator T_M:

$$T_M f(x) = \mathbb{E}_{x_0 \sim M(x)}\left[f(x_0)\right] = \sum_{x_0}\left[p(x_0 \mid x) f(x_0)\right] \tag{8}$$

which averages the values of the function, f, over the augmented images $x_0 = M(x)$ of the data, x. We show that $T_M^T T_M$ is equivalent to T_k, and therefore T_M and T_k have shared eigenfunctions.

Theorem 3.1. *Let $G(x)$ be the infinite Mercer features of the backward data augmentation covariance kernels, k^{DAB}. Let $F(x) = (f_1(x), \ldots, f_{N_k}(x))$ be the features given by minimizing the following data augmentation invariance loss*

$$L(F) = \sum_{i=1}^{N_k} \|T_M f_i - f_i\|^2_{L^2(\rho_X)}, \quad \textit{subject to} \quad (f_i, f_j)_{\rho_X} = \delta_{ij} \tag{9}$$

which includes the orthogonality constraint. Then, $V(F) \subset V(G)$, $\lim_{N_k \to \infty} V(F) = V(G)$.

As shown in the Appendix B, $L(F)$ is equivalent to a constrained optimization formulation of the BarlowTwins loss. Furthermore, $L(F)$ with the additional constraint that $(f_i, f_i) \geq \gamma \; \forall i \in \{1, 2 \ldots N_k\}$ is the constrained optimization formulation of the VICReg loss.

3.2 The implicit bias of gradient descent

Next, we investigate how the use of gradient descent for optimizing $L(F)$ influences the characteristics of the learned feature space, $V(F)$. Given the similarity in its form with that of the BarlowTwins loss, we build on recent findings that demonstrate the sequential nature of learning eigenfunctions when optimizing the BarlowTwins loss under a strong orthogonalization regularization [37]. Since strong orthogonalization is seldom used in practice due to instabilities in training [44, 5], we believe studying the learning dynamics under weak orthogonalization regularization (i.e. low values of β in Equation (1)) is more relevant to provide recommendations for practitioners.

Theorem 3.2. *(Informal) Let us denote the span of the feature space at initialization as $V(F_0)$ and after training as $V(F_T)$. For small initialization of the network's weights, the alignment of $V(F_T)$ with the eigenfunctions of T_k depend on two factors: (i) alignment of $V(F_0)$ with the eigenfunctions of T_k; (ii) singular values of T_k.*

Under weak orthogonalization constraints, the network tends to learn features that are strongly aligned with eigenfunctions corresponding to large singular values. We refer to this property as the "selection" bias of gradient descent, wherein gradient descent selects certain eigenfunctions based on the corresponding singular values. This selection bias leads to redundancy among the learned feature space, thereby reducing the effective dimensionality of the network's output space compared to its ambient dimensionality. We will leverage this finding to improve the parameter overhead of good feature learning using BarlowTwins and VICReg loss frameworks.

3.3 Takeaway 1: Low-dimensional projectors can yield good representations

Given the proximity of the formulation of Equation (9) to that of BarlowTwins and VICReg losses, we will leverage existing heuristics that have been shown to work in practice. As such, BarlowTwins and VICReg frameworks call for high-dimensional projectors while using a weak orthogonalization regularization to facilitate good feature learning. We know, from Theorem 3.1, that the eventual goal of these frameworks is to learn the eigenfunctions of the underlying data similarity graph. For example, since the intrinsic dimensionality of Imagenet is estimated to be ~ 40 [33], it is not unreasonable to expect that the span of desired features would be of similar dimensionality. It is, thus, intriguing that the current practice would suggest using an ~ 8192-dim projector head to capture the intricacies of the corresponding augmentation-defined data similarity kernel. This discrepancy can be explained by analyzing the learning dynamics, as in Theorem 3.2. Notably, a high-dimensional projector is likelier to have a greater initialization span than its low-dimensional counterpart, thereby increasing the alignment between $V(F_0)$ and relevant eigenfunctions of T_k. We hypothesize that a stronger orthogonalization constraint for low-dimensional projectors can rectify this issue, reducing the redundancy in the network's output space and rendering it sufficient for good feature learning.

3.4 Takeaway 2: Multiple augmentations improve kernel approximation

By comparing the invariance criterion formulation in the standard BarlowTwins and VICReg losses to Equation (7), it can be inferred that current practices use a sample estimate of T_k. Using only two augmentations per sample yields a noisy estimate of T_k, yielding spurious eigenpairs [42] (see Appendix C). These spurious eigenpairs add stochasticity in the learning dynamics, and coupled with Theorem 3.2, increase the redundancy in the learned feature space [12]. We hypothesize that improving this estimation error by increasing the number of augmentations could alleviate this issue and improve the speed and quality of feature learning.

Of course, increasing the number of augmentations (m) in the standard BarlowTwins and VICReg loss improves the estimate of T_k but comes with added compute costs – a straightforward approach would involve calculating the invariance loss for every pair of augmentations, resulting in $\mathcal{O}(m^2)$ operations. However, Theorem 3.1 proposes an alternative method that uses the sample estimate of T_M, thereby requiring only $\mathcal{O}(m)$ operations, and hence is computationally more efficient while yielding functionally equivalent features (see Appendix B). In summary, Theorem 3.1 establishes a mechanistic role for the number of data augmentations, paving the way for a computationally efficient multi-augmentation framework:

$$\widehat{L}(F) = \mathbb{E}_{x\sim\rho_X}\left[\sum_{i=1}^{N_k}\sum_{j=1}^{m}\|\overline{f_i(x)} - f_i(x_j)\|^2_{L^2(\rho_X)}\right], \quad \text{subject to} \quad (f_i, f_j)_{\rho_X} = \delta_{ij} \tag{10}$$

where $\overline{f_i(x)} = \frac{1}{m}\sum_{j=1}^{m} f_i(x_j)$ is the sample estimate of $T_M f_i(x)$.

4 Experiments

In our experiments, we seek to (i) provide empirical support for our theoretical insights and (ii) present practical primitives for designing efficient SSL routines. Since our proposed loss function is closest to the formulation of BarlowTwins/VICReg, we present empirical evidence comparing our proposal to these baselines. In summary, with extensive experiments across learning algorithms (BarlowTwins & VICReg) and training datasets (CIFAR-10, STL-10 & Imagenet-100), we establish the following:

- **low-dimensional projectors** can yield *good representations*.
- **multi-augmentation** improves downstream accuracy, as well as convergence rate.
- multi-augmentation **improves sample efficiency** in SSL pretraining, i.e., recovering similar performance with significantly fewer unique unlabelled samples.

Experiment Setup: We evaluate the effectiveness of different pretraining approaches using image classification as the downstream task. Across all experiments, we pretrain a Resnet feature encoder backbone for 100 epochs (see Appendix E.1 for longer pretraining results) and use linear probing on the learned representations[1]. All runs are averaged over 3 seeds; error bars indicate standard deviation. Other details related to optimizers, learning rate, etc., are presented in the Appendix D.

4.1 Low-dimensional projectors can yield good representations

pdim	Barlow Twins		VICReg	
	fixed β	optimal β^*	fixed β	optimal β^*
64	73.6 ± 0.9	82.1 ± 0.2	68.9 ± 0.2	81.9 ± 0.1
256	75.9 ± 0.7	**83.4 ± 0.4**	75.3 ± 0.2	81.9 ± 0.3
1024	81.3 ± 1.0	82.9 ± 0.3	79.2 ± 0.9	**82.5 ± 0.9**
8192	82.2 ± 0.4	82.2 ± 0.4	80.4 ± 1.5	80.4 ± 1.5

Table 1: Optimizing for orthogonality appropriately allows low-dimensional projectors to match the performance (on CIFAR-10) of much higher-dimensional projectors.

Existing works recommend using high-dimensional MLPs as projectors (e.g., d=8192 for Imagenet in [44, 5]), and show significant degradation in performance when using lower-dimensional projectors

[1]Code: https://github.com/kumarkrishna/fastssl

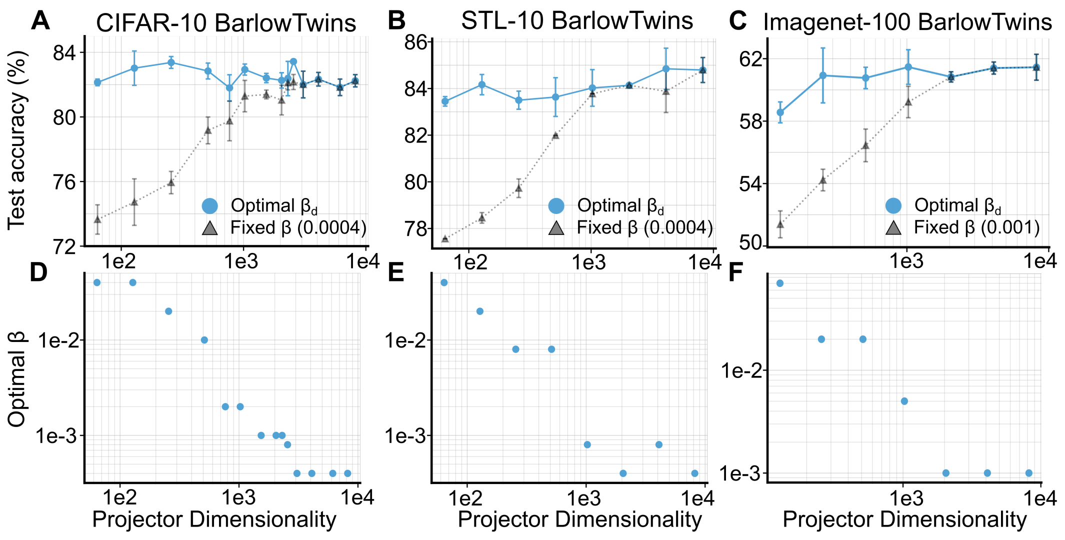


Figure 2: Low-dimensional projectors can yield good representations. We demonstrate that using a higher orthogonality constraint, β, for lower projector dimensionality can achieve similar performance over a wide range of projector dimensions (d).

for a fixed redundancy coefficient (β). To reproduce this result, we run a grid search to find the optimal coefficient (β^*_{8192}) for $d = 8192$ and show that performance progressively degrades for lower d if the same coefficient β^*_{8192} is reused for $d \in \{64, 128, 256, 512, 1024, 2048, 4096, 8192\}$.

Our insights in Section 3.3 suggest low-dimensional projectors should recover similar performance with appropriate orthogonalization. To test this, we find the best β by performing a grid search independently for each $d \in \{64, 128, 256, 512, 1024, 2048, 4096, 8192\}$. As illustrated in Figure 2, using low-dimensional projectors yield features with similar downstream task performance, compared to the features obtained using high-dimensional projectors. Strikingly, we also observe that the optimal $\beta_d \propto 1/d$, which aligns with our theoretical insights.

Recommendation: Start with low-dimensional projector, using $\beta = \mathcal{O}(\frac{1}{d})$, and sweep over $(pdim = d, \beta = \mathcal{O}\left(\frac{1}{d}\right))$ if needed.

4.2 Multiple Augmentations Improve Performance and Convergence

Although some SSL pretraining approaches, like SWaV [11], incorporate more than two views, the most widely used heuristic in non-contrastive SSL algorithms involves using two views jointly encoded by a shared backbone. In line with this observation, our baselines for examining the role of multiple augmentations use two views for computing the cross-correlation matrix.

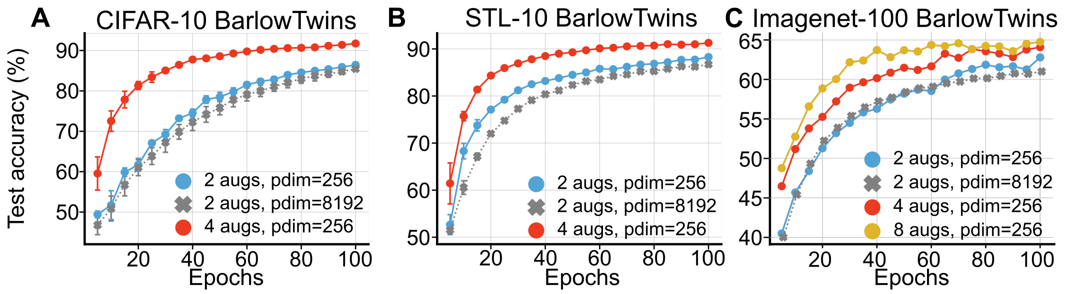


Figure 3: Using multiple augmentations improves representation learning performance and convergence. (A-C) Across BarlowTwins for CIFAR-10, STL-10 and Imagenet-100 pretraining, using 4 augmentations instead of 2 helps improve performance. Please see Appendix E.3 for more results.

augs	pdim	**BarlowTwins** Time (min)	**VICReg** Time (min)
2	8192	99.36 ± 0.01	94.36 ± 0.01
2	256	62.34 ± 0.06	51.73 ± 0.04
4	256	$\mathbf{43.09 \pm 0.20}$	$\mathbf{39.02 \pm 0.05}$

Table 2: Using multiple augmentations yields faster convergence, with reduced time to reach baseline performance on CIFAR-10, i.e. performance of feature encoder pretrained with an 8192-dim projector and 2 augmentations.

To demonstrate the role of multiple augmentations in pretraining, we adapt the invariance criterion of BarlowTwins/VICReg to be in line with Equation (10). In particular, for $\#augs \in \{2, 4, 8\}$, we pretrain a Resnet-50 encoder with our proposed loss. Building on the insight from the previous section, we use a 256-dimensional projector head for all multi-augmentation experiments.

In Figure 3, we track the downstream performance of the pretrained models across training epochs. For performance evaluation, we use the linear evaluation protocol as outlined by [14]. Figure 3(A-C) shows that pretraining with multiple augmentations outperforms the 2-augmentation baseline. Furthermore, we observe that the four-augmentation pretrained models converge faster (both in terms of the number of epochs and wall-clock time) than their two-augmentation counterparts (see Figure 3(D-F)). Additionally, we show in Appendix E.2 that our framework can also be applied to multi-augmentation settings like SWaV, where not all augmentations are of the same resolution.

Recommendatation: Using multiple augmentations (> 2) is likely to improve convergence as well as downstream accuracy.

4.3 Sample Efficient Multi-augmentation Learning

Data Augmentation can be viewed as a form of data inflation, where the number of training samples is increased by k (for k augmentations). In this section, we examine the role of multi-augmentation in improving sample efficiency. In particular, we are interested in understanding if the same performance can be achieved with a fraction of the pretraining dataset, simply by using more augmentations.

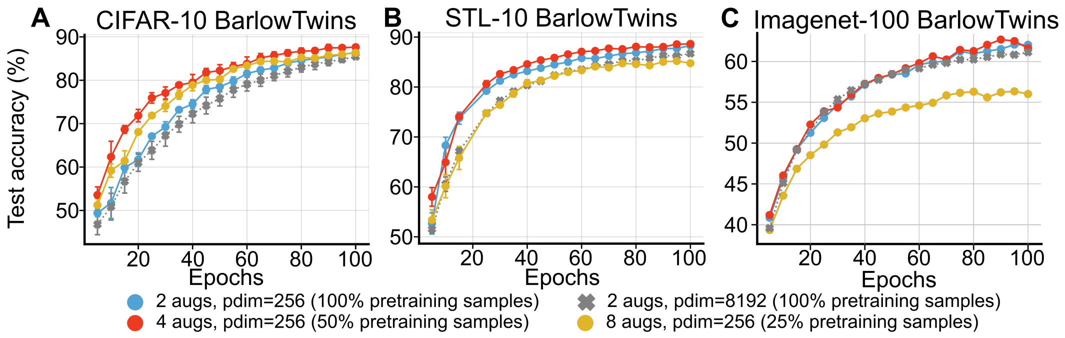


Figure 4: Multi-augmentation improves sample efficiency, recovering similar performance with significantly fewer unique samples in the pretraining dataset. Across BarlowTwins pretraining on CIFAR-10, STL-10 and Imagenet-100 for the same effective dataset size ($\#augs \times \#unique\_samples$), using more patches improves performance at the same epoch (A-C). However, a tradeoff exists wherein more data augmentations fail to improve performance in the scarce data regime.

To examine the relation between the number of augmentations and sample efficiency, we fixed the effective size of the inflated dataset. This is achieved by varying the fraction of the unique samples in the pretraining dataset depending on the number of augmentations $k \in \{2, 4, 8\}$, e.g., we use 50% of the dataset for 4 views. We then evaluate the performance of the pretrained models on the downstream task, where the linear classifier is trained on the same set of labeled samples. Strikingly, Figure 4 shows that using multiple augmentations can achieve similar (sometimes even better) performance with lesser pretraining samples, thereby indicating that more data augmentations can be used for feature learning to compensate for smaller pretraining datasets.

augs	pdim	Percentage of Dataset	**BarlowTwins** Time (min)	**VICReg** Time (min)
2	8192	100 %	63.43 ± 0.02	66.05 ± 0.01
2	256	100 %	39.52 ± 0.04	40.64 ± 0.04
4	256	50 %	28.25 ± 0.01	$\mathbf{32.39 \pm 0.01}$
8	256	25 %	$\mathbf{27.74 \pm 0.01}$	34.76 ± 0.01

Table 3: Time required to pass 80% accuracy on CIFAR-10 when pretraining on fraction of the dataset, while using multiple augmentations. See Figure 5 for further discussion.

Recommendation: In a low-data regime, using diverse & multiple augmentations can be as effective as acquiring more unique samples.

5 Related Work

Self-Supervised Pretraining requires significant compute resources and most practitioners rely on empirical heuristics (see SSL cookbook [3] for a summary). While recent advances in SSL theory explore learning dynamics in linear (or shallow) models [40, 41], with a focus on understanding dimensionality collapse [21, 25], the theoretical underpinnings of most of the heuristics considered essential for good feature learning, are missing.

Contrastive SSL has received more theoretical attention, owing to its connection with metric learning and noise contrastive estimation [30, 4, 26]. In particular, HaoChen *et al.* [23] provide a theoretical framework for the SimCLR loss from an augmentation graph perspective, which leads to practical recommendations. Subsequently, Garrido *et al.* [20] establish a duality between contrastive and non-contrastive learning objectives, further bridging the gap between theory and practice.

Non-contrastive SSL algorithms' theoretical foundations have received more attention recently [10, 45]. Prior works [2, 20, 19] have demonstrated that with modified learning objectives, low-dimensional projectors yield representations with good downstream performance. Similarly, previous works have demonstrated notable performance boosts when using a multi-patch framework in contrastive [18] and non-contrastive SSL [11, 43]. However, the theoretical basis for the benefits and trade-offs of either low-dimensional projectors or multiple augmentations is largely unclear. It is worth noting that Schaeffer *et al.* [35] present an information-theoretic perspective of the recently proposed non-contrastive SSL loss that leverages multiple augmentations, namely MMCR [43], but the computational advantages of using multiple augmentations on the learning dynamics is an active area of research.

Deep Learning theory has made significant strides in understanding the optimization landscape and dynamics of supervised learning [1]. In concurrent works [45, 10], the interplay between the inductive bias of data augmentations, architectures, and generalization has been explored from a purely theoretical perspective, establishing an equivalence among different SSL losses [45]. Furthermore, Simon *et al.* [37] used a more straightforward formulation of the BarlowTwins loss and investigated the learning dynamics in linearized models for the case when the invariance and orthogonalization losses have equal penalties. Although such a setting rarely used in practice, their approach serves as an inspiration for our work in studying the learning dynamics of non-contrastive SSL losses.

6 Discussion

Summary: Our work builds on existing theoretical results that establish an equivalence among different SSL frameworks, and proposes a compute-efficient reformulation of the common SSL loss. Using this loss reformulation and a study of the optimization dynamics, we proposed practical recommendations to improve the sample and compute efficiency of SSL algorithms. Specifically, we recommended low-dimensional projectors with increased orthogonality constraints and multi-augmentation frameworks, and we verified the effectiveness of these recommendations empirically. It is worth noting that our multi-augmentation formulation improves the efficiency of learning without altering the desiderata of SSL, i.e. the network learns the same feature space using our proposed multi-augmentation framework as with the original SSL formulation in the limit of infinite pretraining budget. To demonstrate this equivalence between the original SSL loss and our proposed version, we show in Appendix E.1 that longer pretraining on the 2-augmentation loss leads to similar downstream performance as the multi-augmentation versions (4 and 8 augmentations).

We also showed that the multi-augmentation framework can be used to learn good features from fewer unique samples in the pretraining dataset simply by improving the estimation of the data augmentation kernel. This result has direct implications on improving the Pareto frontier of samples-vs-performance for SSL pretraining, wherein we can achieve better downstream performance when limited number of samples are available in the pretraining dataset.

Pareto Optimal SSL In the context of sample efficiency, training a model using two augmentations with different fractions of the dataset leads to a natural Pareto frontier, i.e., training on the full dataset achieves the best error but takes the most time (**Baseline (2-Aug)**). Our extensive experiments demonstrate that using more than two augmentations improves the overall Pareto frontier, i.e., achieves better convergence while maintaining accuracy (**Multi-Aug**). Strikingly, as shown in Figure 5, we observe that we can either use a larger pretraining dataset or more augmentations for a target error level. Therefore, the number of augmentations can be used as a knob to control the sample efficiency of the pretraining routine.

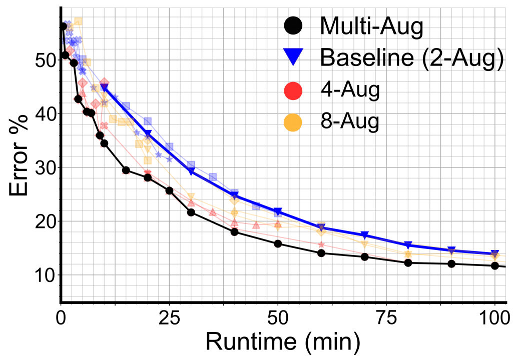


Figure 5: Using > 2 augmentations with a fraction of the dataset improves overall Pareto frontier, speeding runtime up to $\sim 2\times$.

Connections to Downstream performance: While our core theoretical results are aimed at accelerating convergence of the SSL loss itself, our empirical results highlight an improved downstream task performance earlier during pretraining. While this discrepancy might seem counter-intuitive at first, it is worth noting that the SSL loss inherent influences downstream performance as it encourages clustering of semantically similar images in the representation space. Such clustering properties in the representation space facilitates easier classification through methods k-nearest neighbors or linear decoding for a large number of tasks that rely on the semantic content of images. Previous works [21, 2, 19, 38] have discussed in detail how certain geometric properties of the learned representation space are connected to the linear classification performance for arbitrary decision boundaries, in expectation. However, an in-depth analysis of downstream tasks that are more amenable to linear decoding from the learned SSL representation space requires framing metrics of alignment between the pretraining objective (SSL desiderate) and the downstream task labels, and is an active area of research.

Open Questions: Looking ahead, it would be exciting to extend this analysis to other categories of SSL algorithms, such as Masked AutoEncoders (MAE). Furthermore, our insights provide opportunities to explore sample-efficient methods that rely on less data, which is particularly important in critical domains such as medical imaging, where data is often relatively scarce and expensive. On a different note, it is intriguing that animals often spend extended periods of time exploring novel objects, likely to gain multiple views of the object [7, 29]. Given the theoretical underpinnings of the computational benefits of multi-augmentation SSL outlined in our work, it would be exciting to develop models of biological learning that leverage these insights and enable sample-efficient continual learning in similar environments.

Limitations: Our algorithm relies on multiple augmentations of the same image to improve the estimation of the data-augmentation kernel. Though this approach speeds up the learning process, it also adds some extra computational overhead, which means that the impact of faster learning on wall-clock time is less than might be hoped for. One way to mitigate the effects of this limitation would be to scale up to a multi-GPU setting, since the computations for each augmentation can be run on a separate GPU in parallel. This could help ensure that the improved speed of learning directly translates to a significantly reduced wall-clock time for training.

Impact Statement: The goal of our work is to advance the general field of visual representation learning. Although there are potential downstream societal consequences of our work, we feel there are no direct consequences that must be specifically highlighted here.

Acknowledgements

The authors would like to thank Arnab Kumar Mondal for insightful discussions that helped shape the project's scope, Colleen Gillon for aesthetic contribution to the figures and Florian Bordes for helping setup the FFCV-SSL library. The authors are also grateful to Aidan Sirbu, Chen Sun, Jonathan Cornford, Roman Pogodin, Zahraa Chorghay and the anonymous reviewers whose comments and suggestions have significantly enhanced the quality and presentation of the results. This research was generously supported by Vanier Canada Graduate Scholarship (A.G.); NSERC (Discovery Grant: RGPIN-2020-05105; RGPIN-2018-04821; Discovery Accelerator Supplement: RGPAS-2020-00031; Arthur B. McDonald Fellowship: 566355-2022), Healthy Brains, Healthy Lives (New Investigator Award: 2b-NISU-8), and CIFAR Learning in Machines & Brains Program (B.A.R.); Canada CIFAR AI Chair program (A.O. & B.A.R.). The authors also acknowledge the material support of NVIDIA in the form of computational resources, as well as the compute resources, software and technical help provided by Mila (mila.quebec).

References

[1] Madhu S Advani, Andrew M Saxe, and Haim Sompolinsky. High-dimensional dynamics of generalization error in neural networks. *Neural Networks*, 132:428–446, 2020.

[2] Kumar K Agrawal, Arnab Kumar Mondal, Arna Ghosh, and Blake Richards. α-req: Assessing representation quality in self-supervised learning by measuring eigenspectrum decay. *Advances in Neural Information Processing Systems*, 35:17626–17638, 2022.

[3] Randall Balestriero, Mark Ibrahim, Vlad Sobal, Ari Morcos, Shashank Shekhar, Tom Goldstein, Florian Bordes, Adrien Bardes, Gregoire Mialon, Yuandong Tian, et al. A cookbook of self-supervised learning. *arXiv preprint arXiv:2304.12210*, 2023.

[4] Randall Balestriero and Yann LeCun. Contrastive and non-contrastive self-supervised learning recover global and local spectral embedding methods. In *NeurIPS*, 2022.

[5] Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021.

[6] Horace B Barlow et al. Possible principles underlying the transformation of sensory messages. *Sensory communication*, 1(01):217–233, 1961.

[7] Daniel E Berlyne. Novelty and curiosity as determinants of exploratory behaviour. *British journal of psychology*, 41(1):68, 1950.

[8] Florian Bordes, Randall Balestriero, and Pascal Vincent. Towards democratizing joint-embedding self-supervised learning, 2023.

[9] Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[10] Vivien Cabannes, Bobak Kiani, Randall Balestriero, Yann LeCun, and Alberto Bietti. The ssl interplay: Augmentations, inductive bias, and generalization. In *International Conference on Machine Learning*, pages 3252–3298. PMLR, 2023.

[11] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European conference on computer vision (ECCV)*, pages 132–149, 2018.

[12] Feng Chen, Daniel Kunin, Atsushi Yamamura, and Surya Ganguli. Stochastic collapse: How gradient noise attracts sgd dynamics towards simpler subnetworks. *arXiv preprint arXiv:2306.04251*, 2023.

[13] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[14] Yubei Chen, Adrien Bardes, Zengyi Li, and Yann LeCun. Intra-instance vicreg: Bag of self-supervised image patch embedding. *arXiv preprint arXiv:2206.08954*, 2022.

[15] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223. JMLR Workshop and Conference Proceedings, 2011.

[16] Zhijie Deng, Jiaxin Shi, Hao Zhang, Peng Cui, Cewu Lu, and Jun Zhu. Neural eigenfunctions are structured representation learners. *arXiv preprint arXiv:2210.12637*, 2022.

[17] Zhijie Deng, Jiaxin Shi, and Jun Zhu. Neuralef: Deconstructing kernels by deep neural networks. In *International Conference on Machine Learning*, pages 4976–4992. PMLR, 2022.

[18] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. With a little help from my friends: Nearest-neighbor contrastive learning of visual representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9588–9597, 2021.

[19] Quentin Garrido, Randall Balestriero, Laurent Najman, and Yann Lecun. Rankme: Assessing the downstream performance of pretrained self-supervised representations by their rank. In *International Conference on Machine Learning*, pages 10929–10974. PMLR, 2023.

[20] Quentin Garrido, Yubei Chen, Adrien Bardes, Laurent Najman, and Yann LeCun. On the duality between contrastive and non-contrastive self-supervised learning. In *The Eleventh International Conference on Learning Representations*, 2023.

[21] Arna Ghosh, Arnab Kumar Mondal, Kumar Krishna Agrawal, and Blake Richards. Investigating power laws in deep representation learning. *arXiv preprint arXiv:2202.05808*, 2022.

[22] I Gohberg. Classes of linear operator theory. *Advances and Applications*, 49, 1990.

[23] Jeff Z HaoChen, Colin Wei, Adrien Gaidon, and Tengyu Ma. Provable guarantees for self-supervised deep learning with spectral contrastive loss. *Advances in Neural Information Processing Systems*, 34:5000–5011, 2021.

[24] Lajos Horváth and Piotr Kokoszka. *Inference for functional data with applications*, volume 200. Springer Science & Business Media, 2012.

[25] Li Jing, Pascal Vincent, Yann LeCun, and Yuandong Tian. Understanding dimensional collapse in contrastive self-supervised learning. *arXiv preprint arXiv:2110.09348*, 2021.

[26] Daniel D. Johnson, Ayoub El Hanchi, and Chris J. Maddison. Contrastive learning can find an optimal basis for approximately view-invariant functions. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[27] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Technical Report*, 2009.

[28] Guillaume Leclerc, Andrew Ilyas, Logan Engstrom, Sung Min Park, Hadi Salman, and Aleksander Madry. ffcv. https://github.com/libffcv/ffcv/, 2022. commit 3a12966.

[29] Marianne Leger, Anne Quiedeville, Valentine Bouet, Benoît Haelewyn, Michel Boulouard, Pascale Schumann-Bard, and Thomas Freret. Object recognition test in mice. *Nature protocols*, 8(12):2531–2537, 2013.

[30] Yazhe Li, Roman Pogodin, Danica J Sutherland, and Arthur Gretton. Self-supervised learning with kernel dependence maximization. *Advances in Neural Information Processing Systems*, 34:15543–15556, 2021.

[31] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[32] David Pfau, Stig Petersen, Ashish Agarwal, David GT Barrett, and Kimberly L Stachenfeld. Spectral inference networks: Unifying deep and spectral learning. *arXiv preprint arXiv:1806.02215*, 2018.

[33] Phil Pope, Chen Zhu, Ahmed Abdelkader, Micah Goldblum, and Tom Goldstein. The intrinsic dimension of images and its impact on learning. In *International Conference on Learning Representations*, 2021.

[34] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

[35] Rylan Schaeffer, Victor Lecomte, Dhruv Bhandarkar Pai, Andres Carranza, Berivan Isik, Alyssa Unell, Mikail Khona, Thomas Yerxa, Yann LeCun, SueYeon Chung, et al. Towards an improved understanding and utilization of maximum manifold capacity representations. *arXiv preprint arXiv:2406.09366*, 2024.

[36] Kendrick Shen, Robbie M Jones, Ananya Kumar, Sang Michael Xie, Jeff Z HaoChen, Tengyu Ma, and Percy Liang. Connect, not collapse: Explaining contrastive learning for unsupervised domain adaptation. In *International Conference on Machine Learning*, pages 19847–19878. PMLR, 2022.

[37] James B Simon, Maksis Knutins, Liu Ziyin, Daniel Geisz, Abraham J Fetterman, and Joshua Albrecht. On the stepwise nature of self-supervised learning. *arXiv preprint arXiv:2303.15438*, 2023.

[38] Vimal Thilak, Chen Huang, Omid Saremi, Laurent Dinh, Hanlin Goh, Preetum Nakkiran, Joshua M Susskind, and Etai Littwin. Lidar: Sensing linear probing performance in joint embedding ssl architectures. In *The Twelfth International Conference on Learning Representations*, 2024.

[39] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*, pages 776–794. Springer, 2020.

[40] Yuandong Tian, Xinlei Chen, and Surya Ganguli. Understanding self-supervised learning dynamics without contrastive pairs. In *International Conference on Machine Learning*, pages 10268–10278. PMLR, 2021.

[41] Yuandong Tian, Lantao Yu, Xinlei Chen, and Surya Ganguli. Understanding self-supervised learning with dual deep networks. *arXiv preprint arXiv:2010.00578*, 2020.

[42] Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. *arXiv preprint arXiv:1011.3027*, 2010.

[43] Thomas Yerxa, Yilun Kuang, Eero Simoncelli, and SueYeon Chung. Learning efficient coding of natural images with maximum manifold capacity representations. *Advances in Neural Information Processing Systems*, 36:24103–24128, 2023.

[44] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*, pages 12310–12320. PMLR, 2021.

[45] Runtian Zhai, Bingbin Liu, Andrej Risteski, Zico Kolter, and Pradeep Ravikumar. Understanding augmentation-based self-supervised representation learning via rkhs approximation. *arXiv preprint arXiv:2306.00788*, 2023.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our main claims are backed by theoretical and empirical results. Theorem 3.1 presents our functionally-equivalent compute-efficient formulation of the SSL objective, and Theorem 3.2 demonstrates the implicit bias of gradient descent during optimizing the SSL loss. Our empirical results demonstrate the utility of our theoretical insights in improving the parameter overhead of good feature learning, optimization convergence and the sample efficiency.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have a section on limitations in the discussion.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The formal statements alongside proofs are presented in the supplementary material (Appendices A to C).

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide implementation details in the supplementary material (Appendix D). We have also released our code base in the public github repo, FastSSL, to facilitate the implementation of our proposed framework.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes] .

   Justification: We use open-access datasets, like CIFAR, STL and Imagenet. Our code base can be found in the public github repo, FastSSL

   Guidelines:

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We present details of the experiment setup and results in Section 4 of the main paper, and additional implementation details in Appendix D.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: We report standard error bars, computed over 3 seeds, for all result plots and tables.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: All our CIFAR and STL experiments were done on a single 48-GB RTX8000 GPU and all Imagenet experiments were performed on 2 40-GB A100 GPUs. All experiments were performed on the Mila cluster, aided by compute resources, software and technical help provided by Mila (mila.quebec).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: We adhere to the NeurIPS Code of Ethics in our work, and research in general.

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We add a statement on societal impact in the Discussion section. Since the goal of our work is to advance the general field of visual representation learning, we feel there are no direct consequences that must be specifically highlighted here. Although we recognize that there might be potential downstream consequences that warrant attention while building intelligent systems that leverage this work.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We do not release any new data or state-of-the-art models.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We acknowledge and cite the datasets and model architectures used in this work. Our codebase is publicly available on our github repo, FastSSL. Moreover, our codebase relies on the Python packages of PyTorch, FFCV [28] and FFCV-SSL [8], which are referred to in the github repo README.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: No new assets are released.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

Justification: Our work does not involve crowdsourcing nor research with human subjects.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Our work does not involve research with human subjects.

A Hilbert Space of functions

A.1 Functions and inner product space

Definition A.1. Given X, ρ_X, and $f, g : X \to \mathbb{R}$, define the $L^2(\rho_X)$ inner product and norm, respectively,

$$(f,g)_{\rho_X} = \int f(x)g(x)d\rho_X(x), \quad \|f\|^2_{\rho_X} = (f,f)_{\rho_X} \tag{11}$$

Define

$$L^2(\rho, X) = \left\{ f : X \to \mathbb{R} \mid \|f\|^2_{\rho_X} < \infty \right\}$$

to be the (equivalence class) of functions with finite ρ_X norm.

A.2 Spectral theory

In this section we quote the relevant (abstract) Hilbert Space theory.

Definition A.2 (Spectral Operator)**.** Given orthogonal functions, $\Phi = (\phi_i)_{i \in I}$ in $L^2(\rho_X)$, and non-negative $\Lambda = (\lambda_i)_{i \in I}$, with $\|\Lambda\|_2^2 = \sum_{i \in I} \lambda_i^2 < \infty$. Call (Φ, Λ) a spectral pair and define the corresponding spectral operator by

$$T_{\Phi,\Lambda}(h) = \sum_{j=1}^{\infty} \lambda_j \left(h, \phi_j\right) \phi_j, \tag{12}$$

Theorem A.3 (Spectral Decomposition)**.** *Suppose H is a Hilbert space. A symmetric positive-definite Hilbert-Schmidt operator $T : \mathbb{H} \to \mathbb{H}$ admits the spectral decomposition equation 12 with orthonormal ϕ_j which are the eigenfunctions of T, i.e. $T\left(\phi_j\right) = \lambda_j \phi_j$. The ϕ_j can be extended to a basis by adding a complete orthonormal system in the orthogonal complement of the subspace spanned by the original ϕ_j.*

Remark A.4. The ϕ_j in equation 12 can thus be assumed to form a basis, but some λ_j may be zero.

From [24]. Theorem proved in [22]. Denote by $\mathcal{L}$ the space of bounded (continuous) linear operators on $\mathbb{H}$ with the norm

$$\|T\|_{\mathcal{L}} = \sup\{\|T(x)\| \mid \|x\| \leq 1\}.$$

Definition A.5 (Compact Operators)**.** An operator $T \in \mathcal{L}$ is said to be compact if there exist two orthonormal bases $\{g_j\}$ and $\{f_j\}$, and a real sequence $\{\lambda_j\}$ converging to zero, such that

$$T(h) = \sum_{j=1}^{\infty} \lambda_j (h, g_j) f_j, \quad h \in \mathbb{H}, \tag{Compact}$$

The λ_j may be assumed positive. The existence of representation equation Compact is equivalent to the condition: T maps every bounded set into a compact set. Compact operators are also called completely continuous operators. Representation equation Compact is called the singular value decomposition.

Definition A.6 (Hilbert-Schmidt Operators)**.** A compact operator admitting representation equation Compact is said to be a Hilbert-Schmidt operator if $\sum_{j=1}^{\infty} \lambda_j^2 < \infty$. The space $\mathcal{S}$ of Hilbert-Schmidt operators is a separable Hilbert space with the scalar product

$$\langle T_1, T_2 \rangle_{\mathcal{S}} = \sum_{i=1}^{\infty} \left(T_1\left(f_i\right), T_2\left(f_i\right)\right), \tag{13}$$

where $\{f_i\}$ is an arbitrary orthonormal basis. Note the value of equation 13 is independent of the basis. The corresponding norm is

$$\|T\|^2_{\mathcal{S}} = \sum_{j \geq 1} \lambda_j^2 \tag{HS}$$

One can show that

$$\|T\|_{\mathcal{L}} \leq \|T\|_{\mathcal{S}}$$

Definition A.7. An operator $T \in \mathcal{L}$ is said to be symmetric if
$$\langle T(f), g\rangle = \langle f, T(g)\rangle, \quad f, g \in \mathbb{H},$$
and positive-definite if
$$\langle T(f), f\rangle \geq 0, \quad f \in \mathbb{H}.$$
(An operator with the last property is sometimes called positive semidefinite, and the term positive-definite is used when the inequality is strict.)

B Data augmentation kernel perspective of non-contrastive SSL

Theorem B.1. *Let $G(x)$ be the infinite Mercer features of the backward data augmentation covariance kernels, k^{DAB}. Let $F(x) = (f_1(x), f_2(x), \dots, f_k(x))$ be the features given by minimizing the following data augmentation invariance loss*
$$L(F) = \sum_{i=1}^{N_k} \|T_M f_i - f_i\|^2_{L^2(\rho_X)}, \quad \textit{subject to} \quad (f_i, f_j)_{\rho_X} = \delta_{ij} \tag{14}$$
which includes the orthogonality constraint. Then, $V(F) \subset V(G)$, $V(F) \to V(G)$ as $N_k \to \infty$.

The idea of the proof uses the fact that, as linear operators, $T_{k^{DAB}} = T_M^\top T_M$ and that $T_{k^{DAF}} = T_M T_M^\top$. Then we use spectral theory of compact operators, which is analogue of the Singular Value Decomposition in Hilbert Space, to show that eigenfunctions of $T_M^\top T_M$ operator are the same as those obtained from optimizing $L(F)$. A similar result can be obtained using k^{DAF} and $T_M^\top$.

Note that $L(F)$ is the constrained optimization formulation of the BarlowTwins loss. Furthermore, $L(F)$ with the additional constraint that $(f_i, f_i) \geq \gamma \; \forall i \in \{1, 2 \dots N_k\}$ is the constrained optimization formulation of the VICReg loss.

B.1 Proof of theorem 3.1

We show we can factor the linear operator, leading to a practical algorithm. Here, we show that we can capture the backward data augmentation kernel with the forward data augmentation averaging operator

Lemma B.2. *Using the definitions above, and with k in equation 6 given by k^{DAB},*
$$T_k = T_M^\top T_M$$

Proof. First, define the non-negative definite bilinear form
$$B^{VAR}(f, g) = (T_M f, T_M g)_{\rho_X} \tag{15}$$
Given the backwards data augmentation covariance kernel, k^{DAB}, define
$$B^{DAB}(f, g) = (T_k f, g)_{\rho_X}$$
We claim, that
$$B^{VAR} = B^{DA,B} \tag{16}$$
This follows from the following calculation,
$$\begin{aligned} B^{DA,B}(f,g) &= (T_k f, g)_{\rho_X} && (17)\\ &= \mathbb{E}_x[T_k f(x), g(x)] = \mathbb{E}_x \mathbb{E}_z [k_{DA,B}(z,x) f(z) g(x)] && (18)\\ &= \mathbb{E}_x \mathbb{E}_z \mathbb{E}_{x_0} \left[\frac{p(x_0 \mid x)}{\rho(x_0)} \frac{p(x_0 \mid z)}{\rho(x_0)} f(z) g(x) \right] && (19)\\ &= \mathbb{E}_{x_0} \left[\sum_x \left(\frac{\rho(x) p(x_0 \mid x)}{\rho(x_0)} g(x) \right) \sum_z \left(\frac{\rho(z) p(x_0 \mid z)}{\rho(x_0)} f(z) \right) \right] && (20)\\ &= \mathbb{E}_{x_0} \left[\sum_x (p(x \mid x_0) g(x)) \sum_z (p(z \mid x_0) f(z)) \right] \quad \text{[Using Bayes' rule]} && (21)\\ &= \mathbb{E}_{x_0} [T_M f(x_0) T_M g(x_0)] = (T_M f, T_M g)_{\rho_X} = B^{VAR}(f,g) && (22) \end{aligned}$$
□

For implementations, it is more natural to consider *invariance* to data augmentations.

Theorem B.3 (equivalent eigenfunctions)**.** *Assume that T_M is a compact operator. Define the invariance bilinear form*

$$B^{INV}(f,g) = (T_M f - f, T_M g - g) \tag{23}$$

Then B^{INV}, B^{VAR} share the same set of eigenfunctions. Moreover, these are the same as the eigenfunctions of $B^{DA,B}$. In particular, for any eigenfunction f_j of B^{VAR}, with eigenvalue λ_j, then f_j is also and eigenfunction of B^{INV}, with the corresponding eigenvalue given by $(\sqrt{\lambda_j}-1)^2$.

Proof. Define T_{MM} by,

$$T_{MM} f = T_M^\top T_M f \tag{24}$$

Define

$$T_{MS} = (T_M - I)^\top (T_M - I) \tag{25}$$

Note, by the assumption of compactness, T_M has the Singular Value Decomposition, (see the Hilbert Space section for equation SVD),

$$T_M(h) = \sum_{j=1}^{\infty} \lambda_j (h, g_j) f_j \tag{SVD}$$

Let f_j be any right eigenvector of T_M, with eigenvalue μ_j. Then f_j is also a right eigenvector $T_M - I$, with eigenvalue $\mu_j - 1$. So we see that T_{MM} has f_j as an eigenvector, with eigenvalue $\lambda_j = \mu_j^2$ and T_{MS} has f_j as an eigenvector, with eigenvalue $(\sqrt{\lambda_j}-1)^2$. Finally, the fact that there are no other eigenfunctions also follows from equation SVD.

The final part follows from the previous lemma. □

Equivalence of Barlow Twins loss to Equation (9). The BarlowTwins loss from [44] is as follows:

$$\mathcal{L}_{BT} = \sum_i (C_{ii} - 1)^2 + \beta \sum_i \sum_{j \neq i} C_{ij}^2 \tag{26}$$

where C is the cross-correlation matrix computed between the outputs of the network to two different augmentations. First, the BarlowTwins loss can be seen as the unconstrained optimization form of the following constrained optimization objective:

$$\mathcal{L}_{BT} = \sum_i (C_{ii} - 1)^2 \quad \text{, subject to} \quad C_{ij} = 0 \quad \forall j \neq i \tag{27}$$

where β is the Lagrangian multiplier [9]. In [44], the cross-correlation matrix C is computed by a dot product between normalized functions f_i's such that $(f_i, f_i)_{\rho_X} = 1\ \forall i$. The network output for one augmentation of x, a, can be thought of as a Monte-Carlo estimate (with one sample) of $T_M f_i(x)$, where f_i is the i^{th} dimension of the network's output. Therefore, the BarlowTwins loss can be written in its following equivalent form:

$$\hat{L}^{BT}(F) = \sum_{i=1}^{N_k} \left((T_M f_i, T_M f_i)_{\rho_X} - 1\right)^2 \quad \text{, subject to} \quad (f_i, f_j)_{\rho_X} = \delta_{ij} \tag{28}$$

As shown by [45], the eigenvalues of $T_M^T T_M$ are always less than 1. Therefore, we do not need the square in Equation (28). Rewriting it, we get the following:

$$\hat{L}^{BT}(F) = \sum_{i=1}^{N_k} (T_M f_i, T_M f_i)_{\rho_X} \quad \text{, subject to} \quad (f_i, f_j)_{\rho_X} = \delta_{ij} \tag{29}$$

Using Theorem B.3, we show that the loss recovers the equivalent eigenfunctions for the following reason. We can rewrite the loss as

$$\begin{aligned} \hat{L}^{BT}(F) &= \sum_{i=1}^{N_k} ((T_M - I) f_i, (T_M - I) f_i)_{\rho_X} \quad \text{, subject to} \quad (f_i, f_j)_{\rho_X} = \delta_{ij} \\ \implies \hat{L}^{BT}(F) &= \sum_{i=1}^{N_k} \|T_M f_i - f_i\|^2_{L^2(\rho_X)} \quad \text{, subject to} \quad (f_i, f_j)_{\rho_X} = \delta_{ij} \end{aligned} \tag{30}$$

which recovers the loss Equation (9). Note that the VICReg loss [5], in addition to the constraints imposed by the BarlowTwins loss, ensures that the norm of f_i's are more than some threshold. This can be easily incorporated into the constraint with a constant along with δ_{ij}. In conclusion, both BarlowTwins and VICReg losses can be seen as equivalent forms of the loss Equation (9).

Theorem B.4. *(Informal) Let us denote the span of the feature space at initialization as $V(F_0)$ and after training as $V(F_T)$. For small initialization of the network's weights, the alignment of $V(F_T)$ with the eigenfunctions of $\mathcal{T}$ depend on two factors: (i) alignment of $V(F_0)$ with the eigenfunctions of $\mathcal{T}$; (ii) singular values of $\mathcal{T}$.*

Theorem B.4. (Formal) *Let $\Gamma = V\Lambda V^T$ represent the eigendecomposition of Γ, and define z as the projection of the weight vectors in W onto singular vectors of Γ, V. Formally, $z = WV$. Assuming small initialization (as in Simon et al. (2023), i.e. $|z_{pi}(0)| << 1$ for all p, i, we can derive the following conclusions:*

1. *$sign(\frac{\Delta z_{pi}(t)}{z_{pi}(t)}) = sign(\lambda_i)$*

2. *For all $\lambda_i, \lambda_j > 0$, $\frac{z_{pi}(t)}{z_{pi}(0)} = (\frac{z_{pj}(t)}{z_{pj}(0)})^{\frac{\lambda_i}{\lambda_j}}$ where λ_i denotes the i^{th} singular value, i.e. i^{th} element of diagonal matrix Λ.*

Proof. We will first show that the above holds for a linear network, i.e. the output of the network with weights $W \in \mathbf{R}^{m\times n}$ is WX for some input $X \in \mathbf{R}^{n\times b}$, where m is the output dimensionality, n is the input dimensionality and b is the batch size.
Let us first analytically compute the cross-correlation matrix C following [37].

$$C = WXX'^T W^T = W\mathcal{T}W^T$$
$$C_{pq} = \sum_{i,j} W_{pi}\mathcal{T}_{ij}W_{qj} \quad , \quad C_{pp} = \sum_{i,j} W_{pi}\mathcal{T}_{ij}W_{pj}$$

where X and X' are matrices $\in \mathbf{R}^{n\times b}$ containing two augmentations of a each image in a batch of images. Also, we have defined $\mathcal{T} = XX'^T$, i.e. the augmentation-defined data correlation matrix. Rewriting the BarlowTwins loss function from [44]:

$$\mathcal{L}_{BT} = \sum_i (C_{ii} - 1)^2 + \beta \sum_i \sum_{j\neq i} C_{ij}^2$$

To study the learning dynamics, we need to compute the gradient of $\mathcal{L}_{BT}$ w.r.t. the parameters W.

$$\frac{dW_{pq}}{dt} = -\eta \frac{\partial \mathcal{L}_{BT}}{\partial W_{pq}} = -2\eta \sum_i (C_{ii} - 1)\frac{\partial C_{ii}}{\partial W_{pq}} - 2\eta\beta \sum_i \sum_{j\neq i} C_{ij}\frac{\partial C_{ij}}{\partial W_{pq}} \tag{31}$$

Let us now analytically compute the derivatives of C_{ii} and C_{ij} w.r.t W_{pq} to simplify each of the terms in Equation (31).

$$\begin{aligned}
\frac{\partial C_{ii}}{\partial W_{pq}} &= \frac{\partial}{\partial W_{pq}} \sum_{j,k} W_{ij}\mathcal{T}_{jk}W_{ik} = \frac{\partial}{\partial W_{pq}} \sum_{j,k} W_{ij}\mathcal{T}_{jk}W_{ik}\delta_{pi} \\
&= \left(\sum_{j,k} \mathcal{T}_{jk}W_{pk}\delta_{jq} + \sum_{j,k} W_{pj}\mathcal{T}_{jk}\delta_{kq} \right)\delta_{pi} \\
&= \left(\sum_{k} \mathcal{T}_{qk}W_{pk} + \sum_{j} W_{pj}\mathcal{T}_{jq} \right)\delta_{pi} \\
&= 2\,[W\mathcal{T}]_{pq}\,\delta_{pi}
\end{aligned}$$

$$\implies \sum_i (C_{ii} - 1)\frac{\partial C_{ii}}{\partial W_{pq}} = 2(C_{pp} - 1)\,[W\mathcal{T}]_{pq} \tag{32}$$

Using similar algebra steps, we can simplify the second term:

$$\begin{aligned}
\frac{\partial C_{ii}}{\partial W_{pq}} &= [W\mathcal{T}]_{jq}\,\delta_{pi} + [W\mathcal{T}]_{iq}\,\delta_{pj} \\
\implies \sum_i \sum_{j\neq i} C_{ij}\frac{\partial C_{ii}}{\partial W_{pq}} &= \sum_i \sum_{j\neq i} C_{ij}\left([W\mathcal{T}]_{jq}\,\delta_{pi} + [W\mathcal{T}]_{iq}\,\delta_{pj}\right) \\
&= \sum_{j\neq q} C_{pj}\,[W\mathcal{T}]_{jq} + \sum_{i\neq q} C_{ip}\,[W\mathcal{T}]_{iq} \\
&= 2\,[(C-I)W\mathcal{T}]_{pq} - 2(C_{pp}-1)\,[W\mathcal{T}]_{pq} \qquad (33)
\end{aligned}$$

Substituting Equations (32) and (33) into Equation (31), we get:

$$\begin{aligned}
\frac{dW_{pq}}{dt} = -\eta\frac{\partial \mathcal{L}_{BT}}{\partial W_{pq}} &= -4\eta(C_{pp}-1)\,[W\mathcal{T}]_{pq} - 4\eta\beta\,[(C-I)W\mathcal{T}]_{pq} + 4\eta\beta(C_{pp}-1)\,[W\mathcal{T}]_{pq} \\
&= -4\eta(1-\beta)(C_{pp}-1)\,[W\mathcal{T}]_{pq} - 4\eta\beta\,[(C-I)W\mathcal{T}]_{pq} \qquad (34)
\end{aligned}$$

Note that setting $\beta = 1$ yields the dynamics equation presented by [37]. However, in practice, β is orders of magnitude less that 1. For sake of simplicity, we will analyze the extreme case of $\beta = 0$, which will yield us insights into the weak-orthogonality constraint case. Therefore,

$$\frac{dW_{pq}}{dt} \approx -4\eta(C_{pp}-1)\,[W\mathcal{T}]_{pq} \qquad (35)$$

Let us denote the eigendecomposition of $\mathcal{T}$ be written as $\mathcal{T} = V\Lambda V^T$. Here, Λ is a diagonal matrix with singular values as the diagonal elements. Let us also denote the projection of the weight vectors onto the singular vectors of $\mathcal{T}$, i.e. V as z. So, $z = WV$.
Therefore, using these definitions, we can write the following:

$$\begin{aligned}
C_{pp} &= \left[W\mathcal{T}W^T\right]_{pp} = \left[Z\Lambda Z^T\right]_{pp} = \sum_i z_{pi}^2\lambda_i \\
W\mathcal{T} &= WV\Lambda V^T = Z\Lambda V^T
\end{aligned}$$

Now, writing the update equations Equation (35) in terms of z_{pi}:

$$\begin{aligned}
\frac{dz_{pi}}{dt} &= \sum_q \frac{dW_{pq}}{dt}V_{qi} \\
&= -4\eta\left(\sum_j z_{pj}^2\lambda_j - 1\right)\sum_k z_{pk}\lambda_k(\sum_q V_{qk}V_{qi}) \\
&= -4\eta\left(\sum_j z_{pj}^2\lambda_j - 1\right) z_{pi}\lambda_i \qquad (36)
\end{aligned}$$

Assuming small initialization of weights W, we can assume that $\mid z_{pi}(0) \mid << 1$, i.e. magnitude z_{pi} at time 0 is very small.
Let us define $h_p(t) = 1 - \sum_j z_{pj}(t)^2\lambda_j$. For small initialization, $h_p(t) > 0\ \forall t$. Therefore,

$$sign\left(\frac{dz_{pi}(t)}{dt}\frac{1}{z_{pi}}\right) = sign(\lambda_i) \qquad (37)$$

It is clear from Equation (37) that if $\lambda_i < 0$, $\lim_{t\to\infty} z_{pi}(t) = 0$. Similarly, if $\lambda_i = 0$, then $z_{pi}(t) = z_{pi}(0)\ \forall t$.
Therefore, akin to the conclusions of [37], the BarlowTwins loss recovers directions corresponding to positive singular values in the augmentation-defined covariance matrix, $\mathcal{T}$ and suppresses directions corresponding to negative singular values. Thus, the network outputs span the top singular vectors of $\mathcal{T}$.

It is worth noting from Equation (36) that the following holds:

$$
\begin{aligned}
\frac{1}{\lambda_i}\frac{dlog(z_{pi})}{dt} &= \frac{1}{\lambda_j}\frac{dlog(z_{pj})}{dt} \\
\implies \frac{1}{\lambda_i}log\left(\frac{z_{pi}(t)}{z_{pi}(0)}\right) &= \frac{1}{\lambda_j}log\left(\frac{z_{pj}(t)}{z_{pj}(0)}\right) \\
\implies \frac{z_{pi}(t)}{z_{pi}(0)} &= \left(\frac{z_{pj}(t)}{z_{pj}(0)}\right)^{\frac{\lambda_i}{\lambda_j}}
\end{aligned}
\tag{38}
$$

□

Without loss of generality, if $\lambda_i << \lambda_j$, then $z_{pi}(t) \approx z_{pi}(0)$. Therefore, under small initialization, i.e. $z_{pi}(0)$ is small $\forall i$, gradient descent biases the p^{th} weight vector to be more strongly aligned to the eigenvector corresponding to the strongest eigenvalue, for all p's. Hence, under weak orthogonalization constraints, the BarlowTwins loss will over "represent" the strong singular vectors of the augmentation-defined cross-correlation matrix.
When using high-dimensional projectors, specifically when $m >> \sum_i \mathbf{1}_{\lambda_i > 0}$, wherein $\mathbf{1}_\zeta$ is the indicator function that is 1 when condition ζ is true and 0 otherwise, this problem might be ameliorated because there are multiple weight vectors that might be aligned with the top singular vectors of $\mathcal{T}$ at initialization. However, when using low-dimensional projectors, we do not have such a luxury and therefore, using a weak orthogonalization constraint leads to dimensionality collapse in the representation space.

Extending to deep non-linear networks. Similar to the analysis in [37], we can repeat the above analysis by replace X and X' by the corresponding kernel versions, where the kernel corresponds to the Neural Tangent Kernel (NTK) of the network. Therefore, the implicit bias of gradient descent to yield dimensionality collapse in the representation space when using weak orthogonalization constraints still remains.

Dimensionality collapse under noisy optimization. From the rest of this section, we have seen that the BarlowTwins loss is a Monte-Carlo estimate of the true data-augmentation defined covariance matrix. Moreover, stochastic gradient descent adds noise due to mini-batch sampling to the optimization process. Note that there exist symmetries in our linear network, i.e. an orthogonal rotation of the weight matrix yields the same loss function. As explained in [12], such symmetry-invariant sets are potential candidates for stochastic collapse when performing noisy gradient-based optimization. Therefore, the presence of noise in the data-augmentation covariance matrix, $\mathcal{T}$, as well as the batch noise can further worsen the dimensionality collapse problem where different weight vectors become parallel to each other due to noise in updates. One possible mitigation strategy is to obtain a better estimate of the true augmentation-defined covariance matrix (see Figure 7), which we discuss in the next section.

Empirical validation. We empirically validate our results on the learning dynamics on simplistic 2-dimensional settings. These results, demonstrating the difference in feature learning dynamics for weak vs strong orthogonalization, are presented as GIFs in the supplementary material, and can also be viewed at the project website.

C Multi-Augmentation Learning

C.1 Augmentation graph

We use the population augmentation graph formulation introduced in [23]. Briefly, we define a graph $\mathcal{G}(\mathcal{X}, \mathcal{W})$, where the vertex set $\mathcal{X}$ comprises of all augmentations from the dataset (could be infinite when continuous augmentation functions are used) and $\mathcal{W}$ denotes the adjacency matrix with edge weights as defined below:

$$w_{xx'} := \mathbb{E}_{\bar{x}\sim\mathcal{P}_{\bar{X}}}\left[\mathcal{A}(x|\bar{x})\mathcal{A}(x'|\bar{x})\right] \tag{39}$$

, i.e. the joint probability of generating 'patches' x, x' from the same image $\bar{x}$. Here $\mathcal{A}$ defines the set of augmentation functions used in the SSL pipeline. It is worth noting that the magnitude of $w_{xx'}$ captures the relative similarity between x and x'. A higher value of $w_{xx'}$ indicates that it is more

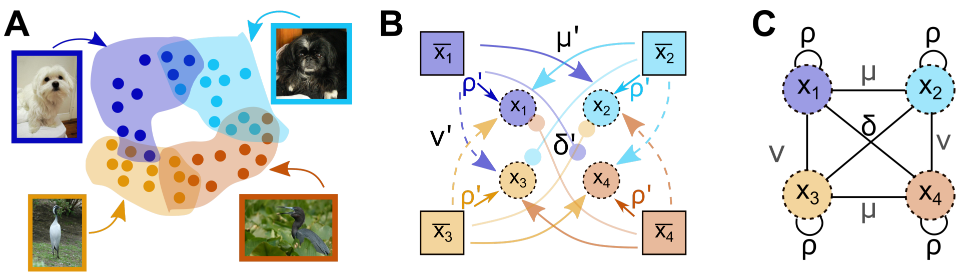


Figure 6: Schematic of augmentation graph. (A) Augmentations from each image span a region in the image space which could overlap with the augmentation span of other images. (B) An augmentation graph schematic that uses probabilities to characterize the interactions among augmentation spans of different instances.

likely that both patches came from the same image, and thereby are more similar. The marginal likelihood of each patch x can also be derived from this formulation:

$$w_x = \mathbb{E}_{x' \sim \mathcal{X}} \left[w_{xx'} \right] \tag{40}$$

C.2 Contrastive and non-contrastive losses suffer from the same issues

We will now show that the proposal of using multiple patches for the $\mathcal{L}_{invariance}$ is pertinent to both the contrastive and non-contrastive SSL. Following [23], we use the spectral contrastive loss formulation and incorporate the augmentation graph relations:

$$\begin{aligned} \mathcal{L}_c &= -\mathbb{E}_{x,x^+} \left[f(x)^T f(x^+) \right] + \beta \mathbb{E}_{x,x'} \left[\left(f(x)^T f(x') \right)^2 \right] \\ \mathcal{L}_c &\propto \| ZZ^T - D^{-\frac{1}{2}} \mathcal{W} D^{-\frac{1}{2}} \|_F^2 = \| ZZ^T - \bar{\mathcal{W}} \|_F^2 \end{aligned} \tag{41}$$

where $z := \sqrt{w_x} f(x)$, D is a $N \times N$ diagonal matrix with entries $\{w_x\}$ and $\bar{\mathcal{W}} = D^{-\frac{1}{2}} \mathcal{W} D^{-\frac{1}{2}}$.

We extend the duality results between contrastive and non-contrastive SSL loss, established by [20], to demonstrate how Equation (41) can be decomposed into the invariance and collapse-preventing loss terms.

$$\| ZZ^T - \bar{\mathcal{W}} \|_F^2 = \| Z^T Z - I_d \|_F^2 + 2Tr \left[Z^T (I_N - \bar{\mathcal{W}}) Z \right] + \kappa \tag{42}$$

$$= \| Z^T Z - I_d \|_F^2 + 2 \sum_i \sum_x (1 - \bar{w}_x) z_i^2 - 2 \sum_i \sum_{x,x'} \bar{w}_{xx'} z_i z_i' + \kappa \tag{43}$$

where κ is some constant independent of Z. The first term in Equation (42) is the covariance regularization term in non-contrastive losses like BarlowTwins (implicit) or VIC-Reg (explicit), and the second term in Equation (43) is the variance regularization. Simplifying the third term in Equation (43) gives us:

$$\begin{aligned} \sum_i \sum_{x,x'} \bar{w}_{xx'} z_i z_i' &= \sum_i \sum_{x,x'} w_{xx'} f(x)_i f(x')_i = \sum_i \sum_{x,x'} \mathbb{E}_{\bar{x} \sim \mathcal{P}_{\bar{X}}} \left[\mathcal{A}(x|\bar{x}) \mathcal{A}(x'|\bar{x}) f(x)_i f(x')_i \right] \\ &= \sum_i \mathbb{E}_{\bar{x} \sim \mathcal{P}_{\bar{X}}} \left[\sum_x \mathcal{A}(x|\bar{x}) (f(x)_i \overline{f(x)}_i - f(x)_i^2) \right] \\ &= \mathbb{E}_{\bar{x} \sim \mathcal{P}_{\bar{X}}} \left[\sum_x \mathcal{A}(x|\bar{x}) \left(f(x)^T \overline{f(x)} - \| f(x) \|^2 \right) \right] \end{aligned} \tag{44}$$

This term encourages $f(x)$ to be similar to $\overline{f(x)}$, i.e. the mean representation across all augmentations of $\bar{x}$, thereby requiring to "sufficiently" sample $A(.|\bar{x})$. Given that both the contrastive and non-contrastive losses rely on learning invariance properties from data augmentations, we believe that our multi-patch proposal would improve the probability density estimation of $A(.|\bar{x})$ and yield better performance with few training epochs.

C.3 Explaining training dynamics in low patch sampling regime

We now turn to a simple form of the augmentation graph to understand how using low number of augmentations affects the evolution of ZZ^T. Minimizing Equation (41) implies that the spectral decomposition of Z would align with the top eigenvectors (and values) of $\overline{\mathcal{W}}$. We will demonstrate that in the low sampling regime (using few augmentations), the eigenvectors of the sampled augmentation graph $\tilde{\mathcal{W}}$ *may not* align with those of $\overline{\mathcal{W}}$.

Augmentation graph setup. We define an augmentation graph with only two instances from two different classes, similar to the one presented in [36]. Let us denote the four instances as $\bar{x}_i$ for $i \in 1, 2, 3, 4$, where $\bar{x}_1, \bar{x}_2$ belong to class 1 (i.e. $y_1, y_2 = 1$) and $\bar{x}_3, \bar{x}_4$ belong to class 2 (i.e. $y_3, y_4 = 4$). Let us further assume that $\bar{x}_1, \bar{x}_3$ have the highest pixel-level similarity among $(\bar{x}_1, \bar{x}_i) \forall i \in 2, 3, 4$, thereby making it more likely to have similar patches. We denote this relationship among input examples using $\mathcal{G}$ to indicate (pixel-wise) global similarity groups. So, $\mathcal{G}_1, \mathcal{G}_3 = 1$ and $\mathcal{G}_2, \mathcal{G}_4 = 2$. We can use the following probabilistic formulation to model our augmentation functions (see Figure 6B):

$$A(x_j|\bar{x}_i) = \begin{cases} \rho' & \text{if } j = i \\ \mu' & \text{if } j \neq i \text{ and } y_j = y_i \text{ and } \mathcal{G}_j \neq \mathcal{G}_i \\ \nu' & \text{if } j \neq i \text{ and } y_j \neq y_i \text{ and } \mathcal{G}_j = \mathcal{G}_i \\ \delta' & \text{if } j \neq i \text{ and } y_j \neq y_i \text{ and } \mathcal{G}_j \neq \mathcal{G}_i \end{cases} \tag{45}$$

In our setting, $\rho' + \mu' + \nu' + \delta' = 1$. The adjacency matrix of our augmentation graph (as shown in Figure 6C) is as follows:

$$\overline{\mathcal{W}} = \begin{bmatrix} \rho & \mu & \nu & \delta \\ \mu & \rho & \delta & \nu \\ \nu & \delta & \rho & \mu \\ \delta & \nu & \mu & \rho \end{bmatrix} \tag{46}$$

We defer the relations between $\rho', \mu', \nu'\delta'$ and ρ, μ, ν, δ to the appendix. The eigenvalues of this matrix are: $(\rho + \mu + \nu + \delta,\ \rho + \mu - \nu - \delta, \rho - \mu + \nu - \delta,\ \rho - \mu - \nu + \delta)$. Corresponding eigenvectors are along $[1, 1, 1, 1]^T$, $[1, 1, -1, -1]^T$. $[1, -1, 1, -1]^T$, $[1, -1, -1, 1]^T$. Assuming that the augmentation functions induce semantically-relevant invariance properties that are relevant for identifying y_i from $f(x_i)$, we can say that $\rho' > max\{\mu', \nu'\}$ and $min\{\nu', \mu'\} > \delta'$. When we have sufficiently sampled the augmentations, any SSL loss will learn Z such that its singular values are span the top eigenvectors of the augmentation graph, and the eigenspectrum of ZZ^T would simply be the above eigenvalues. In practical settings, the augmentation graph would have significantly higher dimension that the feature/embedding dimension [2]. Therefore, singular vectors of Z would span the top eigenvectors of $\overline{\mathcal{W}}$ and the smaller eigenmodes are not learned. When we have accurately sampled the augmentation graph, $\mu > \nu$ and therefore, the class-information preserving information is preferred over pixel-level preserving information during learning. But what happens when we *do not sufficiently sample the augmentation space?*

Ansatz. Based on our empirical experience, we define *an ansatz* pertaining to the eigenvalues of a sampled augmentation graph and validate it in tractable toy settings, such as the one described above. Specifically, we claim that when the augmentation space is not sufficiently sampled, $\{|\mu - \nu|, \delta\} \to 0$. In other words, we claim that when only few augmentations per example are used, it is more likely to have an equal empirical likelihood for augmentations that preserve (pixel-level) global information and class/context information. Moreover, it is very unlikely to have augmentations that change both the class and global information. This is demonstrated in Figure 7.

Consequences of the *Ansatz*. When only a few augmentations are sampled, learning can suppress the class information at the cost of preserving the pixel-level information, thereby leading to an increased smoothness in the learned feature space.

[2]Contrastive algorithms use a large batch size, thereby optimizing a high-dimensional ZZ^T whereas non-contrastive algorithms use a large embedding dimension, thereby optimizing a high-dimensional $Z^T Z$.

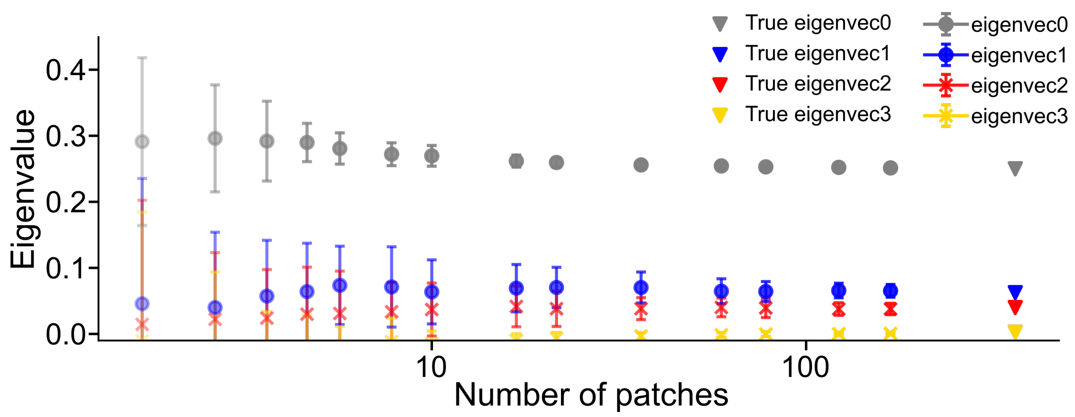


Figure 7: Empirical verification of the subsampling Ansatz.

D Implementation Details

Image Classification Datasets Across all experiments, our settings mainly follow [14]. In particular, Table 4a summarizes our pretraining settings on Cifar-10 [27], STL-10 [15] and Imagenet-100 [34]. The Imagenet-100 dataset was generated by sampling 100 classes from the original Imagenet-1k dataset, according to this list [39]. In Table 4b, we outline the corresponding linear evaluation settings for Resnet-50 (for CIFAR-10 and STL-10) and ResNet-18 (for Imagenet). Note that we add a linear classifier layer to the encoder's features and discard the projection layers for evaluation. Our code base is publicly available on github.

config	value
optimizer	Adam
learning rate	1e-3
batch size	128 (Imagnet), 256 (CIFAR, STL)
epochs	100
weight-decay	1e-6

(a) Pretraining

config	value
optimizer	Adam
learning rate	1e-3
batch size	512
epochs	200
weight-decay	1e-6
test-patches	16

(b) Linear Evaluation

Table 4: Experiment Protocol for comparing SSL algorithms

The key SSL loss functions that we use in this work are BarlowTwins [44] and VICReg [5]. Let us suppose that the embeddings of two augmentations of a batch of images are denoted as z and z'. The BarlowTwins loss function is as follows:

$$\mathcal{L}_{BT} = \sum_i (C_{ii} - 1)^2 + \beta \sum_i \sum_{j \neq i} C_{ij}^2 \tag{47}$$

$$\text{where} \quad C = \frac{1}{n-1} \sum_{k=1}^{n} (z_k - \bar{z})(z'_k - \bar{z'})^T$$

$$\text{and} \quad \bar{z} = \frac{1}{n} \sum_{k=1}^{n} z_k \quad , \quad \bar{z'} = \frac{1}{n} \sum_{k=1}^{n} z'_k \tag{48}$$

C_{ij} is the element of C at row i, column j and n is the batch size. For each projector dimensionality, d, we search for the hyperparameter, β, that yields the best downstream task performance.

The VICReg loss function is as follows:

$$\mathcal{L}_{VIC} = \frac{1}{n}\mu \sum_{k=1}^{n} \|z_k - z'_k\|^2 + \frac{1}{2}\mu\left[v(Z) + v(Z')\right] + \frac{1}{2}\left[c(Z) + c(Z')\right] \tag{49}$$

$$\text{where} \quad v(Z) = \frac{1}{d}\sum_{i=1}^{d} max(0, 1 - Stdev(z_{:,i}))$$

$$\text{and} \quad c(Z) = \frac{1}{d}\sum_{i}\sum_{j\neq i}\left[C(Z)_{ij}\right]^2 \quad , \quad C(Z) = \frac{1}{n-1}\sum_{k=1}^{n}(z_k - \bar{z})(z_k - \bar{z})^T$$

For each projector dimensionality, d, we search for the hyperparameter, μ, that yields the best downstream task performance.

D.1 Empirical results for low-dimensional projectors

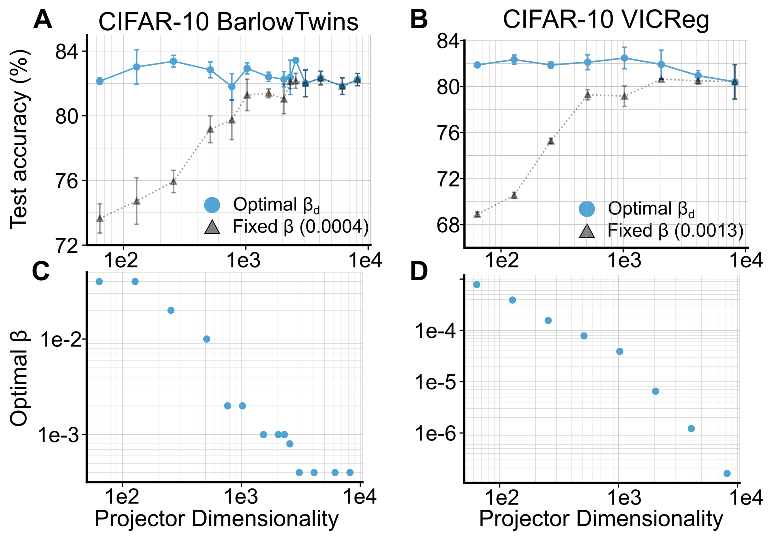


Figure 8: Low-dimensional projectors can yield good representations for both BarlowTwins and VICReg. We demonstrate that using a higher orthogonality constraint, β, for lower projector dimensionality can achieve similar performance over a wide range of projector dimensions (d). Note that for VICReg, we plot the ratio of the coefficient of the covariance loss to the coefficient of the invariance loss, i.e. $\beta = \frac{1}{d*\mu}$, where μ is the coefficient of the invariance loss. (See Equation (49) for details of the loss formulation.)

pdim	*Projector params (approx)*	**Barlow Twins**		**VICReg**	
		fixed β	optimal β^*	fixed β	optimal β^*
64	135k	73.6 ± 0.9	82.1 ± 0.2	68.9 ± 0.2	81.9 ± 0.1
128	278k	74.7 ± 1.4	83.0 ± 1.1	70.6 ± 0.3	82.3 ± 0.4
256	589k	75.9 ± 0.7	83.4 ± 0.4	75.3 ± 0.2	81.9 ± 0.3
512	1.3M	79.2 ± 0.8	82.8 ± 0.5	79.3 ± 0.4	82.1 ± 0.6
1024	3.1M	81.3 ± 1.0	82.9 ± 0.3	79.2 ± 0.9	82.5 ± 0.9
2048	8.3M	81.0 ± 0.9	82.3 ± 0.5	80.6 ± 0.0	81.9 ± 1.2
4096	25.2M	82.3 ± 0.4	82.3 ± 0.4	80.5 ± 0.3	81.0 ± 0.4
8192	83.9M	82.2 ± 0.4	82.2 ± 0.4	80.4 ± 1.5	80.4 ± 1.5

Table 5: Extended version of Table 1. Optimizing for orthogonality appropriately allows low-dimensional projectors to match the performance for BarlowTwins and VICReg (on CIFAR-10) of much higher-dimensional projectors.

D.2 Empirical results with multi-augmentations along with Time

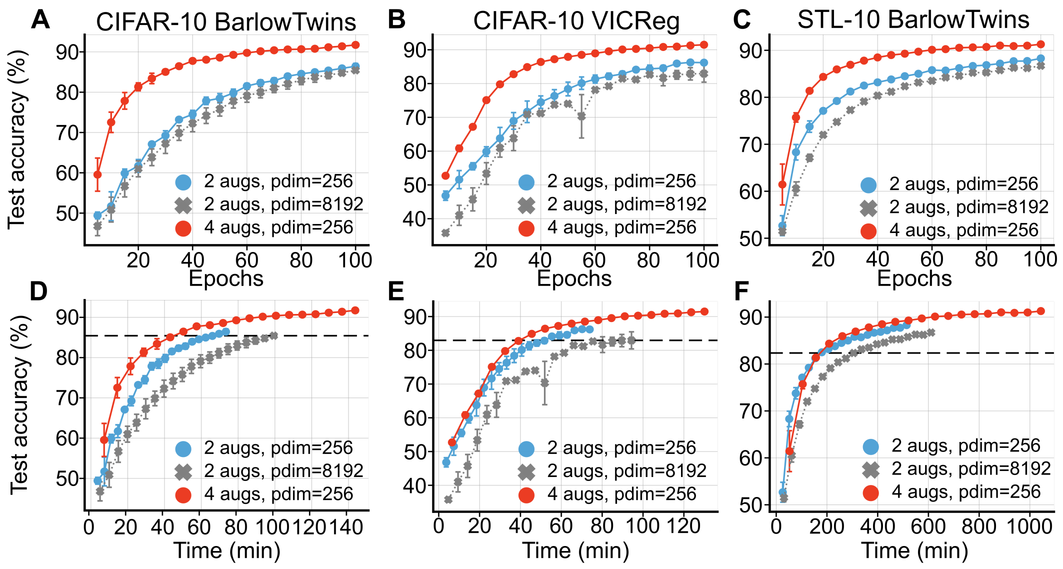


Figure 9: Using multiple augmentations improves representation learning performance and convergence. (A-C) Across BarlowTwins and VICReg for CIFAR-10 and STL-10 pretraining, using 4 augmentations instead of 2 helps improve performance. (D-F) Although the 4-augmentations take longer for each epoch, its performance still trumps the 2-augmentation version of the algorithm at the same wall clock time. Please see Appendix E.3 for more results.

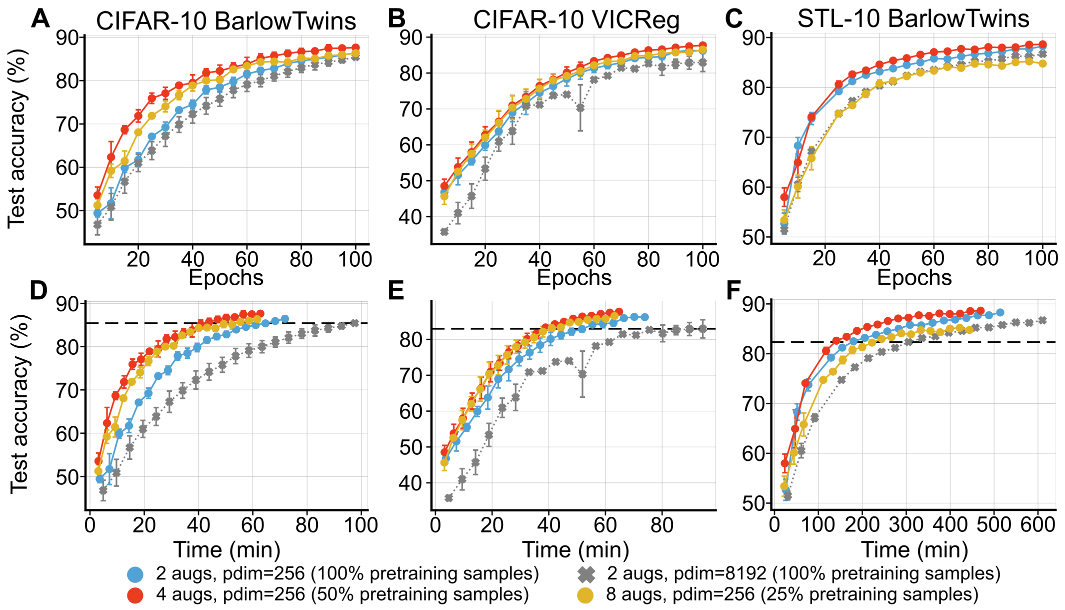


Figure 10: Multi-augmentation improves sample efficiency, recovering similar performance with significantly fewer unique samples in the pretraining dataset. Across BarlowTwins and VICReg pretraining on CIFAR-10 and STL-10, for the same effective dataset size ($\#augs \times \#unique\_samples$), using more patches improves performance at the same epoch (A-C) or wall clock time (D-F). However, a tradeoff exists wherein more data augmentations fail to improve performance in the scarce data regime.

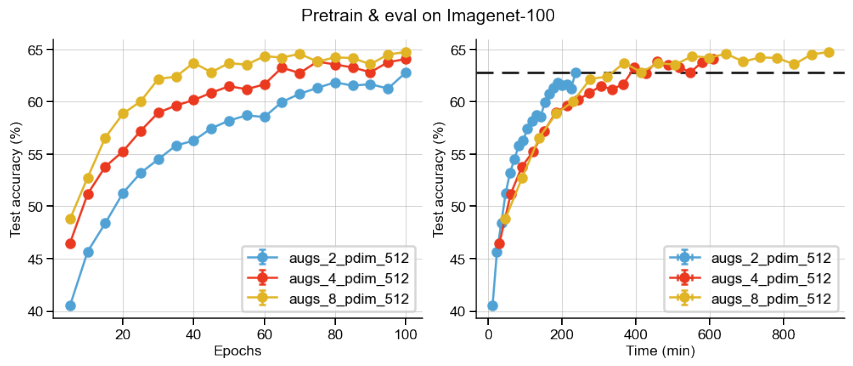


Figure 11: BarlowTwins pretraining on full Imagenet-100 dataset with 2, 4 and 8 augmentations.

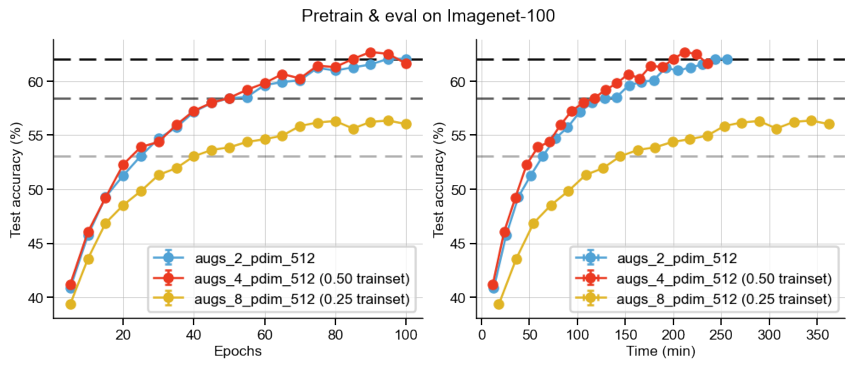


Figure 12: BarlowTwins pretraining on fraction of Imagenet-100 dataset with 2, 4 and 8 augmentations.

D.3 Empirical results on transfer learning

In this section, we present extended version of results presented in Figure 3, Figure 4 but pretraining on CIFAR-10 (or STL-10) and evaluating on STL-10 (or CIFAR-10). These results, coupled with the ones in Figure 3 Figure 4, present a strong case for the advantage of using the proposed multi-augmentation loss for better convergence as well as downstream accuracy.

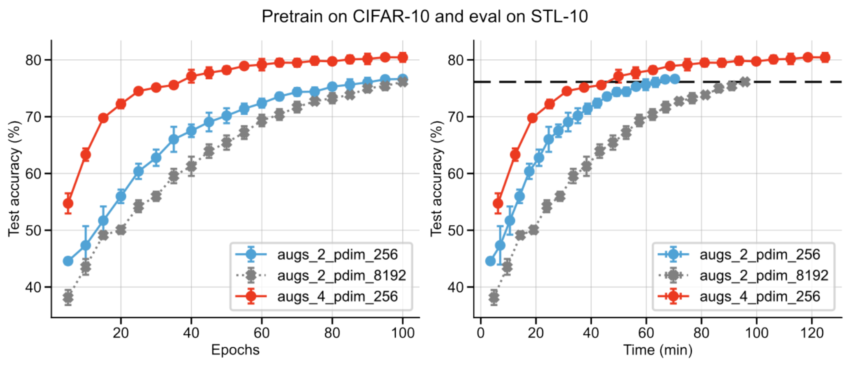


Figure 13: BarlowTwins pretraining on CIFAR-10, linear evaluation on STL-10 labelled set.

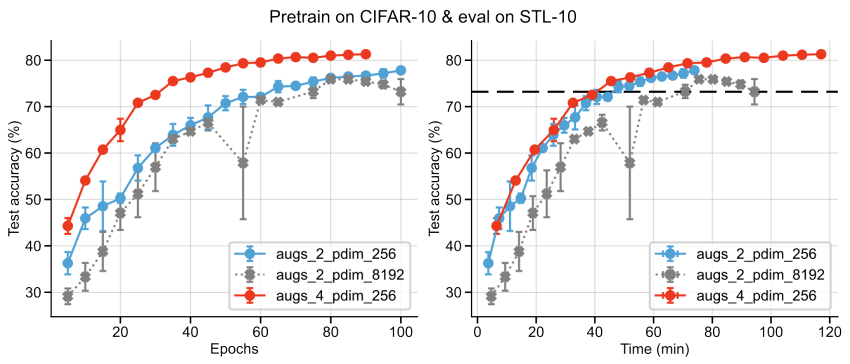


Figure 14: VICReg pretraining on CIFAR-10, linear evaluation on STL-10 labelled set.

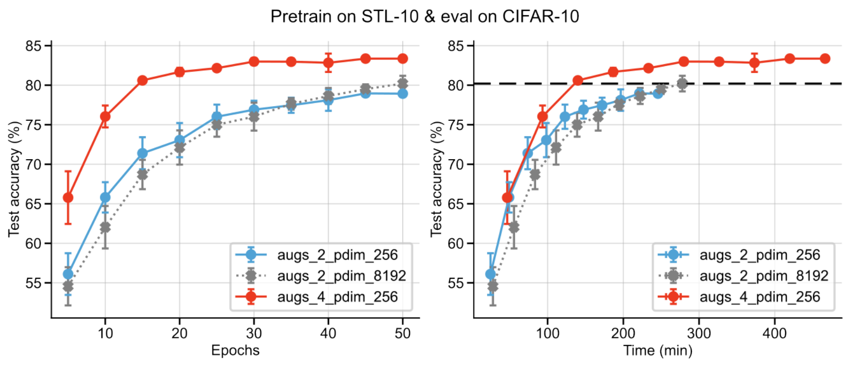


Figure 15: BarlowTwins pretraining on STL-10, linear evaluation on CIFAR-10 labelled set.

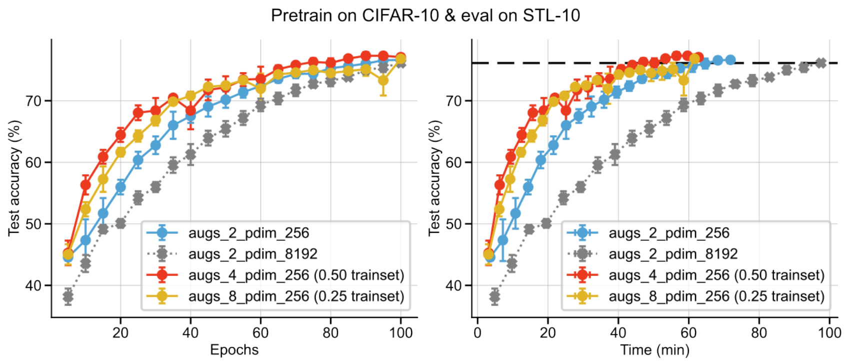


Figure 16: BarlowTwins pretraining on fraction of CIFAR-10 trainset, linear evaluation on STL-10 labelled set.

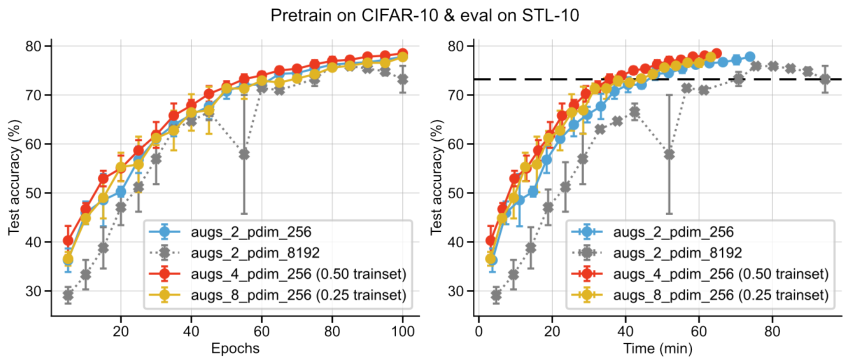


Figure 17: VICReg loss pretraining on fraction of CIFAR-10 trainset, linear evaluation on STL-10 labelled set.

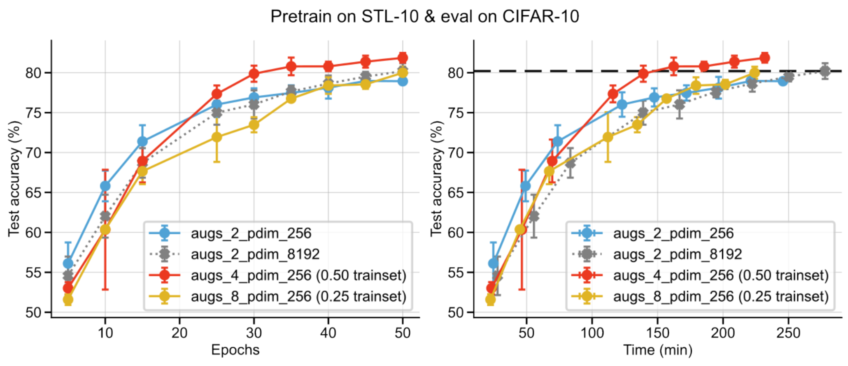


Figure 18: BarlowTwins loss pretraining on fraction of STL-10 unlabelled set, linear evaluation on CIFAR-10 train set.

E Additional Experiments probing multi-augmentation learning

E.1 Longer Pretraining to determine early stopping

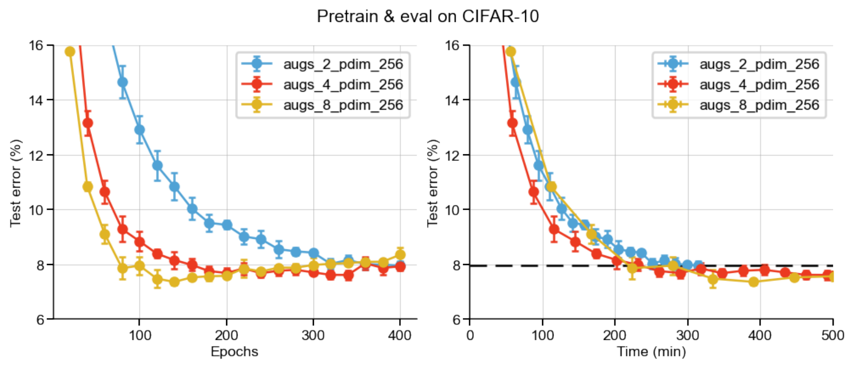


Figure 19: BarlowTwins pretraining on full CIFAR-10 dataset for 400 epochs.

Algorithm	Best accuracy	Best accuracy @ epoch
Barlow-Twins (2-augs) w/ pdim=256	92.04 +/- 0.16	400
Barlow-Twins (4-augs) w/ pdim=256	92.39 +/- 0.17	340
Barlow-Twins (8-augs) w/ pdim=256	92.64 +/- 0.10	140

Table 6: BarlowTwins pretraining on full CIFAR-10 dataset at 400 epochs (with early stopping)

E.2 SwAV-like augmentations for compute efficient multi-augmentation framework

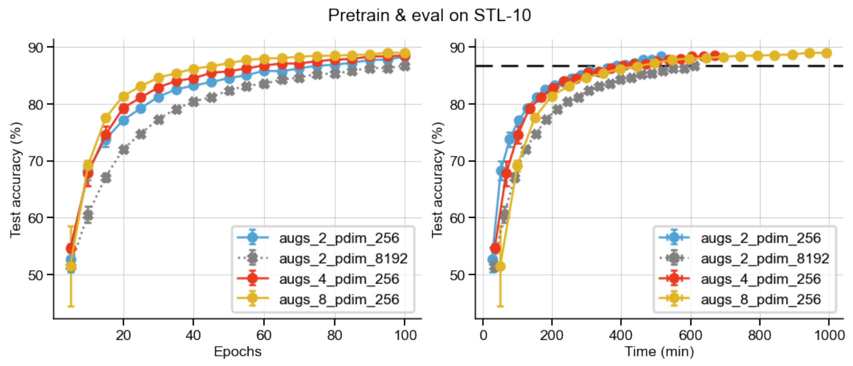


Figure 20: BarlowTwins pretraining on full STL-10 dataset for 100 epochs using SwAV-like augmentations. Specifically, the 2-augmentations setting uses two views that are 64×64, whereas the 4 (or 8) augmentation setting uses additional two (or six) augmentations that are 32×32.

E.3 Training with full dataset with 4/8 augmentations

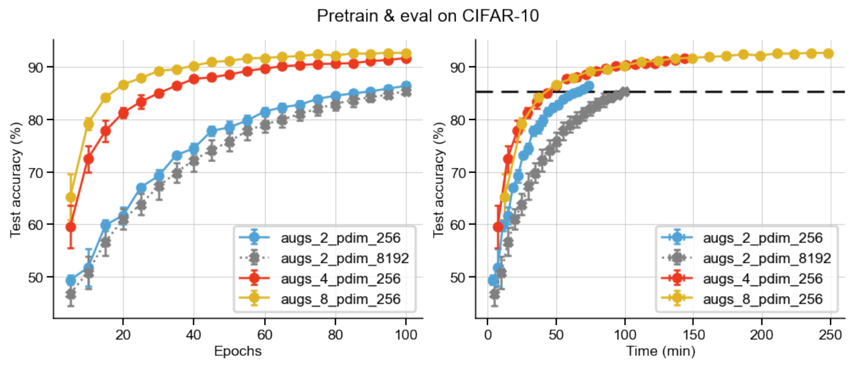


Figure 21: BarlowTwins pretraining on full CIFAR-10 dataset with 2, 4 and 8 augmentations.

Algorithm	**#augs=2**	**#augs=4**	**#augs=8**
Barlow-Twins w/ pdim=256	86.43 +/- 0.72	91.73 +/- 0.16	92.71 +/- 0.19
Barlow-Twins w/ pdim=8192	85.44 +/- 0.54	91.40 +/- 0.32	92.40 +/- 0.13

Table 7: BarlowTwins pretraining on full CIFAR-10 dataset at 100 epochs

Part IV

Learning with approximate gradients

7 Preface to Part IV

In Part IV, we focus on a different aspect of the learning rule pillar of the NeuroAI framework, specifically the credit assignment problem. Previously, I presented evidence for how tools from deep learning theory can be used to study the dynamics of feature learning and understand the impact of certain biologically-inspired mechanisms on the learning dynamics in ANNs trained using SSL. While this line of research can elucidate the computational advantages of employing certain neural mechanisms, it relies on using properties of gradient-based optimization. A common critique, often by neuroscientists, is that unlike ANNs, the brain cannot use gradients of the loss function computed using the backpropagation algorithm for learning. While it is true that the brain cannot use backpropagation as ANNs do, the brain may employ alternate strategies that might do something similar, i.e. compute some approximate gradient signal that can be used as the credit signal for synaptic weight updates. A natural question that arises at this juncture is *Can brains learn using some approximation to the gradient signal?* In Part IV, we study the impact of approximating the gradient signal on learning in a controlled setting. Specifically, we use ANNs as a model to study the impact of approximating the gradient signal on its learning and generalization performance, as well as different architectural factors that influence this impact.

Advances in neuroscience have focused on mechanistic or phenomenological models of plasticity in the brain by studying specific neurophysiological substrates underlying synaptic plasticity [Bredenberg and Savin, 2024]. In contrast, studies using ANNs have attempted to propose normative models of synaptic plasticity by developing biologically-plausible rules for learning complex behavior [Lillicrap et al., 2020]. While the aim of most normative models of synaptic plasticity has been to approximate the backpropagation algorithm as closely as possible using biologically-plausible mechanisms, it can be argued that phenomenological models of synaptic plasticity can also be framed as approximations of gradient of certain loss function [Richards and Kording, 2023]. In Part IV, we demonstrate that it is indeed possible to learn complex behavior in ANNs even when using approximation to the true gradient signal. We also study the architectural factors that might help mitigate the harmful effects of gradient approximation, thereby demonstrating that the problem of credit assignment should not be treated as independently from the system architecture. Overall, this chapter supports the utility of ANN studies using gradient-based optimization in elucidating potential mechanisms of learning in the brain.

* * *

This work is accepted to be published at ICLR 2023 and can be cited as Ghosh, A., Liu, Y.H., Lajoie, G., Kording, K. and Richards, B.A. How gradient estimator variance and bias impact learning in neural networks. ICLR 2023.

8

How gradient estimator variance and bias could impact learning in neural circuits

HOW GRADIENT ESTIMATOR VARIANCE AND BIAS COULD IMPACT LEARNING IN NEURAL CIRCUITS

Arna Ghosh
McGill University &
Mila-Quebec AI Institute
Montréal, QC, Canada
arna.ghosh@mail.mcgill.ca

Yuhan Helena Liu
University of Washington
Seattle, WA, USA
Mila-Quebec AI Institute
Montréal, QC, Canada

Guillaume Lajoie
Unversite de Montréal &
Mila-Quebec AI Institute
Montréal, QC, Canada

Konrad Körding
University of Pennsylvania
Philadelphia, PA, USA
CIFAR Learning in Machines & Brains
Toronto, ON, Canada

Blake A. Richards
McGill University, Mila-Quebec AI Institute &
Montreal Neurological Institute
Montréal, QC, Canada
CIFAR Learning in Machines & Brains
Toronto, ON, Canada
blake.richards@mcgill.ca

ABSTRACT

There is growing interest in understanding how real brains may approximate gradients and how gradients can be used to train neuromorphic chips. However, neither real brains nor neuromorphic chips can perfectly follow the loss gradient, so parameter updates would necessarily use gradient estimators that have some variance and/or bias. Therefore, there is a need to understand better how variance and bias in gradient estimators impact learning dependent on network and task properties. Here, we show that variance and bias can impair learning on the training data, but some degree of variance and bias in a gradient estimator can be beneficial for generalization. We find that the ideal amount of variance and bias in a gradient estimator are dependent on several properties of the network and task: the size and activity sparsity of the network, the norm of the gradient, and the curvature of the loss landscape. As such, whether considering biologically-plausible learning algorithms or algorithms for training neuromorphic chips, researchers can analyze these properties to determine whether their approximation to gradient descent will be effective for learning given their network and task properties.

1 INTRODUCTION

Artificial neural networks (ANNs) typically use gradient descent and its variants to update their parameters in order to optimize a loss function (LeCun et al., 2015; Rumelhart et al., 1986). Importantly, gradient descent works well, in part, because when making small updates to the parameters, the loss function's gradient is along the direction of greatest reduction.[1] Motivated by these facts, a longstanding question in computational neuroscience is, does the brain approximate gradient descent (Lillicrap et al., 2020; Whittington & Bogacz, 2019)? Over the last few years, many papers show that, in principle, the brain could approximate gradients of some loss function (Murray, 2019; Liu et al., 2021; Payeur et al., 2021; Lillicrap et al., 2016; Scellier & Bengio, 2017). Also inspired by the brain, neuromorphic computing has engineered unique materials and circuits that emulate biological networks in order to improve efficiency of computation (Roy et al., 2019; Li et al., 2018b). But, unlike ANNs, both real neural circuits and neuromorphic chips must rely on approximations to the true gradient. This is due to noise in biological synapses and memristors, non-differentiable operations such as spiking, and the requirement for weight updates that do not use non-local information (which can lead to bias) (Cramer Benjamin et al., 2022; M. Payvand et al., 2020b; Laborieux et al., 2021; Shimizu et al., 2021; Neftci et al., 2017; N. R. Shanbhag et al., 2019). Thus, both areas of research

[1]If we assume a Euclidean metric in weight space.

could benefit from a principled analysis of how learning is impacted by loss gradient variance and bias.

In this work, we ask how different amounts of noise and/or bias in estimates of the loss gradient affect learning performance. As shown in a simple example in Fig. 1, learning performance can be insensitive to some degree of variance and bias in the gradient estimate, and even benefit from it, but excessive amounts of variance and/or bias clearly hinder learning performance. Results from optimization theory shed light on why imperfectly following the gradient—e.g. via stochastic gradient descent (SGD) or other noisy GD settings—can improve generalization in ANNs (Foret et al., 2020; Chaudhari et al., 2019; Yao et al., 2018; Ghorbani et al., 2019). However, most of these results treat unbiased gradient estimators. In contrast, in this work, we are concerned with the specific case of weight updates with intrinsic but known variance and bias, as is often the case in computational neuroscience and neuromorphic engineering. Moreover, we also examine how variance and bias can *hinder* training, because the amount of variance and bias in biologically-plausible and neuromorphic learning algorithms is often at levels that impair, rather than improve, learning (Laborieux et al., 2021), and sits in a different regime than that typically considered in optimization theory.

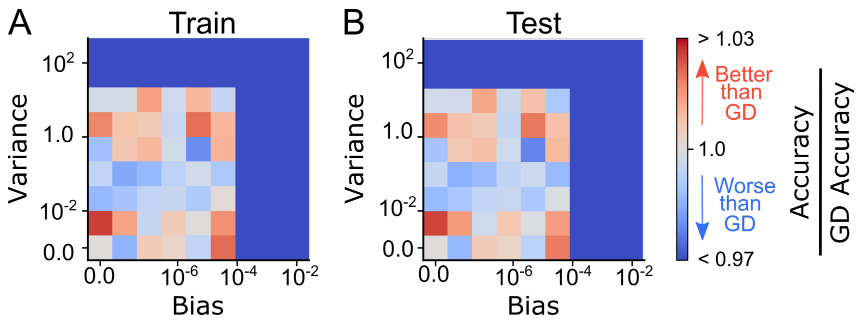


Figure 1: Train and test accuracy of a VGG-16 network trained for 50 epochs (to convergence) on CIFAR-10 using full-batch gradient descent (with no learning rate schedule) with varying amount of variance and bias (as a fraction of the gradient norm) added to the gradient estimates. These results (avg of 20 seeds) indicate that excessive noise and bias harms learning, but a small amount can aid it.

The observations in Fig. 1 give rise to an important question for computational neuroscientists and neuromorphic chip designers alike: what amount of variance and bias in a loss gradient estimate is tolerable, or even desirable? To answer this question, we first observe how variance and bias in gradient approximations impact the loss function in a single parameter update step on the training data. (We also extend this to multiple updates in Appendix A.) We utilize an analytical and empirical framework that is agnostic to the actual learning rule, and derive the factors that affect performance in an imperfect gradient setting. Specifically, we assume that each update is comprised of the contribution of the true gradient of the loss function with respect to the parameter, a fixed amount of bias, and some noise. Similar to Raman et al. (2019), we derive an expression for the change in the loss function after a discrete update step in parameter space: $w(t + \Delta t) = w(t) + \Delta w(t)\Delta t$, where Δt is akin to learning rate in standard gradient descent algorithms. We then characterize the impact on learning using the decrease in loss function under an approximate gradient setting as compared to the decrease in loss function when following the true gradient.

Our analysis demonstrates that the impact of variance and bias are independent of each other. Furthermore, we empirically validate our inferences in ANNs, both toy networks, and various VGG configurations (Vedaldi & Zisserman, 2016) trained on CIFAR-10 (Krizhevsky & Hinton, 2009). Our findings can be summarized as follows:

1. The impact of variance is second order in nature. It is lower for networks with a threshold non-linearity, as compared to parameter matched linear networks. Under specific conditions, variance matters lesser for wider and deeper networks.
2. The impact of bias increases linearly with the norm of the gradient of the loss function, and depends on the direction of the gradient as well as the eigenvectors of the loss Hessian.
3. Both variance and bias can help to prevent the system from converging to sharp minima, which may improve generalization performance.

Altogether, our results provide guidelines for what network and task properties computational neuroscientists and neuromorphic engineers need to consider when designing and using noisy and/or biased gradient estimators for learning.

2 Estimating the impact of variance and bias on training

Our analysis focuses on situations where the synaptic weights of a network, w, are trained to optimize a loss, $\mathcal{L}[\boldsymbol{w}]$. We use an auxiliary variable, t, to denote different points in this optimization trajectory and Δt to denote a single step in this trajectory. (We extend to the multi-step case in Corollary A.2 of the Appendix.) Compared to standard optimization protocols in the ANN literature, Δt is akin to the learning rate. Therefore, the loss as measured at one particular point in the trajectory is denoted as $\mathcal{L}[\boldsymbol{w}(t)]$, and the loss at the next point in the trajectory, i.e., after a weight update step, is denoted by $\mathcal{L}[\boldsymbol{w}(t+\Delta t)]$. In the rest of this work, we characterize $\mathcal{L}[\boldsymbol{w}(t+\Delta t)] - \mathcal{L}[\boldsymbol{w}(t)]$ as a function of the variance and bias in the weight update.

We assume that $\mathcal{L}$ is twice differentiable and Δt is small enough such that $\mathcal{L}[\boldsymbol{w}(t+\Delta t)]$ can be effectively approximated around $\mathcal{L}[\boldsymbol{w}(t)]$ using a second order Taylor series, i.e., higher order terms beyond the second order can be ignored. Furthermore, we define a weight update equation that consists of the gradient, a fixed bias, and some noise. We denote the bias vector as $b\overrightarrow{\boldsymbol{\beta}}$, where $\overrightarrow{\boldsymbol{\beta}}$ is a norm $\sqrt{N}$ vector that indicates the direction of bias in the N-dimensional parameter space. A special case of $\overrightarrow{\boldsymbol{\beta}}$ would be $\overrightarrow{\boldsymbol{\beta}} = \overrightarrow{\mathbf{1}}$ when all parameters have the same bias. Similarly, we assume white noise in the gradient estimates. With these assumptions, we define the weight update equation to be:

$$\Delta w(t) := -\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)] + b\overrightarrow{\boldsymbol{\beta}} + \sigma\sqrt{f(N)}\hat{n} \tag{1}$$

where $\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]$ denotes the first derivative of the gradient of $\mathcal{L}$ evaluated at $w(t)$ and $\hat{n}$ is a unit vector in the N-dimensional parameter space whose elements are zero-mean i.i.d. (see Appendix A for generality of this assumption), such that the total variance of $\Delta w(t)$ is $\sigma^2 f(N)$, where $f(N)$ denotes the dependence of variance on the total number of network parameters, N. Finally, we define $\Delta\mathcal{L}_t(b,\sigma^2)$ as the change in $\mathcal{L}[\boldsymbol{w}(t)]$ when performing the aforementioned weight update as compared to the change in $\mathcal{L}[\boldsymbol{w}(t)]$ when the weight update step follows the true gradient, i.e.,

$$\Delta\mathcal{L}_t(b,\sigma^2) = [\mathcal{L}[\boldsymbol{w}(t+\Delta t)] - \mathcal{L}[\boldsymbol{w}(t)]]_{(b,\sigma^2)} - [\mathcal{L}[\boldsymbol{w}(t+\Delta t)] - \mathcal{L}[\boldsymbol{w}(t)]]_{(0,0)} \tag{2}$$

where $[\mathcal{L}[\boldsymbol{w}(t+\Delta t)] - \mathcal{L}[\boldsymbol{w}(t)]]_{(b,\sigma^2)}$ is the change in loss with bias b and variance σ^2. For brevity, we drop the t is the subscript in the rest of the paper. Formally, we present the following Lemma:

Lemma 2.1. ***Second order Taylor series expansion.*** *Assuming a small learning rate, the bias, b, and variance, σ^2, in gradient estimates can be linked to changes in $\mathcal{L}$ upon one weight update step as compared to the change in $\mathcal{L}$ under a true gradient descent weight update step:*

$$\begin{aligned}\mathbb{E}_{\hat{n}}\left[\Delta\mathcal{L}(b,\sigma^2)\right] &= \mathbb{E}_{\hat{n}}\left[[\mathcal{L}[\boldsymbol{w}(t+\Delta t)] - \mathcal{L}[\boldsymbol{w}(t)]]_{(b,\sigma^2)} - [\mathcal{L}[\boldsymbol{w}(t+\Delta t)] - \mathcal{L}[\boldsymbol{w}(t)]]_{(0,0)}\right] \\ &= b\left\langle\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)], \overrightarrow{\boldsymbol{\beta}}\right\rangle\Delta t + \frac{1}{2}b^2\left\langle\overrightarrow{\boldsymbol{\beta}}, \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\overrightarrow{\boldsymbol{\beta}}\right\rangle\Delta t^2 \\ &\quad -\frac{1}{2}b\left\langle\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)], (\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)] + \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]^T)\overrightarrow{\boldsymbol{\beta}}\right\rangle\Delta t^2 + \frac{1}{2}\frac{\sigma^2 f(N)}{N}\mathrm{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]]\Delta t^2\end{aligned} \tag{3}$$

where $\langle\cdot,\cdot\rangle$ denotes the dot product, Tr *denotes the Trace operator of a square matrix, orange terms indicate the impact of bias, and the green term indicates the impact of variance.*

In the next section, we present an in-depth analysis of the impact of variance when $f(N)$ is sublinear, as is the case for some learning algorithms, e.g. Equilibrium Propagation (Laborieux et al., 2021). We present a more general analysis along with the proofs of all lemmas and theorems in Appendix A.

2.1 Analysis of the impact of variance and bias on training loss

One of the corollaries that follow from Lemma 2.1 is that the impact of variance and bias on the expected $\Delta\mathcal{L}_t(b, \sigma^2)$ are independent of each other. Moreover, it is clear that the impact of variance is directly proportional to the trace of $\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]$, i.e., the loss Hessian, and inversely proportional to N, i.e., the number of trainable parameters in the network.

Previous work investigating the object categorization loss landscape for feedforward ANNs demonstrated a pronounced effect of width, i.e., increasing width leads to smoother loss landscapes (Li et al., 2018a), thereby leading to lower trace of Hessian (Hochreiter & Schmidhuber, 1994). This empirical observation in deep ANNs implies that increasing the width both lowers the trace of the Hessian in addition to its obviously increasing the number of trainable network parameters. Therefore, we can say directly from Lemma 2.1 that the impact of variance is lower for wider networks. However, the role of depth is more complicated. Notably, increasing depth (with no skip connections) could theoretically make the loss landscape less smooth (Li et al., 2018a). Thus, we begin by analyzing the case of increasing depth in linear feedforward networks. We prove that, in fact, increasing depth leads to lower values for $\frac{\text{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}]}{N}$, thereby implying a lower impact of variance.

Theorem 2.2. ***Increasing depth lowers impact of variance*** *For linear feedforward ANNs, the impact of variance on $\Delta\mathcal{L}_t(0, \sigma^2)$ for a $(L+1)$ layer network is less than that of a L layer network, i.e.,*

$$\Delta\mathcal{L}_t(0, \sigma^2)_{(L+1)-layer} \leq \Delta\mathcal{L}(0, \sigma^2)_{L-layer} \tag{4}$$

where each layer has d units and weights initialized by the Xavier initialization strategy and the total variance in gradient estimates does not grow with the size of the network, i.e., $f(N)$ is some constant[2]

Although these assumptions on layer weights may not strictly hold over the course of training, empirical experience suggests that there exists a loose correspondence among these quantities if the conditions hold true during initialization. Taken together, overparameterization in either width or depth leads to a lower impact of variance in gradient estimates on the network's training.

Besides the network architecture, the non-linearity used in the network also impacts the function implemented by the network, and therefore affects the trace of the loss Hessian. Specifically, we demonstrate that the trace of the loss Hessian is lower for networks with a threshold non-linearity with gain less than or equal to 1, such as a ReLU operation, as compared to a linear network. Formally,

Theorem 2.3. ***ReLU lowers impact of variance*** *Let $\phi(.)$ denote a threshold non-linearity function. The impact of variance on $\Delta\mathcal{L}_t(0, \sigma^2)$ for a network with such non-linearities is less than that of an equivalent linear network, i.e.,*

$$\Delta\mathcal{L}_t(0, \sigma^2)_{\phi} \leq \Delta\mathcal{L}_t(0, \sigma^2)_{Linear} \tag{5}$$

where the gain of $\phi(.)$ is less than or equal to 1 (e.g. ReLU).

Summarily, the standard practices in deep neural networks, like overparameterized networks or the use of ReLU as a non-linearity, lead to a lower impact of variance on the network's training. Interestingly, brains are also arguably overparameterized and characterized by threshold non-linearities that lead to sparse activations. These properties could offer a potential solution to countering any negative impact of variance in gradient estimates in the brain and allow biological agents to reliably train complex tasks despite not being able to actually conduct true gradient descent (Lillicrap et al., 2020). It also suggests neuromorphic chips could tolerate more noise in gradient estimators if they are made larger.

In contrast to the impact of variance, the impact of bias is more nuanced and depends on the norm and direction of gradient, the direction of the bias vector, as well as the eigenvectors of the loss Hessian. Using a few algebraic manipulations, Lemma 2.1 can be extended to show that:

Theorem 2.4. *The impact of bias on $\Delta\mathcal{L}_t(b, 0)$ grows linearly with the norm of the gradient.*

$$\Delta\mathcal{L}_t(b, 0) = \frac{1}{2}b^2 Q\Delta t^2 + b\|\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\|P\Delta t \tag{6}$$

where P, Q are terms that are independent of the norm of the gradient, $\|\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\|$.

[2]Check Appendix A for an extension to the general case where variance changes with number of parameters.

This result implies that initialization and architectural considerations could play important roles in mitigating the impact of bias on training by limiting the norm of the gradient. In the next section, we will empirically validate these results in linear and shallow networks trained on toy datasets as well as in VGG networks trained on CIFAR-10.

2.2 Empirical verification of the variance and bias results

Our experimental setup (illustrated in Fig. 2) is motivated by Raman et al. (2019), and follows the standard student-teacher framework. A "teacher" ANN is initialized with fixed parameters. The task is for a randomly initialized "student" network to learn the input-output mapping defined by the teacher. We defer the reader to the Appendix B for experimental details.

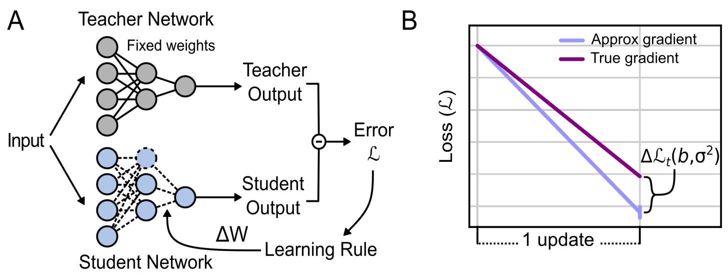


Figure 2: Experimental setup for empirical validation of our results, focusing on $\Delta\mathcal{L}_t(b, \sigma^2)$ to understand how gradient approximations impact learning in one update step. (A) The student-teacher framework motivated by Raman et al. (2019). (B) Illustration of $\Delta\mathcal{L}_t(b, \sigma^2)$ (defined in Eq. (2))

First, we validate our analytical results pertaining to the impact of variance, i.e., Theorems 2.2 and 2.3 in relatively shallow (1-8 hidden layers) fully connected ANNs. We fix the teacher network architecture and use a ReLU non-linearity to threshold the intermediate layer activations. Subsequently, we vary the depth and width of the student network with no non-linearity and observe $\Delta\mathcal{L}_t(0, \sigma^2)$ at different points in the loss landscape. For each such setting, we add a ReLU non-linearity to the student network to validate the effect of a threshold non-linearity on $\Delta\mathcal{L}_t(0, \sigma^2)$. We plot the mean $\Delta\mathcal{L}_t(0, \sigma^2)$ across different points in the loss landscape in Fig. 3. Note that in order to use a sample accurately reflecting the loss landscape that would be encountered during a learning trajectory, we train a control network with the same architecture and non-linearity as the student network using the true gradient and evaluate $\Delta\mathcal{L}_t(0, \sigma^2)$ for each update step along the learning trajectory. In doing so, we are able to observe the desired quantities across a wide range of gradient norm values.

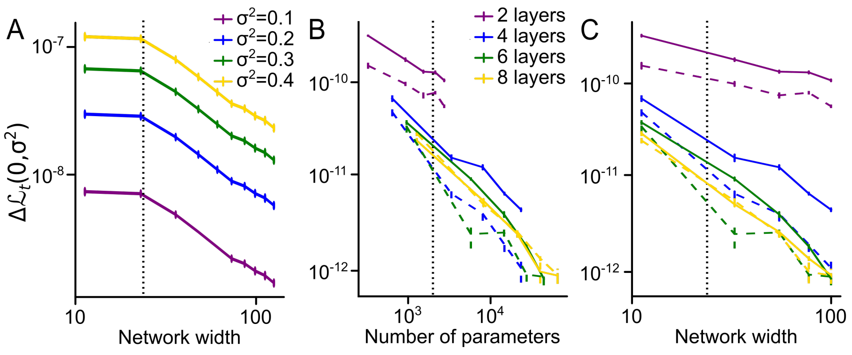


Figure 3: Impact of variance is lower for wider, deeper, and ReLU networks. Solid lines indicate linear networks, dashed lines indicate networks with ReLU non-linearity, and dotted line indicate the teacher network architecture. (A) Varying hidden layer width in one hidden layer linear ANNs, keeping input/output dimensions same. (B) Linear and ReLU networks with varying width and depth plotted with number of parameters in x-axis. (C) Same as B but plotted with network width in x-axis.

Furthermore, we validate Theorems 2.2 and 2.3 on VGG networks trained on the CIFAR-10 dataset. Specifically, we maintain the student-teacher framework and fix the teacher network to be a VGG-19 network trained on CIFAR-10 to 92.6% test accuracy, and we use different networks from the VGG family as student networks (with variance set to $\sigma^2 = 20$). For each network, we use both random weight initialization and Imagenet-pretrained weights to determine whether initialization has an effect on the impact of variance. Fig. 5 demonstrates that our results pertaining to depth and width hold for deep convolutional neural networks (CNNs).

Similarly, we validate Theorem 2.4, first in shallow fully connected ANNs and subsequently in deep CNNs trained on CIFAR-10. Specifically, we demonstrate in Fig. 4 that the impact of bias (with $b = 0.02$) on performance of a linear shallow network grows with the norm of the gradient. Note that the sign duality of $\Delta\mathcal{L}_t(b, 0)$ in this figure follows from Theorem 2.4 where the quantities P and Q depend on the direction of the bias vector with respect to the direction of gradient and the eigenvectors of the loss Hessian (see proof in Appendix A for more details). Therefore, bias can help or hinder learning depending on its direction. This behaviour is in contrast to variance, which always hinders learning for a convex loss (Lemma 2.1). Nonetheless, it is worth noting that when the gradient norm is small as bias increases it always hinders training (see Fig. 4B).

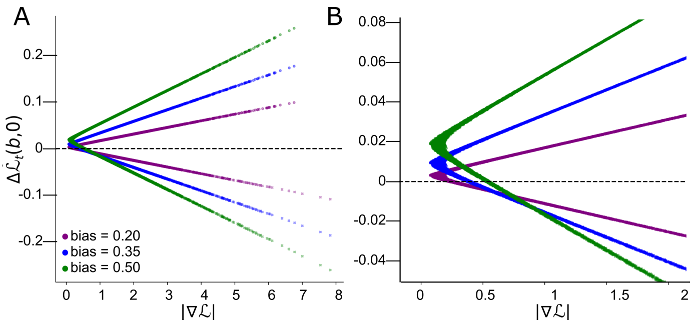


Figure 4: Impact of bias grows linearly with gradient norm and has a quadratic relationship with the amount of bias when the gradient norm tends to 0, i.e., validating expression from Theorem 2.4: $\Delta\mathcal{L}_t(b, 0) = \frac{1}{2}b^2 Q\Delta t^2 + b\|\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\|P\Delta t$. (A) Impact of bias grows linearly with gradient norm, with the slope being the amount of bias (see second term in equation). (B) Impact of bias grows quadratically with amount of bias when gradient norm is small (see first term in equation).

Following this, we measure the absolute value of $\Delta\mathcal{L}_t(b, 0)$ and plot its mean across multiple update steps in VGG networks trained on CIFAR-10. In Fig. 6, we show that the relation stated in Theorem 2.4 holds across different VGG architectures, irrespective of the weight initialization. Furthermore, deeper VGG networks have a lower impact of bias owing to a lower gradient norm.

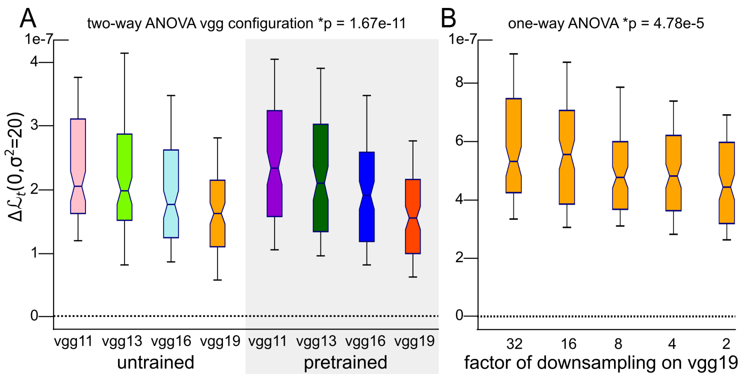


Figure 5: Impact of variance is lower for deeper and wider VGG networks when training on CIFAR-10. (A) Deeper VGG configurations, both Imagenet-pretrained or untrained, exhibit lower impact of variance. (B) Wider VGG-19 configuration networks exhibit lower impact of variance. Factor of downsampling indicates the reduction in the number of filters in convolutional layers and dimensionality of intermediate fully connected layers compared to the original VGG-19 configuration.

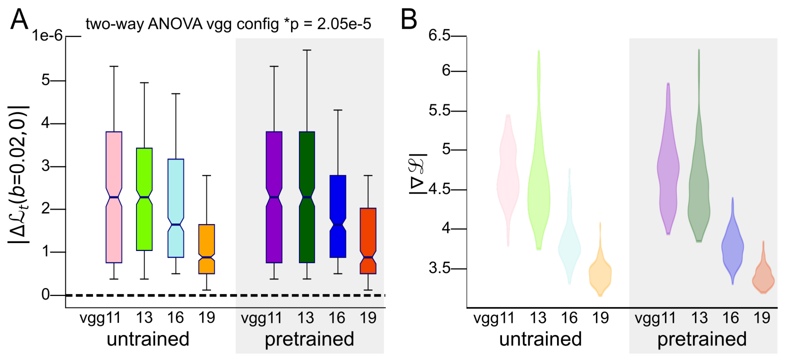


Figure 6: Impact of bias is lower for (A) deeper VGG networks when training on CIFAR-10. (B) Deeper VGG networks exhibit lower gradient norm, thereby mitigating the impact of bias in training.

3 Estimating the Impact of Variance and Bias on Generalization

In the previous section, we studied how variance and bias in gradient estimates impacts training. However, from a machine learning perspective, an important question is to understand the impact on the system's generalization. Specifically, we ask the question: under what conditions could variance and bias in gradient estimates aid generalization performance? Interestingly, current deep learning practices rely on not following the exact gradient of the loss function to train models, which have been demonstrated to help generalization performance. Some common examples of such practices include stochastic gradient descent (SGD) and dropout[3].

In this section, we use our framework to understand how variance and bias in gradient estimates could alter the learning trajectory, thereby impacting generalization. Specifically, we investigate the conditions under which a parameter update would necessarily lead to descent on the loss landscape. This matters for understanding generalization because the flatness of the loss landscape is a good proxy for generalization (Baldassi et al., 2020; Jiang et al., 2019; Sankar et al., 2021; Tsuzuku et al., 2020; Petzka et al., 2021): flatter minima tend to have better generalization performance than sharper ones. We leverage this viewpoint to understand the impact of variance and bias on generalization. Specifically, we investigate the conditions under which gradient approximations could help to avoid sharp minima, thereby promoting convergence to wide flat minima.

3.1 Analysis of the Impact of Variance and Bias Descent of Narrow Minima

To quantify the flatness of loss landscape, we use eigenvalues of the loss Hessian. Without loss of generality, we assume that the loss Hessian is a normal square matrix, i.e., its eigenvectors form an orthogonal basis of the N-dimensional parameter space. In order to understand the properties of loss minima that the network could potentially converge to, we derive the sufficiency conditions under which a parameter update ascends (or descends) the loss landscape. This strategy indicates that noise in gradient estimates would lead to the system not descending the loss landscape when the minima is too sharp, which can aid generalization.

Theorem 3.1. ***Noise helps avoid sharp minima*** *The sufficient condition under which noise prevents the system from descending the loss landscape, i.e.,* $\mathcal{L}[\boldsymbol{w}(t+\Delta t)] \geq \mathcal{L}[\boldsymbol{w}(t)]$*, is*

$$\lambda_1 \geq \lambda_N \geq \frac{2}{\Delta t}\frac{1}{1+\left(\frac{\sigma}{\|\nabla_{\boldsymbol{w}}\mathcal{L}\|}\right)^2} \tag{7}$$

where $\lambda_1 \geq \lambda_2 ... \geq \lambda_N$ *denote the eigenvalues of* $\nabla^2_{\boldsymbol{w}}\mathcal{L}$ *around the minima.*

Interestingly, the condition presented in Theorem 3.1 presents a lower bound for the smallest eigenvalue, i.e., the eigenvalue associated with the direction of least curvature, and connects it to the variance to gradient norm ratio. This ratio can be thought of as the inverse noise ratio when there is

[3]Discussion on the relationship between our analysis and SGD and dropout in the Appendix C.2

no bias in the approximation. Furthermore, we also show in Appendix A that the sufficient condition for descending the loss landscape is similar to the one presented in Theorem 3.1, but provides an upper bound for the largest eigenvalue, i.e., $\lambda_1 \leq \frac{2}{\Delta t} \frac{1}{1+\left(\frac{\sigma}{\|\nabla_{\boldsymbol{w}}\mathcal{L}\|}\right)^2}$. Note that as we get closer to the minimum, the gradient norm would decrease (property of smooth $\mathcal{L}$), and therefore, the inverse noise ratio would increase. Thus, the upper bound on curvature of loss minima that the system can potentially converge to decreases. Taken together, this shows that variance helps the system avoid converging to sharp minima and instead promotes converging to wide flat minima.

Bias, on the other hand, has a more nuanced effect on the learning trajectory. This complexity is unsurprising given the dependence of the impact of bias on direction of the bias vector, the direction of the gradient, and the eigenvectors of the loss Hessian. Once again, we derive the sufficiency condition for when a parameter update step would produce a decrease in the loss function. It turns out this condition depends on both minima flatness and amount of bias relative to the gradient norm.

Theorem 3.2. ***Bias prevents descent into local minima dependent on Hessian spectrum.*** *Bias in gradient estimates could prevent converging to a minima. Specifically, the sufficient condition for the weight updates to produce a decrease in $\mathcal{L}$ when using a biased estimate of the gradient is:*

$$\frac{b}{\|\nabla_{\boldsymbol{w}}\mathcal{L}\|} \leq \frac{1}{\sqrt{N}} \frac{2}{1+\psi\Delta t+\sqrt{1+4\psi\Delta t+\psi^2\Delta t^2}} \approx \frac{1}{\sqrt{N}} \frac{1}{(1+\frac{3}{2}\psi\Delta t)} \tag{8}$$

where $\psi^2 = \sum_i \lambda_i^2$, i.e., the Frobenius norm of the loss Hessian, and $\lambda_1 \geq \lambda_2 ... \geq \lambda_N$ denote the eigenvalues of $\nabla^2_{\boldsymbol{w}}\mathcal{L}$ around the minima and $\nabla^2_{\boldsymbol{w}}\mathcal{L}$ is a normal matrix.

The condition in Theorem 3.2 indicates that a system that follows a biased gradient approximation will descend the loss landscape until some neighbourhood around a minimum (characterized by the minima flatness and relative bias). Inside this neighbourhood, the system may not descend the loss landscape and therefore not converge to the minimum. The condition depends on minima flatness due to the ψ term (Eq. (8)): sharper (higher curvature) minima should have higher ψ, which makes it less likely to satisfy the condition. The condition also depends on the amount of bias relative to gradient norm, i.e.,, higher relative bias makes it less likely to satisfy the condition for descent, which we also support through simulation (see Fig. 7C). Taken together, variance and bias in gradient estimates promote gradient descent dynamics that converge to wide flat minima, which can aid generalization. This effect is empirically demonstrated by our first example of VGG-16 networks trained on CIFAR-10 (Fig. 1).

3.2 Empirical verification of the impact of variance and bias on generalization

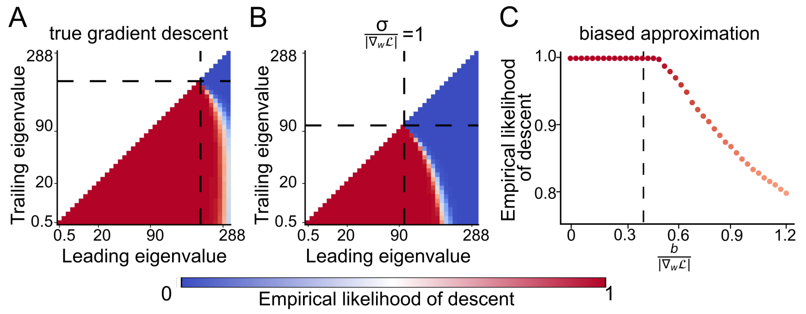


Figure 7: Empirical verification of Theorems 3.1 and 3.2 in linear ANNs. Dashed lines indicate the theoretical limit and colours indicate the empirical probability of descent. (A) True gradient descent. (B) Noise in gradient estimates, $\sigma = \|\nabla_{\boldsymbol{w}}\mathcal{L}\|$. (C) Bias in gradient estimates, ratio $\frac{b}{\|\nabla_{\boldsymbol{w}}\mathcal{L}\|}$ is varied.

We follow the same experimental setting as described in Section 2.2 for verifying Theorems 3.1 and 3.2. Owing to computational bottlenecks in loss Hessian estimation for high-dimensional nonlinear settings, we restrict our empirical validation to training linear networks for optimizing MSE loss. In doing so, we gain full control over the loss Hessian via the input covariance statistics[4]—Fig. 7A & B demonstrate that our inferences from Theorem 3.1 hold empirically. When the leading

[4]for MSE loss at minima, the Hessian is equal to the input covariance matrix barring a constant factor

eigenvalue is *less than* the theoretical limit, a weight update step always leads to a decrease in $\mathcal{L}$; when the trailing eigenvalue is *greater* than the theoretical limit, a weight update step always leads to an increase in $\mathcal{L}$. Furthermore, the theoretical limit is higher for the case when there is no variance than with variance. These observations demonstrate that following a noisy version of the gradient helps the network avoid sharp local minima. Similarly, Fig. 7C demonstrates that our inferences from Theorem 3.2 hold empirically. Specifically, when the ratio of bias to gradient norm is below the theoretical threshold, a weight update step always leads to a decrease in $\mathcal{L}$.

4 Discussion

In both computational neuroscience and neuromorphic computing, gradient estimators are known to have variance and bias due to both algorithmic and hardware constraints (Laborieux et al., 2021; M. Payvand et al., 2020a; Lillicrap et al., 2020; Pozzi et al., 2018; Sacramento et al., 2018; Rubin et al., 2021; Roelfsema & Holtmaat, 2018; Bellec et al., 2020; Neftci et al., 2019; Huh & Sejnowski, 2018; Zenke & Neftci, 2021). Using mathematical analysis and empirical verification, we found that the impact of variance and bias on learning is determined by the network size, activation function, gradient norm, and loss Hessian, such that the amount of variance and bias that are permissible or even desirable depends on these properties. Though our empirical verification was done using feedforward ANNs, our mathematical analysis was formulated to be as general as possible, and we believe that our assumptions are reasonable for most cases that computational neuroscientists and neuromorphic chip engineers face. Furthermore, we add a corollary to Lemma 2.1 and discuss extensions of our analysis to multi-step optimization settings in Appendix A (see Corollary A.2). Thus, our work can inform research in these areas by providing a guide for how much variance and bias relative to the gradient norm is reasonable for a given network.

4.1 Related work

Our analysis was inspired in part by recent work by Raman et al. (2019) characterizing bounds on learning performance in neural circuits. They demonstrated that in the absence of noise in the gradient estimator, larger networks are always better at training, but with noise added, there is a size limit beyond which training is impaired. Our work was also informed by research into generalization in deep ANNs, which provided the rationale for our analyses examining the potential for a learning algorithm to descend into sharp minima or not (Foret et al., 2020; Chaudhari et al., 2019; Yao et al., 2018; Ghorbani et al., 2019; Smith et al., 2021). As well, our work has some clear relationship to other work examining the variance of gradient estimators (e.g. Werfel et al. (2003)), but here, we are asking a novel question, namely, how does a given amount of variance and bias impact performance for different network parameters?

More broadly, our work relates to the deep learning literature because modern ANNs rarely use the true loss gradient over the entire dataset to make weight updates. Instead, it is common practice to add noise in different forms, e.g. SGD (Bottou, 2012), dropout (Srivastava et al., 2014), dropconnect (Wan et al., 2013) and label noise (Blanc et al., 2020; HaoChen et al., 2021; Damian et al., 2021). But, the noise structure in these scenarios may differ from assumptions in our work. For example, SGD does not exhibit white noise structure (Xie et al., 2021; Zhu et al., 2019). [5] Likewise, dropout can be thought of as adding noise, but for a truly unbiased estimate of the gradient, one would have to consider all possible configurations of dropped neurons, something quite uncommon in practice.

4.2 Limitations and future work

This work focused on characterizing the impact of variance and bias in gradient estimates, but it lacks any in-depth analysis of specific bio-plausible or neuromorphic learning algorithms. Similarly, our work only characterizes learning performance, but does not provide a concrete proposal to mitigate excessive noise or bias in gradient approximations. As such, our analyses can be used by other researchers to assess their learning algorithms but they do not directly speak to any specific learning algorithm nor provide mechanisms for improving existing algorithms. Nonetheless, we believe that our work provides a framework to help develop such strategies, and we leave it to future work.

[5]Though several analytical studies have used white noise to model SGD dynamics, and concluded that the SGD noise acts as a regularizer against converging to sharp minima (Li et al., 2021), similar to our analyses.

Reproducibility Statement

We strongly believe that reproducibility is critical for research progress in both machine learning and computational neuroscience. To this end, we have provided thorough experimental details in the appendix, and in the supplementary materials, we have included the code to run all of the experiments and generate the figures. Our code can also be accessed from the project's github repo. We believe that this information will allow the community to validate/replicate our results and further build on our work.

Acknowledgements

The authors would like to thank Abhishek Banerjee, Jonathan Cornford, and Shahab Bakhtiari for insightful discussions and helpful feedback over the course of the project, as well as Zahraa Chorghay and Colleen Gillon for their aesthetic contribution to the manuscript and figures. This research was generously supported by Vanier Canada Graduate scholarship and Healthy Brains for Healthy Lives fellowship (A.G.); NSERC PGS-D and NSF AccelNet IN-BIC program, Grant No. OISE-2019976 AM02 (Y.H.L.); NSERC Discovery Grant RGPIN-2018-04821, FRQS Research Scholar Award, Junior 1 LAJGU0401-253188, Canada Research Chair in Neural Computations and Interfacing (G.L.); NSERC (Discovery Grant: RGPIN-2020-05105; Discovery Accelerator Supplement: RGPAS-2020-00031), Healthy Brains, Healthy Lives (New Investigator Award: 2b-NISU-8) (B.A.R.); CIFAR Learning in Machines and Brains Program (K.K. & B.A.R.); Canada CIFAR AI Chair program (G.L. & B.A.R.). This work was enabled by the material support of NVIDIA in the form of computational resources and support provided by Mila.

References

Carlo Baldassi, Fabrizio Pittorino, and Riccardo Zecchina. Shaping the learning landscape in neural networks around wide flat minima. *Proceedings of the National Academy of Sciences*, 117(1): 161–170, 2020.

Guillaume Bellec, Franz Scherr, Anand Subramoney, Elias Hajek, Darjan Salaj, Robert Legenstein, and Wolfgang Maass. A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature Communications*, 11(1):3625, July 2020. ISSN 2041-1723. doi: 10.1038/s41467-020-17236-y. URL `https://www.nature.com/articles/s41467-020-17236-y`. Number: 1 Publisher: Nature Publishing Group.

Guy Blanc, Neha Gupta, Gregory Valiant, and Paul Valiant. Implicit regularization for deep neural networks driven by an ornstein-uhlenbeck like process. In *Conference on learning theory*, pp. 483–513. PMLR, 2020.

Léon Bottou. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pp. 421–436. Springer, 2012.

Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124018, 2019.

Cramer Benjamin, Billaudelle Sebastian, Kanya Simeon, Leibfried Aron, Grübl Andreas, Karasenko Vitali, Pehle Christian, Schreiber Korbinian, Stradmann Yannik, Weis Johannes, Schemmel Johannes, and Zenke Friedemann. Surrogate gradients for analog neuromorphic computing. *Proceedings of the National Academy of Sciences*, 119(4):e2109194119, January 2022. doi: 10.1073/pnas.2109194119. URL `https://doi.org/10.1073/pnas.2109194119`. Publisher: Proceedings of the National Academy of Sciences.

Alex Damian, Tengyu Ma, and Jason D Lee. Label noise sgd provably prefers flat global minimizers. *Advances in Neural Information Processing Systems*, 34, 2021.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. An investigation into neural net optimization via hessian eigenvalue density. In *International Conference on Machine Learning*, pp. 2232–2241. PMLR, 2019.

Jeff Z HaoChen, Colin Wei, Jason Lee, and Tengyu Ma. Shape matters: Understanding the implicit bias of the noise covariance. In *Conference on Learning Theory*, pp. 2315–2357. PMLR, 2021.

Sepp Hochreiter and Jürgen Schmidhuber. Simplifying neural nets by discovering flat minima. *Advances in neural information processing systems*, 7, 1994.

Dongsung Huh and Terrence J Sejnowski. Gradient descent for spiking neural networks. *Advances in neural information processing systems*, 31, 2018.

Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. In *International Conference on Learning Representations*, 2019.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Master's thesis*, 2009.

Axel Laborieux, Maxence Ernoult, Benjamin Scellier, Yoshua Bengio, Julie Grollier, and Damien Querlioz. Scaling Equilibrium Propagation to Deep ConvNets by Drastically Reducing Its Gradient Estimator Bias. *Frontiers in Neuroscience*, 15, 2021. ISSN 1662-453X. URL `https://www.frontiersin.org/article/10.3389/fnins.2021.633674`.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31, 2018a.

Yibo Li, Zhongrui Wang, Rivu Midya, Qiangfei Xia, and J Joshua Yang. Review of memristor devices in neuromorphic computing: materials sciences and device challenges. *Journal of Physics D: Applied Physics*, 51(50):503002, September 2018b. ISSN 0022-3727. doi: 10.1088/1361-6463/aade3f. URL `http://dx.doi.org/10.1088/1361-6463/aade3f`. Publisher: IOP Publishing.

Zhiyuan Li, Sadhika Malladi, and Sanjeev Arora. On the validity of modeling sgd with stochastic differential equations (sdes). *Advances in Neural Information Processing Systems*, 34, 2021.

Timothy P. Lillicrap, Daniel Cownden, Douglas B. Tweed, and Colin J. Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications*, 7: 13276, November 2016. URL `http://dx.doi.org/10.1038/ncomms13276`.

Timothy P Lillicrap, Adam Santoro, Luke Marris, Colin J Akerman, and Geoffrey Hinton. Backpropagation and the brain. *Nature Reviews Neuroscience*, 21(6):335–346, 2020.

Yuhan Helena Liu, Stephen Smith, Stefan Mihalas, Eric Shea-Brown, and Uygar Sümbül. Cell-type–specific neuromodulation guides synaptic credit assignment in a spiking neural network. *Proceedings of the National Academy of Sciences*, 118(51), 2021.

M. Payvand, M. E. Fouda, F. Kurdahi, A. Eltawil, and E. O. Neftci. Error-triggered Three-Factor Learning Dynamics for Crossbar Arrays. In *2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pp. 218–222, September 2020a. doi: 10.1109/AICAS48895.2020.9073998. Journal Abbreviation: 2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS).

M. Payvand, M. E. Fouda, F. Kurdahi, A. M. Eltawil, and E. O. Neftci. On-Chip Error-Triggered Learning of Multi-Layer Memristive Spiking Neural Networks. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 10(4):522–535, December 2020b. ISSN 2156-3365. doi: 10.1109/JETCAS.2020.3040248.

Alan Murray and Peter Edwards. Synaptic weight noise during mlp learning enhances fault-tolerance, generalization and learning trajectory. *Advances in neural information processing systems*, 5, 1992.

James M Murray. Local online learning in recurrent networks with random feedback. *ELife*, 8: e43299, 2019.

N. R. Shanbhag, N. Verma, Y. Kim, A. D. Patil, and L. R. Varshney. Shannon-Inspired Statistical Computing for the Nanoscale Era. *Proceedings of the IEEE*, 107(1):90–107, January 2019. ISSN 1558-2256. doi: 10.1109/JPROC.2018.2869867.

Emre O. Neftci, Charles Augustine, Somnath Paul, and Georgios Detorakis. Event-Driven Random Back-Propagation: Enabling Neuromorphic Deep Learning Machines. *Frontiers in Neuroscience*, 11, 2017. ISSN 1662-453X. URL `https://www.frontiersin.org/article/10.3389/fnins.2017.00324`.

Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019.

Antonio Orvieto, Hans Kersting, Frank Proske, Francis Bach, and Aurelien Lucchi. Anticorrelated noise injection for improved generalization. *arXiv preprint arXiv:2202.02831*, 2022.

Alexandre Payeur, Jordan Guerguiev, Friedemann Zenke, Blake A. Richards, and Richard Naud. Burst-dependent synaptic plasticity can coordinate learning in hierarchical circuits. *Nature Neuroscience*, 24(7):1010–1019, July 2021. ISSN 1546-1726. doi: 10.1038/s41593-021-00857-x. URL `https://doi.org/10.1038/s41593-021-00857-x`.

Henning Petzka, Michael Kamp, Linara Adilova, Cristian Sminchisescu, and Mario Boley. Relative flatness and generalization. *Advances in Neural Information Processing Systems*, 34, 2021.

Isabella Pozzi, Sander Bohté, and Pieter Roelfsema. A biologically plausible learning rule for deep learning in the brain. *arXiv preprint arXiv:1811.01768*, 2018.

Dhruva Venkita Raman, Adriana Perez Rotondo, and Timothy O'Leary. Fundamental bounds on learning performance in neural circuits. *Proceedings of the National Academy of Sciences*, 116 (21):10537–10546, 2019. ISSN 0027-8424.

Pieter R. Roelfsema and Anthony Holtmaat. Control of synaptic plasticity in deep cortical networks. *Nature Reviews Neuroscience*, 19(3):166–180, feb 2018. ISSN 14710048. doi: 10.1038/nrn.2018.6.

Thorsteinn Rögnvaldsson. On langevin updating in multilayer perceptrons. *Neural computation*, 6 (5):916–926, 1994.

Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784):607–617, November 2019. ISSN 1476-4687. doi: 10.1038/s41586-019-1677-2. URL `https://doi.org/10.1038/s41586-019-1677-2`.

Jonathan E Rubin, Catalina Vich, Matthew Clapp, Kendra Noneman, and Timothy Verstynen. The credit assignment problem in cortico-basal ganglia-thalamic networks: A review, a problem and a possible solution. *European Journal of Neuroscience*, 53(7):2234–2253, 2021.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(9):533–536, 1986.

João Sacramento, Rui Ponte Costa, Yoshua Bengio, and Walter Senn. Dendritic cortical microcircuits approximate the backpropagation algorithm. *Advances in neural information processing systems*, 31, 2018.

Adepu Ravi Sankar, Yash Khasbage, Rahul Vigneswaran, and Vineeth N Balasubramanian. A deeper look at the hessian eigenspectrum of deep neural networks and its applications to regularization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35:11, pp. 9481–9488, 2021.

Benjamin Scellier and Yoshua Bengio. Equilibrium Propagation: Bridging the Gap between Energy-Based Models and Backpropagation. *Frontiers in Computational Neuroscience*, 11:24, 2017. ISSN 1662-5188. doi: 10.3389/fncom.2017.00024. URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC5415673/`.

Genki Shimizu, Kensuke Yoshida, Haruo Kasai, and Taro Toyoizumi. Computational roles of intrinsic synaptic dynamics. *Computational Neuroscience*, 70:34–42, October 2021. ISSN 0959-4388. doi: 10.1016/j.conb.2021.06.002. URL https://www.sciencedirect.com/science/article/pii/S0959438821000672.

Samuel L Smith, Benoit Dherin, David Barrett, and Soham De. On the origin of implicit regularization in stochastic gradient descent. In *International Conference on Learning Representations*, 2021.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Yusuke Tsuzuku, Issei Sato, and Masashi Sugiyama. Normalized flat minima: Exploring scale invariant definition of flat minima for neural networks using pac-bayesian analysis. In *International Conference on Machine Learning*, pp. 9636–9647. PMLR, 2020.

Andrea Vedaldi and Andrew Zisserman. Vgg convolutional neural networks practical. *Department of Engineering Science, University of Oxford*, 66, 2016.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *International conference on machine learning*, pp. 1058–1066. PMLR, 2013.

Justin Werfel, Xiaohui Xie, and H Seung. Learning curves for stochastic gradient descent in linear feedforward networks. *Advances in neural information processing systems*, 16, 2003.

James C.R. Whittington and Rafal Bogacz. Theories of Error Back-Propagation in the Brain. *Trends in Cognitive Sciences*, 2019. ISSN 1364-6613. doi: 10.1016/j.tics.2018.12.005. URL https://doi.org/10.1016/j.tics.2018.12.005.

Zeke Xie, Issei Sato, and Masashi Sugiyama. A diffusion theory for deep learning dynamics: Stochastic gradient descent exponentially favors flat minima. In *International Conference on Learning Representations*, 2021.

Zhewei Yao, Amir Gholami, Qi Lei, Kurt Keutzer, and Michael W Mahoney. Hessian-based analysis of large batch training and robustness to adversaries. *Advances in Neural Information Processing Systems*, 31, 2018.

Friedemann Zenke and Emre O Neftci. Brain-inspired learning on neuromorphic substrates. *Proceedings of the IEEE*, 109(5):935–950, 2021.

Zhanxing Zhu, Jingfeng Wu, Bing Yu, Lei Wu, and Jinwen Ma. The anisotropic noise in stochastic gradient descent: Its behavior of escaping from sharp minima and regularization effects. In *International Conference on Machine Learning*, pp. 7654–7663. PMLR, 2019.

A Proofs

Lemma A.1. ***Second order Taylor series expansion.*** *Let us assume for a small learning rate, $\mathcal{L}[\boldsymbol{w}(t+\Delta t)]$ can be approximated using the Taylor series expansion about the point $w(t)$ and dropping the third and higher order terms. The resulting expression linking bias, b, and variance, $\sigma^2 f(N)$, in gradient estimates to changes in $\mathcal{L}$ upon one weight update step as compared to the change in $\mathcal{L}$ under a true gradient descent weight update step is:*

$$\begin{aligned}\mathbb{E}_{\hat{n}}\left[\Delta Loss(b,\sigma^2)\right] &= \mathbb{E}_{\hat{n}}\left[[\mathcal{L}[\boldsymbol{w}(t+\Delta t)]-\mathcal{L}[\boldsymbol{w}(t)]]_{(b,\sigma^2)} - [\mathcal{L}[\boldsymbol{w}(t+\Delta t)]-\mathcal{L}[\boldsymbol{w}(t)]]_{(0,0)}\right]\\ &= b\left\langle \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)], \overrightarrow{\boldsymbol{\beta}}\right\rangle \Delta t + \frac{1}{2}b^2\left\langle \overrightarrow{\boldsymbol{\beta}}, \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\overrightarrow{\boldsymbol{\beta}}\right\rangle \Delta t^2\\ &- \frac{1}{2}b\left\langle \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)], (\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)] + \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]^T)\overrightarrow{\boldsymbol{\beta}}\right\rangle \Delta t^2\\ &+ \frac{1}{2}\frac{\sigma^2 f(N)}{N}\,\mathrm{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]]\Delta t^2 \end{aligned} \tag{9}$$

where N denotes the dimensionality of $w(t)$, $f(N)$ indicates how the total noise in gradient estimates changes with the number of parameters in the model, $\hat{n}$ denotes a random unit vector (entries drawn i.i.d.) in the N-dimensional parameter space and $b\overrightarrow{\boldsymbol{\beta}}$ denotes the bias vector, where $\overrightarrow{\boldsymbol{\beta}}$ is a norm $\sqrt{N}$ vector (i.e. a vector with 2-norm equal to $\sqrt{N}$) that indicates the direction of bias in the N-dimensional parameter space. A special case of $\overrightarrow{\boldsymbol{\beta}}$ would be $\overrightarrow{\boldsymbol{\beta}} = \overrightarrow{\mathbf{1}}$ when all parameters have the same bias level. Also, Tr *denotes the Trace operator of a square matrix.*

Proof. In order to understand how the task error changes as system parameters evolve, we rely on the second order Taylor series expansion of the task error. We denote the system parameters at time t as $w(t)$ and the corresponding task error as $\mathcal{L}[\boldsymbol{w}(t)]$. We assume the parameter update rule to be a noisy version of the gradient descent update:

$$\Delta w(t) = -\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)] + b\overrightarrow{\boldsymbol{\beta}} + \sigma\sqrt{f(N)}\hat{n} \tag{10}$$

Writing out the Taylor series expansion for a small step Δt in the above direction,

$$\mathcal{L}[\boldsymbol{w}(t+\Delta t)] = \mathcal{L}[\boldsymbol{w}(t)] + \langle \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)], \Delta w(t)\rangle \Delta t + \frac{1}{2}\left\langle \Delta w(t), \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\Delta w(t)\right\rangle \Delta t^2 \tag{11}$$

We compare the reduction in task error while following the above parameter update rule to the reduction in task error while following the uncorrupted or true gradient descent updates. Specifically, we observe the following quantity:

$$\begin{aligned}\Delta\mathcal{L}_t(bias=b, var=\sigma^2) := {}& [\mathcal{L}[\boldsymbol{w}(t+\Delta t)] - \mathcal{L}[\boldsymbol{w}(t)]]_{(bias=b,var=\sigma^2)}\\ &- [\mathcal{L}[\boldsymbol{w}(t+\Delta t)] - \mathcal{L}[\boldsymbol{w}(t)]]_{(bias=0,var=0)}\end{aligned} \tag{12}$$

For sake of brevity, we drop the t from the subscript in the rest of the derivations. Plugging in Equations 10 and 11 in Equation 12,

$$\begin{aligned}\Delta\mathcal{L}(b,\sigma^2) = {}& \left[b\left\langle \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)], \overrightarrow{\boldsymbol{\beta}}\right\rangle + \sigma\sqrt{f(N)}\,\langle \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)], \hat{n}\rangle\right]\Delta t\\ &+ \frac{1}{2}\left[-b\left\langle \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)], \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\overrightarrow{\boldsymbol{\beta}}\right\rangle - \sigma\sqrt{f(N)}\left\langle \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)], \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\hat{n}\right\rangle\right.\\ &- b\left\langle \overrightarrow{\boldsymbol{\beta}}, \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\right\rangle - \sigma\sqrt{f(n)}\left\langle \hat{n}, \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\right\rangle\\ &+ b\sigma\sqrt{f(n)}\left\langle \overrightarrow{\boldsymbol{\beta}}, \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\hat{n}\right\rangle + b\sigma\sqrt{f(n)}\left\langle \hat{n}, \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\overrightarrow{\boldsymbol{\beta}}\right\rangle\\ &\left.+b^2\left\langle \overrightarrow{\boldsymbol{\beta}}, \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\overrightarrow{\boldsymbol{\beta}}\right\rangle + \sigma^2 f(N)\,\langle \hat{n}, \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\hat{n}\rangle\right]\Delta t^2\end{aligned} \tag{13}$$

We can further simplify the above expression by taking the expectation over different samples to evaluate the task error and multiple weight update steps to understand the average impact of bias and variance in parameter update estimates. Under the aforementioned expectation, dot product of a deterministic vector with the noise vector $\hat{n}$ is assumed to be 0. Therefore, the expression for $\Delta\mathcal{L}$ simplifies to:

$$\begin{aligned}\mathbb{E}_{\hat{n}}[\Delta\mathcal{L}(b,\sigma^2)] = b\left\langle \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)], \overrightarrow{\boldsymbol{\beta}}\right\rangle \Delta t + \frac{1}{2}\Big[&-b\left\langle \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)], \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\overrightarrow{\boldsymbol{\beta}}\right\rangle \\ &- b\left\langle \overrightarrow{\boldsymbol{\beta}}, \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\right\rangle + b^2\left\langle \overrightarrow{\boldsymbol{\beta}}, \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\overrightarrow{\boldsymbol{\beta}}\right\rangle \\ &+\sigma^2 f(N)\mathbb{E}_{\hat{n}}[\langle \hat{n}, \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\hat{n}\rangle]\Big]\,\Delta t^2 \end{aligned} \tag{14}$$

Furthermore, we assume the following properties for elements of the N-dimensional isotropic noise vector $\hat{n}$:

$$\mathbb{E}_{\hat{n}}[\hat{n}_i^2] = \frac{1}{N} \quad \text{And} \quad \mathbb{E}_{\hat{n}}[\hat{n}_i\hat{n}_j] = 0 \quad \forall \quad i \neq j \in 1...N$$

Therefore,
$$\begin{aligned}\mathbb{E}_{\hat{n}}\left[\langle \hat{n}, \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\hat{n}\rangle\right] &= \mathbb{E}_{\hat{n}}\left[\sum_{i,j}\hat{n}_i\hat{n}_j\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]_{ij}\right] \\ &= \sum_i \mathbb{E}_{\hat{n}}\left[\hat{n}_i^2\right]\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]_{ii} + \sum_{i\neq j}\sum_j \mathbb{E}_{\hat{n}}\left[\hat{n}_i\hat{n}_j\right]\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]_{ij} \\ &= \frac{1}{N}\sum_i \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]_{ii} = \frac{1}{N}\,\mathrm{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]] \end{aligned} \tag{15}$$

Note that Eq. (15) is similar to Hutchinson's trace estimator. We can also leverage the following relationship:

$$\begin{aligned}&\left\langle \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)], \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\overrightarrow{\boldsymbol{\beta}}\right\rangle + \left\langle \overrightarrow{\boldsymbol{\beta}}, \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\right\rangle \\ &\qquad = \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]^T\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\overrightarrow{\boldsymbol{\beta}} + \overrightarrow{\boldsymbol{\beta}}^T\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)] \\ &\qquad\qquad [\text{Since } \langle a,b\rangle = a^Tb] \\ &\qquad = \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]^T\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\overrightarrow{\boldsymbol{\beta}} + \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]^T\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]^T\overrightarrow{\boldsymbol{\beta}} \\ &\qquad [\text{Since each term is a scalar, transpose of a term is equal to itself}] \\ &\qquad = \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]^T(\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)] + \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]^T)\overrightarrow{\boldsymbol{\beta}} \\ &\qquad = \left\langle \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)], (\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)] + \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]^T)\overrightarrow{\boldsymbol{\beta}}\right\rangle \end{aligned} \tag{16}$$

Now, incorporating Equations 15 and 16 in order to simplify Equation 14:

$$\begin{aligned}\mathbb{E}_{\hat{n}}\left[\Delta\mathcal{L}(b,\sigma^2)\right] &= b\left\langle \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)], \overrightarrow{\boldsymbol{\beta}}\right\rangle \Delta t \\ &+ \frac{1}{2}b^2\left\langle \overrightarrow{\boldsymbol{\beta}}, \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\overrightarrow{\boldsymbol{\beta}}\right\rangle \Delta t^2 \\ &- \frac{1}{2}b\left\langle \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)], (\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)] + \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]^T)\overrightarrow{\boldsymbol{\beta}}\right\rangle \Delta t^2 \\ &+ \frac{1}{2}\frac{\sigma^2 f(N)}{N}\,\mathrm{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]]\Delta t^2 \end{aligned} \tag{17}$$

□

Note on i.i.d. noise assumption: Our assumption of i.i.d. noise allows us to derive a closed form for $\Delta\mathcal{L}(b,\sigma^2)$ and investigate the impact of noise in gradient estimates in detail (below). However, it should be noted that there may be learning algorithms that do not adhere to these assumptions, which does mean we cannot state that our analysis is completely learning rule agnostic. Nonetheless, these assumptions (zero in expectation and iid) are sufficiently general that they apply to a number of existing algorithms (e.g. noise perturbation, AGREL, and EQ-prop with the beta parameters selected to have expectation of zero (Murray & Edwards, 1992; Scellier & Bengio, 2017)), and we suspect they can apply to many more.

Corollary A.2. ***Extending Taylor series expansion result to multiple sequential update steps.*** *With the same assumptions as Lemma A.1, let $\tilde{\boldsymbol{w}}_k$ be the weights of the network after k sequential noisy updates and $\boldsymbol{w}_k$ be the weights of the network after k steps of gradient descent, given both networks start from the same initial weights, $\boldsymbol{w}_0$. Let us denote the bias and variance vectors at each update step as $\boldsymbol{b}_i$ and $\sigma_i \hat{n}_i$. Then,*

$$\nabla_{\boldsymbol{w}} \mathcal{L}[\tilde{\boldsymbol{w}}_k] = \nabla_{\boldsymbol{w}} \mathcal{L}[\boldsymbol{w}_k] + \Delta t \left(\nabla^2_{\boldsymbol{w}} \mathcal{L}[\boldsymbol{w}_k] (\sum_{i=1}^{k} \boldsymbol{b}_i) + \nabla^2_{\boldsymbol{w}} \mathcal{L}[\boldsymbol{w}_k] (\sum_{i=1}^{k} \sigma_i \hat{n}_i) \right) + \mathcal{O}(\Delta t^2) \tag{18}$$

$$\begin{aligned} \mathbb{E}_{\hat{n}_1, \hat{n}_2 \dots \hat{n}_k} [\mathcal{L}[\tilde{\boldsymbol{w}}_k]] = {} & \mathcal{L}[\boldsymbol{w}_k] + \Delta t \sum_{i=1}^{k} \langle \boldsymbol{b}_i, \nabla_{\boldsymbol{w}} \mathcal{L}[\boldsymbol{w}_{i-1}] \rangle + \frac{1}{2} \Delta t^2 \sum_{i=1}^{k} \langle \boldsymbol{b}_i, \nabla^2_{\boldsymbol{w}} \mathcal{L}[\boldsymbol{w}_{i-1}] \boldsymbol{b}_i \rangle \\ & - \frac{1}{2} \Delta t^2 \sum_{i=1}^{k} \langle \boldsymbol{b}_i, \left(\nabla^2_{\boldsymbol{w}} \mathcal{L}[\boldsymbol{w}_{i-1}] + \nabla^2_{\boldsymbol{w}} \mathcal{L}[\boldsymbol{w}_{i-1}]^T \right) \nabla_{\boldsymbol{w}} \mathcal{L}[\boldsymbol{w}_{i-1}] \rangle \\ & + \frac{1}{2} \Delta t^2 \frac{1}{N} \sum_{i=1}^{k} \sigma_i^2 f(N) \operatorname{Tr}[\nabla^2_{\boldsymbol{w}} \mathcal{L}[\boldsymbol{w}_{i-1}]] \\ & + \Delta t^2 \sum_{i=1}^{k} \left\langle \nabla^2_{\boldsymbol{w}} \mathcal{L}[\boldsymbol{w}_{i-1}] \left(\sum_{j=1}^{i-1} \boldsymbol{b}_j \right), \boldsymbol{b}_i - 2 \nabla_{\boldsymbol{w}} \mathcal{L}[\boldsymbol{w}_{i-1}] \right\rangle \end{aligned} \tag{19}$$

Proof. We will prove the corollary by induction. First, we will use the Taylor series expansion of the gradient to show that Eq. (18) holds for $k = 1$:

$$\begin{aligned} \nabla_{\boldsymbol{w}} \mathcal{L}[\tilde{\boldsymbol{w}}_1] &= \nabla_{\boldsymbol{w}} \mathcal{L}[\boldsymbol{w}_0 - \nabla_{\boldsymbol{w}} \mathcal{L}[\boldsymbol{w}_0] \Delta t + (\boldsymbol{b}_1 + \sigma_1 \hat{n}_1) \Delta t] \\ &= \nabla_{\boldsymbol{w}} \mathcal{L}[\boldsymbol{w}_1 + (\boldsymbol{b}_1 + \sigma_1 \hat{n}_1) \Delta t] \\ &= \nabla_{\boldsymbol{w}} \mathcal{L}[\boldsymbol{w}_1] + \nabla^2_{\boldsymbol{w}} \mathcal{L}[\boldsymbol{w}_1] (\boldsymbol{b}_1 + \sigma_1 \hat{n}_1) \Delta t + \mathcal{O}(\Delta t^2) \end{aligned} \tag{20}$$

Assuming Eq. (18) holds for for the first k steps, we will now prove it holds for $(k+1)^{th}$ step. Given that any change in the loss Hessian, i.e. $\nabla^2 \mathcal{L}$, will only result in third order effects, we will ignore these changes although we will use a different subscript to denote the step at which the Hessian is computed. This approximation is akin to assuming that the step sizes are small enough such that the loss landscape is roughly quadratic in nature. Similar to the proof for $k = 1$, we use the Taylor series expansion:

$$\begin{aligned} \nabla_{\boldsymbol{w}} \mathcal{L}[\tilde{\boldsymbol{w}}_{k+1}] &= \nabla_{\boldsymbol{w}} \mathcal{L}[\tilde{\boldsymbol{w}}_k - \nabla_{\boldsymbol{w}} \mathcal{L}[\tilde{\boldsymbol{w}}_k] \Delta t + (\boldsymbol{b}_{k+1} + \sigma_{k+1} \hat{n}_{k+1}) \Delta t] \\ &= \nabla_{\boldsymbol{w}} \mathcal{L}[\boldsymbol{w}_k + \tilde{\boldsymbol{w}}_k - \boldsymbol{w}_k - \nabla_{\boldsymbol{w}} \mathcal{L}[\tilde{\boldsymbol{w}}_k] \Delta t + (\boldsymbol{b}_{k+1} + \sigma_{k+1} \hat{n}_{k+1}) \Delta t] \\ &= \nabla_{\boldsymbol{w}} \mathcal{L}[\boldsymbol{w}_k + \tilde{\boldsymbol{w}}_k - \boldsymbol{w}_k - \nabla_{\boldsymbol{w}} \mathcal{L}[\boldsymbol{w}_k] \Delta t + \mathcal{O}(\Delta t^2) + (\boldsymbol{b}_{k+1} + \sigma_{k+1} \hat{n}_{k+1}) \Delta t] \\ &= \nabla_{\boldsymbol{w}} \mathcal{L}[\boldsymbol{w}_{k+1} + \tilde{\boldsymbol{w}}_k - \boldsymbol{w}_k + (\boldsymbol{b}_{k+1} + \sigma_{k+1} \hat{n}_{k+1}) \Delta t] + \mathcal{O}(\Delta t^2) \end{aligned} \tag{21}$$

Now, we can simplify the expression for $\tilde{\boldsymbol{w}}_k - \boldsymbol{w}_k$ as:

$$\begin{aligned} \tilde{\boldsymbol{w}}_k - \boldsymbol{w}_k &= \boldsymbol{w}_0 + \Delta t \sum_{i=1}^{k} (-\nabla_{\boldsymbol{w}} \mathcal{L}[\tilde{\boldsymbol{w}}_{i-1}] + \boldsymbol{b}_i + \sigma_i \hat{n}_i) - \boldsymbol{w}_0 + \Delta t \sum_{i=1}^{k} \nabla_{\boldsymbol{w}} \mathcal{L}[\boldsymbol{w}_{i-1}] \\ &= -\Delta t \sum_{i=1}^{k} (\nabla_{\boldsymbol{w}} \mathcal{L}[\tilde{\boldsymbol{w}}_{i-1}] - \nabla_{\boldsymbol{w}} \mathcal{L}[\boldsymbol{w}_{i-1}]) + \Delta t \sum_{i=1}^{k} (\boldsymbol{b}_i + \sigma_i \hat{n}_i) \\ &= \Delta t \sum_{i=1}^{k} (\boldsymbol{b}_i + \sigma_i \hat{n}_i) + \mathcal{O}(\Delta t^2) \qquad \text{[Using Eq. (18)]} \end{aligned} \tag{22}$$

Plugging Eq. (22) in Eq. (21), we can show the desired result:

$$\begin{aligned}\nabla_{\boldsymbol{w}}\mathcal{L}[\tilde{\boldsymbol{w}}_{k+1}] &= \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_{k+1} + \Delta t\sum_{i=1}^{k+1}(\boldsymbol{b}_i + \sigma_i\hat{n}_i) + \mathcal{O}(\Delta t^2)] + \mathcal{O}(\Delta t^2) \\ &= \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_{k+1}] + \Delta t\left(\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_{k+1}](\sum_{i=1}^{k+1}\boldsymbol{b}_i) + \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_{k+1}](\sum_{i=1}^{k+1}\sigma_i\hat{n}_i)\right) + \mathcal{O}(\Delta t^2)\end{aligned} \tag{23}$$

Now, we know that Eq. (19) is true for $k = 1$ from Lemma A.1. Assuming that it holds for the first k steps, we will show that it holds for the $(k+1)^{th}$ step too. Using Eq. (11), we can write:

$$\begin{aligned}\mathcal{L}[\tilde{\boldsymbol{w}}_{k+1}] &= \mathcal{L}[\tilde{\boldsymbol{w}}_k] + \langle\nabla_{\boldsymbol{w}}\mathcal{L}[\tilde{\boldsymbol{w}}_k], \tilde{\boldsymbol{w}}_{k+1} - \tilde{\boldsymbol{w}}_k\rangle + \frac{1}{2}\left\langle\tilde{\boldsymbol{w}}_{k+1} - \tilde{\boldsymbol{w}}_k, \nabla^2_{\boldsymbol{w}}\mathcal{L}[\tilde{\boldsymbol{w}}_k](\tilde{\boldsymbol{w}}_{k+1} - \tilde{\boldsymbol{w}}_k)\right\rangle \\ &= \mathcal{L}[\tilde{\boldsymbol{w}}_k] + \mathcal{O}(\Delta t^3) \\ &\left\langle\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k] + \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k]\left(\sum_{i=1}^{k}(\boldsymbol{b}_i + \sigma_i\hat{n}_i)\right)\Delta t, -\nabla_{\boldsymbol{w}}\mathcal{L}[\tilde{\boldsymbol{w}}_k] + \boldsymbol{b}_{k+1} + \sigma_{k+1}\hat{n}_{k+1}\right\rangle\Delta t \\ &+ \frac{1}{2}\left\langle-\nabla_{\boldsymbol{w}}\mathcal{L}[\tilde{\boldsymbol{w}}_k] + \boldsymbol{b}_{k+1} + \sigma_k\hat{n}_{k+1}, \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k]\left(-\nabla_{\boldsymbol{w}}\mathcal{L}[\tilde{\boldsymbol{w}}_k] + \boldsymbol{b}_{k+1} + \sigma_{k+1}\hat{n}_{k+1}\right)\right\rangle\Delta t^2\end{aligned}$$

Imposing the independence of the noise statistics and ignoring higher order terms, i.e. $\mathcal{O}(\Delta t^3)$, we get to our desired result:

$$\begin{aligned}\mathbb{E}_{\hat{n}_1,\hat{n}_2\ldots\hat{n}_{k+1}}[\mathcal{L}[\tilde{\boldsymbol{w}}_{k+1}]] = \mathbb{E}_{\hat{n}_1,\hat{n}_2\ldots\hat{n}_k}&[\mathcal{L}[\tilde{\boldsymbol{w}}_k]] - \|\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k]\|^2\Delta t \\ &+ \frac{1}{2}\left\langle\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k], \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k]\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k]\right\rangle\Delta t^2 \\ &+ \langle\boldsymbol{b}_{k+1}, \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k]\rangle\Delta t + \frac{1}{2}\left\langle\boldsymbol{b}_{k+1}, \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k]\boldsymbol{b}_{k+1}\right\rangle\Delta t^2 \\ &- \frac{1}{2}\left\langle\boldsymbol{b}_{k+1}, (\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k] + \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k]^T)\nabla_{\boldsymbol{w}}\mathcal{L}[w_k]\right\rangle\Delta t^2 \\ &+ \frac{1}{2}\frac{\sigma^2_{k+1}f(N)}{N}\operatorname{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k]]\Delta t^2 \\ &+ \left\langle\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k]\left(\sum_{j=1}^{k}\boldsymbol{b}_j\right), \boldsymbol{b}_{k+1} - 2\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k]\right\rangle\Delta t^2\end{aligned} \tag{24}$$

Now, using Eq. (19) for the k^{th} step and using the relation that $\mathcal{L}[\boldsymbol{w}_{k+1}] = \mathcal{L}[\boldsymbol{w}_k] - \|\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k]\|^2\Delta t + \frac{1}{2}\left\langle\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k], \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k]\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k]\right\rangle\Delta t^2$, we get the desired relation for the $(k+1)^{th}$ step. □

A qualitative description of the effect of bias: The impact of bias across iterative optimization can be thought of in three key scenarios: (1) the network converges to the same loss basin as gradient descent but the bias alters the specific steady state; (2) bias alters the network training trajectory to the extent that it converges in a loss basin that is different from what gradient descent would converge to; (3) the overall effect of bias hinders network training to the extent it becomes unstable and performance collapses.

In the first case, we can suppose that either the bias is relatively small, or, generally speaking, there is not a strong correlation in the degree of misalignment between the bias vectors and the gradient and Hessian over K steps. In this case, the bias will have an impact that is similar to that of the isotropic noise, and the ultimate result can be understood from a steady state perspective. Notably, at the steady state, the bias would balance the gradient, thereby causing the expected weight change to be 0. Assuming a second order approximation of the loss basin, the bias in the gradients would imply an excess loss of $\frac{1}{2}\boldsymbol{b}_i^T H\boldsymbol{b}_i$ as compared to the minima of the loss basin, which is where gradient descent would have converged to. This would result in minor degradation in performance, as shown in Fig. 1 for lower bias values.

The second case is complicated and ultimately requires further learning rule specific assumptions. Given that our goal in this paper is to remain learning rule agnostic, we can only say that this case would occur when the bias vectors have a semi-structured relationship to the gradient and the Hessian that pushes the weight updates in very different trajectories compared to gradient descent.

The third case would result from either excessive bias or a very strong relationship between the K bias vectors and their product with the gradient or the Hessian. Thus, we can say that learning rules with very strong or highly systematic bias may not converge.

Additional assumption on loss function in Theorems A.3 - A.5: Below, we prove the relationship between network width, depth or non-linearity and the impact of noise on learning for the mean squared error loss. Although we do not prove the result for general loss functions, we believe that our proofs present an outline that other researchers can leverage in future work to extend our results to other loss functions, based on their specific application, and thereby extend our results to guide their network design choices.

Theorem A.3. ***[$f(N) = \mathcal{O}(1)$ case] Increasing depth lowers impact of variance*** *For linear feedforward ANNs, the impact of variance on $\Delta\mathcal{L}(0, \sigma^2)$, for a $(L+1)$ layer network is less than that of a L layer network, i.e.*

$$\Delta\mathcal{L}(0, \sigma^2)_{(L+1)-layer} \leq \Delta\mathcal{L}(0, \sigma^2)_{L-layer} \tag{25}$$

where each layer has d units and weights initialized by the Xavier initialization strategy and the total variance in gradient estimates does not grow with the size of the network, i.e. $f(N)$ is some constant

Proof. We will first show the theorem holds for $L = 1$, i.e. the impact of variance on a single layer neural network is more than that of a two layer neural network. For the sake of simplicity, we assume $\mathcal{L}$ to be the standard mean squared error loss, i.e. $\mathcal{L} = \frac{1}{|\mathbf{D}|}\sum_{n=1}^{|\mathbf{D}|}(y_n - \hat{y}_n)^2$, where $\hat{y}_n$ denotes the output of the neural network and $\mathbf{D} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), (\boldsymbol{x}_2, \boldsymbol{y}_2) \ldots (\boldsymbol{x}_n, \boldsymbol{y}_n)\}$ denotes the training dataset. Note that for mean squared error, $\nabla^2_{\hat{y}_n}\mathcal{L} = \frac{2}{|\mathbf{D}|}$. We assume that there is no bias term in determining the unit activations.

Let us denote the weight matrix of a single layer network as $\boldsymbol{W} \in \mathbf{R}^d$ such that $\hat{y}_n = \sum_j W_j x_{nj}$. Thus, we can use the chain rule of differentiation to infer:

$$\begin{aligned}
\nabla_{W_k}\mathcal{L} &= \sum_n (\nabla_{\hat{y}_n}\mathcal{L}) x_{nk} \\
\nabla^2_{W_k}\mathcal{L} &= \sum_n (\nabla^2_{\hat{y}_n}\mathcal{L}) x^2_{nk} \\
\text{Hence,} \quad \text{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}] &= \sum_k \nabla^2_{W_k}\mathcal{L} = \frac{2}{|\mathbf{D}|}\sum_{n,k} x^2_{nk}
\end{aligned} \tag{26}$$

Let us assume that x is i.i.d. and normalized to have zero mean and unit variance in all dimensions, i.e. $\frac{1}{|\mathbf{D}|}\sum_n x^2_{ni} = 1$. Therefore, for a $L = 1$ network: $\text{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}] = 2d$. Let us denote the weight matrices of a two layer network as $W^{(1)} \in \mathbf{R}^{d\times d}$ and $W^{(2)} \in \mathbf{R}^d$ such that $\hat{y}_n = \sum_{i,j} W^{(2)}_i W^{(1)}_{ij} x_{nj}$. Again, using the chain rule:

$$
\begin{aligned}
\nabla_{W_k^{(2)}}\mathcal{L} &= \sum_n (\nabla_{\hat{y}_n}\mathcal{L})(\sum_j W_{kj}^{(1)} x_{nj}) \\
\nabla^2_{W_k^{(2)}}\mathcal{L} &= \sum_n (\nabla^2_{\hat{y}_n}\mathcal{L})(\sum_j W_{kj}^{(1)} x_{nj})^2 \\
\nabla_{W_{kl}^{(1)}}\mathcal{L} &= \sum_n (\nabla_{\hat{y}_n}\mathcal{L}) W_k^{(2)} x_{nl} \\
\nabla^2_{W_{kl}^{(1)}}\mathcal{L} &= \sum_n (\nabla^2_{\hat{y}_n}\mathcal{L}) (W_k^{(2)} x_{nl})^2 \\
\text{Hence,}\quad \mathrm{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}] &= \sum_k \nabla^2_{W_k^{(2)}}\mathcal{L} + \sum_{k,l} \nabla^2_{W_{kl}^{(1)}}\mathcal{L} \\
&= \sum_n (\nabla^2_{\hat{y}_n}\mathcal{L}) \left[\sum_k (\sum_j W_{kj}^{(1)} x_{nj})^2 + \sum_{k,l} (W_k^{(2)} x_{nl})^2 \right] \\
&= \frac{2}{|\mathbf{D}|} \sum_n \left[\sum_k (\sum_j W_{kj}^{(1)} x_{nj})^2 + \sum_k (W_k^{(2)})^2 \sum_l (x_{nl})^2 \right] \\
&= \frac{\sum_{n,k}(\sum_j W_{kj}^{(1)} x_{nj})^2}{\sum_{n,l} x_{nl}^2} \frac{2}{|\mathbf{D}|} \sum_{n,l} x_{nl}^2 + \frac{2}{|\mathbf{D}|} \sum_k (W_k^{(2)})^2 \sum_{n,l} (x_{nl})^2 \\
&= (\rho + \sum_k (W_k^{(2)})^2)\, \mathrm{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}]_{1-layer} \qquad \text{[From Eq. (26)]} \qquad (27) \\
\text{where,}\quad \rho &= \frac{\sum_{n,k}(\sum_j W_{kj}^{(1)} x_{nj})^2}{\sum_{n,l} x_{nl}^2}
\end{aligned}
$$

Now, we assume that all weights are initialized according to the standard Xavier initialization strategy, i.e. $\mathbf{E}_{\boldsymbol{W}}[W_{jk}^{(1)}] = \mathbf{E}_{\boldsymbol{W}}[W_k^{(2)}] = 0$ and $\mathbf{E}_{\boldsymbol{W}}[(W_{jk}^{(1)})^2] = \mathbf{E}_{\boldsymbol{W}}[(W_k^{(2)})^2] = \frac{1}{d}$. Thus, $\sum_k (W_k^{(2)})^2 \approx 1$ where the approximation is stronger for larger values of d. Similarly, $\sum_{j,k}(W_{jk}^{(1)})^2 \approx d$. Now, we can use the dot product inequality to simplify ρ:

$$
\begin{aligned}
\rho = \frac{\sum_{n,k}(\sum_j W_{kj}^{(1)} x_{nj})^2}{\sum_{n,l} x_{nl}^2} &\leq \frac{\sum_{n,k}(\sum_j (W_{kj}^{(1)})^2 \sum_j (x_{nj})^2}{\sum_{n,l} x_{nl}^2} \\
\implies \rho &\leq \sum_{j,k} (W_{kj}^{(1)})^2 \approx d \\
\implies \rho + \sum_k (W_k^{(2)})^2 &\leq d + 1
\end{aligned}
$$

$$
\begin{aligned}
\text{Therefore, from Eq. (27)}\qquad \mathrm{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}]_{2-layer} &= (\rho + \sum_k (W_k^{(2)})^2)\, \mathrm{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}]_{1-layer} \\
&\leq (1 + d)\, \mathrm{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}]_{1-layer} \qquad (28)
\end{aligned}
$$

Also, note that by applying the above relations, we can write for a $L = 2$ network: $\mathrm{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}] = 2d(\rho + 1) \leq 2d(d + 1)$. Now, the impact of variance, $\Delta\mathcal{L}(0, \sigma^2)$, on a 2-layer network depends on the term $\frac{1}{N}\mathrm{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}]_{2-layer}$, where $N = d + d^2$ is the total number of parameters in the network. Similarly, the impact of variance on a 1-layer network depends on the term $\frac{1}{N}\mathrm{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}]_{1-layer}$, where $N = d$. Therefore, extending inequality 28:

$$
\begin{aligned}
\Delta\mathcal{L}(0,\sigma^2)_{2-layer} &= \frac{1}{d+d^2}\operatorname{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}]_{2-layer} \leq \frac{1+d}{d+d^2}\operatorname{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}]_{1-layer} \\
&\implies \Delta\mathcal{L}(0,\sigma^2)_{2-layer} \leq \frac{1}{d}\operatorname{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}]_{1-layer} = \Delta\mathcal{L}(0,\sigma^2)_{1-layer} \\
\text{Hence,} \quad &\Delta\mathcal{L}(0,\sigma^2)_{2-layer} \leq \Delta\mathcal{L}(0,\sigma^2)_{1-layer} \qquad (29)
\end{aligned}
$$

We can extend this result to compare L layer networks to $(L+1)$ layer networks. Notably, a L layer network can be thought of as a single-layer network operating on a $(L-1)$ layer network and a $(L+1)$ layer network as a two-layer network operating on an equivalent $(L-1)$ layer network. Using the above result, we can therefore conclude that $\Delta\mathcal{L}(0,\sigma^2)_{(L+1)-layer} \leq \Delta\mathcal{L}(0,\sigma^2)_{L-layer}$. □

Theorem A.4. ***[$f(N) = \mathcal{O}(N)$ case] Increasing width over depth lowers impact of variance*** *For linear feedforward ANNs, the impact of variance on $\Delta\mathcal{L}(0,\sigma^2)$, for a $(L+1)$ layer network with d' units in each hidden layer is more than that of a L layer network with same number of parameters, N when $f(N)$ grows linearly with number of parameters if $d' \leq d^2$, where d is the input dimensionality.*

Proof. When $f(N) = \mathcal{O}(N)$, the impact of variance of $\Delta\mathcal{L}(0,\sigma^2)$ can be written as:

$$\Delta\mathcal{L}(0,\sigma^2) = \frac{1}{2}\sigma^2 \operatorname{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}]\Delta t^2$$

and therefore, only depends on the trace. Similar to the proof for Theorem A.3, we will first show the result holds for $L = 2$ and extend it for any L. As before, we assume that x is i.i.d. and normalized to have zero mean and unit variance in all dimensions, i.e. $\frac{1}{|\mathbf{D}|}\sum_n x_{ni}^2 = 1$. Furthermore, we assume that the weights in all layers are initialized to maintain this zero-mean and unit-variance property in each hidden unit, i.e. $\frac{1}{|\mathbf{D}|}\sum_n h_{ni}^{(l)} = 0$ and $\frac{1}{|\mathbf{D}|}\sum_n \left(h_{ni}^{(l)}\right)^2 = 1$. This property imposes certain constraints on the weight matrices as show below:

$$
\begin{aligned}
\frac{1}{|\mathbf{D}|}\sum_n \left(h_{ni}^{(l)}\right)^2 = 1 = \frac{1}{|\mathbf{D}|}\sum_n \left(\sum_j W_{ij}^{(l)} h_{nj}^{(l-1)}\right)^2 &= \frac{1}{|\mathbf{D}|}\sum_n \sum_{jk} W_{ij}^{(l)} W_{ik}^{(l)} h_{nj}^{(l-1)} h_{nk}^{(l-1)} \\
&= \sum_{jk} W_{ij}^{(l)} W_{ik}^{(l)} \left[\frac{1}{|\mathbf{D}|}\sum_n h_{nj}^{(l-1)} h_{nk}^{(l-1)}\right] \\
\implies 1 = \sum_j \left(W_{ij}^{(l)}\right)^2 \qquad &[\text{i.i.d. assumption on } h^{(l-1)}] \qquad (30)
\end{aligned}
$$

Therefore, $\sum_{ij}\left(W_{ij}^{(l)}\right)^2 = \sum_i 1 = d^{(l)}$, where $d^{(l)}$ is the output dimensionality of layer l.

Using these relations and the results from the proof of Theorem A.3, we can write the Trace for a 2-layer network with input dimensionality as d, hidden layer dimensionality as $\hat{d}$ and output dimensionality 1 as:

$$
\begin{aligned}
\operatorname{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}]_{2-layer} &= \frac{1}{|\mathbf{D}|}\sum_n \left[\sum_i \left(h_{ni}^{(1)}\right)^2 + \sum_{ij}\left(W_i^{(2)}\right)^2 x_{nj}^2\right] \\
&= \sum_{i=1}^{\hat{d}} \left(\frac{1}{|\mathbf{D}|}\sum_n \left(h_{ni}^{(1)}\right)^2\right) + \left[\sum_{i=1}^{\hat{d}}\left(W_i^{(2)}\right)^2\right]\left[\sum_{j=1}^{d}\left(\frac{1}{|\mathbf{D}|}\sum_n x_{nj}^2\right)\right] \\
&= \hat{d} + d \qquad (31)
\end{aligned}
$$

Similarly, the Trace of a 3-layer network with input dimensionality as d, hidden layer dimensionality as d', with weights of each layer being i.i.d. with respect to other layers, and output dimensionality as 1 as:

$$\begin{aligned}\text{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}]_{3-layer} &= \frac{1}{|\mathbf{D}|}\sum_n \left[\sum_i \left(h^{(2)}_{ni}\right)^2 + \sum_{ij}\left(W^{(3)}_i\right)^2\left(h^{(1)}_{nj}\right)^2 + \sum_k\sum_j\left(\sum_i W^{(3)}_i W^{(2)}_{ij}\right)^2 x^2_{nk}\right] \\ &= d' + d' + \left[\sum_i \left(W^{(3)}_i\right)^2\right]\left[\sum_{ij}\left(W^{(2)}_{ij}\right)^2\right]\left[\frac{1}{|\mathbf{D}|}\sum_{nk} x^2_{nk}\right] \qquad \text{[i.i.d. weights]} \\ &= 2d' + dd' \qquad (32)\end{aligned}$$

Now, number of parameters in the 2-layer network are: $(d \times \hat{d}) + (\hat{d} \times 1) = \hat{d}(d+1)$; and number of parameters in the 3-layer network are: $(d \times d') + (d' \times d') + (d' \times 1) = dd' + (d')^2 + d'$. For equal number of parameters in both networks, $\hat{d} = d' + \frac{(d')^2}{d+1}$. Therefore, the trace of the 2-layer network can be equivalently written as: $\text{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}]_{2-layer} = d' + \frac{(d')^2}{d+1} + d$. Solving for the condition when the wider 2-layer network has a lower trace than the 3-layer trace, we get to the desired conditions, i.e. $d' \le d^2$. Moreover, if the networks are extremely wide, i.e. in the regime where the hidden layer dimensionality d' is greater than d^2, then the trace of the deeper network will be lower. □

Theorem A.5. ***ReLU lowers impact of variance*** *Let $\phi(.)$ denote a threshold non-linearity function. The impact of variance on $\Delta\mathcal{L}$ for a network with such non-linearities is less than that of an equivalent linear network, i.e.*

$$\Delta\mathcal{L}(0,\sigma^2)_\phi \le \Delta\mathcal{L}(0,\sigma^2)_{Linear} \qquad (33)$$

where the gain of $\phi(.)$ is less than or equal to 1 (e.g. ReLU).

Proof. We will show this result for a two layer neural network but the result holds for other cases too. Let $W^{(1)} \in \mathbf{R}^{d\times d}$ and $W^{(2)} \in \mathbf{R}^d$ denote the weight matrices for the first and second layer respectively. As above, we assume for sake of simplicity that there is no bias term in determining the network activations. Therefore, the output of the network can be expressed as: $\hat{y}_n = \sum_{i,j} W^{(2)}_i W^{(1)}_{ij} x_{nj}$ for a linear network; and $\hat{y}_n = \sum_i W^{(2)}_i \phi\left(\sum_j W^{(1)}_{ij} x_{nj}\right)$ for a network with the aforementioned threshold non-linearity. We used the chain rule to derive an expression for the trace of the hessian for a two-layer linear network in Eq. (27). Similarly, using the chain rule we can compute the trace of Hessian for the non-linear networks:

$$\begin{aligned}\nabla_{W^{(2)}_k}\mathcal{L} &= \sum_n (\nabla_{\hat{y}_n}\mathcal{L})\phi\left(\sum_j W^{(1)}_{kj} x_{nj}\right) \\ \nabla^2_{W^{(2)}_k}\mathcal{L} &= \sum_n (\nabla^2_{\hat{y}_n}\mathcal{L})\phi\left(\sum_j W^{(1)}_{kj} x_{nj}\right)^2 \\ \nabla_{W^{(1)}_{kl}}\mathcal{L} &= \sum_n (\nabla_{\hat{y}_n}\mathcal{L}) W^{(2)}_k \phi'\left(\sum_j W^{(1)}_{kj} x_{nj}\right) x_{nl} \\ \nabla^2_{W^{(1)}_{kl}}\mathcal{L} &= \sum_n (\nabla^2_{\hat{y}_n}\mathcal{L})\left[W^{(2)}_k \phi'\left(\sum_j W^{(1)}_{kj} x_{nj}\right) x_{nl}\right]^2 \qquad \text{[Assuming } \phi''(.) \approx 0\text{]} \\ \text{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}]_\phi &= \sum_k \nabla^2_{W^{(2)}_k}\mathcal{L} + \sum_{k,l}\nabla^2_{W^{(1)}_{kl}}\mathcal{L} \\ &= \sum_n (\nabla^2_{\hat{y}_n}\mathcal{L})\left[\sum_k \phi\left(\sum_j W^{(1)}_{kj} x_{nj}\right)^2 + \sum_{k,l}\left[W^{(2)}_k \phi'\left(\sum_j W^{(1)}_{kj} x_{nj}\right) x_{nl}\right]^2\right] \qquad (34)\end{aligned}$$

For a threshold non-linearity with gain ≤ 1, $\phi(z) \leq z$ and $\phi'(z) \leq 1 \; \forall z \in \mathbf{R}$. Using these two relations, we can further simplify Eq. (34):

$$\text{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}]_\phi \leq \sum_n (\nabla^2_{\hat{y}_n}\mathcal{L}) \left[\sum_k \left(\sum_j W^{(1)}_{kj} x_{nj} \right)^2 + \sum_{k,l} \left(W^{(2)}_k x_{nl} \right)^2 \right]$$

Plugging in the expression of Trace from Eq. (27):

$$\begin{aligned} &\text{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}]_\phi \leq \text{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}]_{Linear} \\ \implies &\Delta\mathcal{L}(0, \sigma^2)_\phi \leq \Delta\mathcal{L}(0, \sigma^2)_{Linear} \end{aligned} \tag{35}$$

The above relation can be easily extended to other network architectures to complete the proof. □

Note on commonly used non-linear activation functions: By construction, ReLU fits the definition of ϕ and therefore, the above theorem holds. However, the conditions that $\phi(z) \leq z$ and $\phi'(z) \leq 1$ are not satisfied by Sigmoid non-linearity. Due to its non-zero offset, i.e. for zero input the activation function returns 0.5, the first condition ($\phi(z) \leq z$) is not satisfied by the standard Sigmoid function.

Theorem A.6. *The impact of bias on $\Delta\mathcal{L}$ grows linearly with the norm of the gradient.*

$$\Delta\mathcal{L}(b, 0) = \frac{1}{2} b^2 Q \Delta t^2 + b \|\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\| P \Delta t \tag{36}$$

where P, Q are terms that are independent of the norm of the gradient, $\|\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\|$.

Proof. Since the impact of bias and variance are decoupled, we can study each of their impacts independently. Therefore, we look at the expression in Lemma A.1 for some bias value b and variance, $\sigma = 0$.

$$\begin{aligned} \mathbb{E}_{\hat{n}}\left[\Delta\mathcal{L}(b, 0)\right] &= b \left\langle \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)], \overrightarrow{\boldsymbol{\beta}} \right\rangle \Delta t + \frac{1}{2} b^2 \left\langle \overrightarrow{\boldsymbol{\beta}}, \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)] \overrightarrow{\boldsymbol{\beta}} \right\rangle \Delta t^2 \\ &\quad - \frac{1}{2} b \left\langle \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)], (\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)] + \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]^T) \overrightarrow{\boldsymbol{\beta}} \right\rangle \Delta t^2 \\ &= \frac{1}{2} b^2 \left\langle \overrightarrow{\boldsymbol{\beta}}, \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)] \overrightarrow{\boldsymbol{\beta}} \right\rangle \Delta t^2 + b \|\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\| P \Delta t \end{aligned} \tag{37}$$

(38)

where, $P = \left\langle \widehat{\nabla_{\boldsymbol{w}}\mathcal{L}}, \overrightarrow{\boldsymbol{\beta}} \right\rangle - \frac{1}{2} \left\langle \widehat{\nabla_{\boldsymbol{w}}\mathcal{L}}, (\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)] + \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]^T) \overrightarrow{\boldsymbol{\beta}} \right\rangle \Delta t$ and $\widehat{\nabla_{\boldsymbol{w}}\mathcal{L}}$ denotes the unit vector along the gradient. It is clear that P is independent of the norm of the gradient and only depends on the direction of the bias vector, the gradient of the loss function and the eigenvectors of the loss Hessian. □

Theorem A.7. ***Noise helps avoid sharp minima*** *The sufficient condition under which noise prevents the system from descending the loss landscape, i.e. $\mathcal{L}[\boldsymbol{w}(t + \Delta t)] \geq \mathcal{L}[\boldsymbol{w}(t)]$, is*

$$\lambda_1 \geq \lambda_N \geq \frac{2}{\Delta t} \frac{1}{1 + \left(\frac{\sigma}{\|\nabla_{\boldsymbol{w}}\mathcal{L}\|} \right)^2} \tag{39}$$

where $\lambda_1 \geq \lambda_2 ... \geq \lambda_N$ denote the eigenvalues of $\nabla^2_{\boldsymbol{w}}\mathcal{L}$ around the minima.

Proof. For sake of brevity, we assume that the bias in gradient estimates is 0 and focus only on the effect of variance. We start the proof by recalling the Taylor series expansion from Eq. (11). It is clear that while following the true gradient:

$$[\mathcal{L}[\boldsymbol{w}(t + \Delta t)] - \mathcal{L}[\boldsymbol{w}(t)]]_{(bias=0, var=0)} = -\|\nabla_{\boldsymbol{w}}\mathcal{L}\|^2 \Delta t + \frac{1}{2} \left\langle \nabla_{\boldsymbol{w}}\mathcal{L}, \nabla^2_{\boldsymbol{w}}\mathcal{L} \nabla_{\boldsymbol{w}}\mathcal{L} \right\rangle \Delta t^2 \tag{40}$$

Using Lemma A.1 and Eq. (40), we can write the expected decrease in $\mathcal{L}$ in one weight update step (with 0 bias and σ^2 variance) as:

$$[\mathcal{L}[\boldsymbol{w}(t + \Delta t)] - \mathcal{L}[\boldsymbol{w}(t)]]_{(0, \sigma^2)} = -\|\nabla_{\boldsymbol{w}}\mathcal{L}\|^2 \Delta t + \frac{1}{2} \left(\left\langle \nabla_{\boldsymbol{w}}\mathcal{L}, \nabla^2_{\boldsymbol{w}}\mathcal{L} \nabla_{\boldsymbol{w}}\mathcal{L} \right\rangle + \frac{\sigma^2}{N} \text{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}] \right) \Delta t^2 \tag{41}$$

We note the following inequalities from linear algebra which will be useful in proving the result:

$$\begin{aligned}\lambda_N \|\nabla_{\boldsymbol{w}}\mathcal{L}\|^2 \leq \left\langle \nabla_{\boldsymbol{w}}\mathcal{L}, \nabla^2_{\boldsymbol{w}}\mathcal{L}\nabla_{\boldsymbol{w}}\mathcal{L}\right\rangle \leq \lambda_1 \|\nabla_{\boldsymbol{w}}\mathcal{L}\|^2 \\ \lambda_N \leq \quad \frac{1}{N}\operatorname{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}] \quad \leq \lambda_1\end{aligned} \tag{42}$$

Using the relations in ineq. 42, we can lower bound the change in $\mathcal{L}$ from Eq. (41):

$$\mathcal{L}[\boldsymbol{w}(t+\Delta t)] - \mathcal{L}[\boldsymbol{w}(t)] \geq -\|\nabla_{\boldsymbol{w}}\mathcal{L}\|^2 \Delta t + \frac{1}{2}\left(\lambda_N \|\nabla_{\boldsymbol{w}}\mathcal{L}\|^2 + \sigma^2 \lambda_N\right)\Delta t^2 \tag{43}$$

In order to understand the conditions under which a weight update step causes an increase in $\mathcal{L}$, we focus on the conditions when $\mathcal{L}[\boldsymbol{w}(t+\Delta t)] - \mathcal{L}[\boldsymbol{w}(t)] \geq 0$. As such, if the RHS of inequality 43 is greater than 0, then the weight update step necessarily implies $\mathcal{L}[\boldsymbol{w}(t+\Delta t)] \geq \mathcal{L}[\boldsymbol{w}(t)]$. Therefore,

$$\begin{aligned}-\|\nabla_{\boldsymbol{w}}\mathcal{L}\|^2 \Delta t + \frac{1}{2}\left(\lambda_N \|\nabla_{\boldsymbol{w}}\mathcal{L}\|^2 + \sigma^2 \lambda_N\right)\Delta t^2 \geq 0 \\ \lambda_N \geq \frac{2}{\Delta t}\frac{1}{1+\left(\frac{\sigma}{\|\nabla_{\boldsymbol{w}}\mathcal{L}\|}\right)^2}\end{aligned} \tag{44}$$

Similarly, we can upper bound $\mathcal{L}[\boldsymbol{w}(t+\Delta t)] - \mathcal{L}[\boldsymbol{w}(t)]$ and set the upper bound to be less than or equal to 0 in order to study the sufficient conditions under which the weight update step leads to decrease in the loss function.

$$\begin{aligned}\mathcal{L}[\boldsymbol{w}(t+\Delta t)] - \mathcal{L}[\boldsymbol{w}(t)] \leq -\|\nabla_{\boldsymbol{w}}\mathcal{L}\|^2 \Delta t + \frac{1}{2}\left(\lambda_1 \|\nabla_{\boldsymbol{w}}\mathcal{L}\|^2 + \sigma^2 \lambda_1\right)\Delta t^2 \leq 0 \\ \implies \lambda_1 \leq \frac{2}{\Delta t}\frac{1}{1+\left(\frac{\sigma}{\|\nabla_{\boldsymbol{w}}\mathcal{L}\|}\right)^2}\end{aligned} \tag{45}$$

□

Theorem A.8. ***Bias prevents descent into local minima dependent on Hessian spectrum.*** *Bias in gradient estimates could prevent converging to a minima. Specifically, the sufficient condition for the weight updates to produce a decrease in $\mathcal{L}$ when using a biased estimate of the gradient is:*

$$\frac{b}{\|\nabla_{\boldsymbol{w}}\mathcal{L}\|} \leq \frac{1}{\sqrt{N}}\frac{2}{1+\psi\Delta t + \sqrt{1+4\psi\Delta t + \psi^2\Delta t^2}} \approx \frac{1}{\sqrt{N}}\frac{1}{(1+\frac{3}{2}\psi\Delta t)} \tag{46}$$

where $\psi^2 = \sum_i \lambda_i^2$, i.e. the Frobenius norm of the loss Hessian, and $\lambda_1 \geq \lambda_2 ... \geq \lambda_N$ denote the eigenvalues of $\nabla^2_{\boldsymbol{w}}\mathcal{L}$ around the minima and $\nabla^2_{\boldsymbol{w}}\mathcal{L}$ is a normal matrix.

Proof. For sake of brevity, we assume that the noise in gradient estimates is 0 and focus only on the effect of bias. We start the proof by combining Lemma A.1 and Eq. (40) to get the expected weight decrease in $\mathcal{L}$ in one weight update step (with bias=b and no variance):

$$\begin{aligned}\mathcal{L}[\boldsymbol{w}(t+\Delta t)] - \mathcal{L}[\boldsymbol{w}(t)] = -\|\nabla_{\boldsymbol{w}}\mathcal{L}\|^2 \Delta t + \frac{1}{2}\left\langle \nabla_{\boldsymbol{w}}\mathcal{L}, \nabla^2_{\boldsymbol{w}}\mathcal{L}\nabla_{\boldsymbol{w}}\mathcal{L}\right\rangle \Delta t^2 + b\left\langle \nabla_{\boldsymbol{w}}\mathcal{L}, \overrightarrow{\boldsymbol{\beta}}\right\rangle \Delta t \\ + \frac{1}{2}b^2\left\langle \overrightarrow{\boldsymbol{\beta}}, \nabla^2_{\boldsymbol{w}}\mathcal{L}\overrightarrow{\boldsymbol{\beta}}\right\rangle \Delta t^2 - \frac{1}{2}b\left\langle \nabla_{\boldsymbol{w}}\mathcal{L}, (\nabla^2_{\boldsymbol{w}}\mathcal{L} + \nabla^2_{\boldsymbol{w}}\mathcal{L}^T)\overrightarrow{\boldsymbol{\beta}}\right\rangle \Delta t^2\end{aligned} \tag{47}$$

Furthermore, we denote the eigenvalues of the Loss Hessian, i.e. $\nabla^2_{\boldsymbol{w}}\mathcal{L}$, around the minima as $\lambda_1 \geq \lambda_2 ... \geq \lambda_N$ and the corresponding eigenvectors as $\boldsymbol{\nu}_1, \boldsymbol{\nu}_2 ... \boldsymbol{\nu}_N$. Assuming $\nabla^2_{\boldsymbol{w}}\mathcal{L}$ is normal, we can express $\nabla_{\boldsymbol{w}}\mathcal{L}$ and $\overrightarrow{\boldsymbol{\beta}}$ in the basis space of $\{\boldsymbol{\nu}_1, \boldsymbol{\nu}_2 ... \boldsymbol{\nu}_N\}$, i.e. $\nabla_{\boldsymbol{w}}\mathcal{L} = \sum_i \alpha_i \boldsymbol{\nu}_i$ and

$\overrightarrow{\boldsymbol{\beta}} = \sum_i \beta_i \boldsymbol{\nu}_i$. Using standard results from linear algebra, we can write the following expressions:

$$\begin{aligned}
\left\langle \nabla_{\boldsymbol{w}}\mathcal{L}, \nabla^2_{\boldsymbol{w}}\mathcal{L}\nabla_{\boldsymbol{w}}\mathcal{L}\right\rangle &= \sum_i \lambda_i\alpha_i^2 \\
\left\langle \nabla_{\boldsymbol{w}}\mathcal{L}, \overrightarrow{\boldsymbol{\beta}}\right\rangle &= \sum_i \alpha_i\beta_i \\
\left\langle \overrightarrow{\boldsymbol{\beta}}, \nabla^2_{\boldsymbol{w}}\mathcal{L}\overrightarrow{\boldsymbol{\beta}}\right\rangle &= \sum_i \lambda_i\beta_i^2 \\
\left\langle \nabla_{\boldsymbol{w}}\mathcal{L}, (\nabla^2_{\boldsymbol{w}}\mathcal{L} + \nabla^2_{\boldsymbol{w}}\mathcal{L}^T)\overrightarrow{\boldsymbol{\beta}}\right\rangle &= \left\langle \nabla_{\boldsymbol{w}}\mathcal{L}, \nabla^2_{\boldsymbol{w}}\mathcal{L}\overrightarrow{\boldsymbol{\beta}}\right\rangle + \left\langle \nabla_{\boldsymbol{w}}\mathcal{L}, \nabla^2_{\boldsymbol{w}}\mathcal{L}^T\overrightarrow{\boldsymbol{\beta}}\right\rangle \\
&= \left\langle \nabla_{\boldsymbol{w}}\mathcal{L}, \nabla^2_{\boldsymbol{w}}\mathcal{L}\overrightarrow{\boldsymbol{\beta}}\right\rangle + \nabla_{\boldsymbol{w}}\mathcal{L}^T\nabla^2_{\boldsymbol{w}}\mathcal{L}^T\overrightarrow{\boldsymbol{\beta}} \\
&= \left\langle \nabla_{\boldsymbol{w}}\mathcal{L}, \nabla^2_{\boldsymbol{w}}\mathcal{L}\overrightarrow{\boldsymbol{\beta}}\right\rangle + \left\langle \nabla^2_{\boldsymbol{w}}\mathcal{L}\nabla_{\boldsymbol{w}}\mathcal{L}, \overrightarrow{\boldsymbol{\beta}}\right\rangle \\
&= 2\sum_i \lambda_i\alpha_i\beta_i
\end{aligned} \tag{48}$$

Plugging in the relations from Eq. (48) into Eq. (47), we get:

$$\begin{aligned}
\mathcal{L}[\boldsymbol{w}(t+\Delta t)] - \mathcal{L}[\boldsymbol{w}(t)] &= -\|\nabla_{\boldsymbol{w}}\mathcal{L}\|^2\Delta t + \frac{1}{2}\sum_i \lambda_i\alpha_i^2\Delta t^2 + b\sum_i \alpha_i\beta_i\Delta t \\
&\quad + \frac{1}{2}b^2\sum_i \lambda_i\beta_i^2\Delta t^2 - b\sum_i \lambda_i\alpha_i\beta_i\Delta t^2 \\
&= -\|\nabla_{\boldsymbol{w}}\mathcal{L}\|^2\Delta t + \frac{\Delta t}{2}\sum_i \left(\lambda_i\alpha_i^2\Delta t + 2b\alpha_i\beta_i(1-\lambda_i\Delta t) + \lambda_i b^2\Delta t\beta_i^2\right)
\end{aligned} \tag{49}$$

We can factorize the quadratic in the second summation term as follows:

$$\lambda_i\alpha_i^2\Delta t + 2b\alpha_i\beta_i(1-\lambda_i\Delta t) + \lambda_i b^2\Delta t\beta_i^2 = \alpha_i^2\left(\lambda_i\Delta t + 2\gamma_i(1-\lambda_i\Delta t) + \lambda_i\Delta t\gamma_i^2\right)$$

$$\text{where,} \quad \gamma_i = \frac{b\beta_i}{\alpha_i}$$

$$\text{The roots of the quadratic are:} \quad \frac{-(1-\lambda_i\Delta t) \pm \sqrt{1-2\lambda_i\Delta t}}{\lambda_i\Delta t}$$

Assuming small lr, i.e. $\lambda_i\Delta t << 1$ and using the relation: $(1+x)^n \approx 1 + nx \quad \text{when } |x| << 1$

$$\text{The roots are:} \quad \approx \frac{-(1-\lambda_i\Delta t) \pm (1-\frac{1}{2}2\lambda_i\Delta t)}{\lambda_i\Delta t}$$

$$\text{Therefore, the roots are:} \quad 0, \quad -2\frac{1-\lambda_i\Delta t}{\lambda_i\Delta t}$$

$$\begin{aligned}
\text{So,} \quad \lambda_i\alpha_i^2\Delta t + 2b\alpha_i\beta_i(1-\lambda_i\Delta t) + \lambda_i b^2\Delta t\beta_i^2 &= \alpha_i^2\lambda_i\Delta t\gamma_i\left(\gamma_i + 2\frac{1-\lambda_i\Delta t}{\lambda_i\Delta t}\right) \\
&= b\beta_i(b\beta_i\lambda_i\Delta t + 2\alpha_i(1-\lambda_i\Delta t))
\end{aligned} \tag{50}$$

We can further impose the dot product inequality to bound the summation term in Eq. (49):

$$\begin{aligned}
\sum_i \lambda_i\alpha_i^2\Delta t + 2b\alpha_i\beta_i(1-\lambda_i\Delta t) + \lambda_i b^2\Delta t\beta_i^2 &= \sum_i \left(b\beta_i(b\beta_i\lambda_i\Delta t + 2\alpha_i(1-\lambda_i\Delta t))\right) \\
&\le \sqrt{\sum_i b^2\beta_i^2}\sqrt{\sum_i (b\beta_i\lambda_i\Delta t + 2\alpha_i(1-\lambda_i\Delta t))^2}
\end{aligned}$$

$$\text{Using the norm relation:} \quad \|\overrightarrow{\boldsymbol{\beta}}\|^2 = \sum_i \beta_i^2 = N,$$

$$\sum_i \lambda_i\alpha_i^2\Delta t + 2b\alpha_i\beta_i(1-\lambda_i\Delta t) + \lambda_i b^2\Delta t\beta_i^2 \le b\sqrt{N}\|b\Delta t\zeta_1 + 2\zeta_2 - 2\Delta t\zeta_3\| \tag{51}$$

where $\zeta_1 = [\lambda_1\beta_1 \quad \lambda_2\beta_2 \quad ... \quad \lambda_i\beta_i \quad ...]^T$, $\zeta_2 = [\alpha_1 \quad \alpha_2 \quad ... \quad \alpha_i \quad ...]^T$ and $\zeta_3 = [\lambda_1\alpha_1 \quad \lambda_2\alpha_2 \quad ... \quad \lambda_i\alpha_i \quad ...]^T$. We can bound the norms of each of these vectors to further bound the RHS of inequality 51:

$$\begin{aligned}
\|\zeta_1\| &= \sqrt{\sum_i \lambda_i^2\beta_i^2} \leq \sqrt{\sum_i \lambda_i^2}\sqrt{\sum_i \beta_i^2} = \psi\sqrt{N} \\
\|\zeta_2\| &= \sqrt{\sum_i \alpha_i^2} = \|\nabla_{\boldsymbol{w}}\mathcal{L}\| \\
\|\zeta_3\| &= \sqrt{\sum_i \lambda_i^2\alpha_i^2} \leq \sqrt{\sum_i \lambda_i^2}\sqrt{\sum_i \alpha_i^2} = \psi\|\nabla_{\boldsymbol{w}}\mathcal{L}\| \qquad (52)
\end{aligned}$$

where $\psi = \sqrt{\sum_i \lambda_i^2} = \|\nabla^2_{\boldsymbol{w}}\mathcal{L}\|_F$, i.e. ψ is the Frobenius norm of the loss Hessian. Applying the triangle inequality and ineq. 52, we can upper bound the expression in inequality 51:

$$\begin{aligned}
\|b\Delta t\zeta_1 + 2\zeta_2 - 2\Delta t\zeta_3\| &\leq b\Delta t\|\zeta_1\| + 2\|\zeta_2\| + 2\Delta t\|\zeta_3\| \\
&\leq b\Delta t\psi\sqrt{N} + 2\|\nabla_{\boldsymbol{w}}\mathcal{L}\| + 2\Delta t\psi\|\nabla_{\boldsymbol{w}}\mathcal{L}\|
\end{aligned}$$

$$\sum_i(\lambda_i\alpha_i^2\Delta t + 2b\alpha_i\beta_i(1-\lambda_i\Delta t) + \lambda_i b^2\Delta t\beta_i^2) \leq b\sqrt{N}(b\psi\Delta t\sqrt{N} + 2\|\nabla_{\boldsymbol{w}}\mathcal{L}\| + 2\psi\Delta t\|\nabla_{\boldsymbol{w}}\mathcal{L}\|)$$

$$\sum_i(\lambda_i\alpha_i^2\Delta t + 2b\alpha_i\beta_i(1-\lambda_i\Delta t) + \lambda_i b^2\Delta t\beta_i^2) \leq b^2\psi\Delta tN + 2b\sqrt{N}\|\nabla_{\boldsymbol{w}}\mathcal{L}\|(1+\psi\Delta t) \qquad (53)$$

Plugging inequality 53 into Eq. (49), we can upper bound the change in $\mathcal{L}$.

$$\begin{aligned}
&\mathcal{L}[\boldsymbol{w}(t+\Delta t)] - \mathcal{L}[\boldsymbol{w}(t)] \leq -\|\nabla_{\boldsymbol{w}}\mathcal{L}\|^2\Delta t + \frac{\Delta t}{2}(b^2\psi\Delta tN + 2b\sqrt{N}\|\nabla_{\boldsymbol{w}}\mathcal{L}\|(1+\psi\Delta t)) \\
\implies &\mathcal{L}[\boldsymbol{w}(t+\Delta t)] - \mathcal{L}[\boldsymbol{w}(t)] \leq -\|\nabla_{\boldsymbol{w}}\mathcal{L}\|^2\Delta t + \|\nabla_{\boldsymbol{w}}\mathcal{L}\|b\sqrt{N}\Delta t(1+\psi\Delta t) + \frac{b^2N\psi\Delta t^2}{2} \qquad (54)
\end{aligned}$$

To derive the conditions when $\mathcal{L}$ will *necessarily* decrease upon one step of weight update, we can set the above upper limit to be ≤ 0, implying that $\mathcal{L}[w(t+\Delta t] \leq \mathcal{L}[\boldsymbol{w}(t)]$. Therefore,

$$\begin{aligned}
&-\|\nabla_{\boldsymbol{w}}\mathcal{L}\|^2\Delta t + \|\nabla_{\boldsymbol{w}}\mathcal{L}\|b\sqrt{N}\Delta t(1+\psi\Delta t) + \frac{b^2N\psi\Delta t^2}{2} \leq 0 \\
\implies &Q(\|\nabla_{\boldsymbol{w}}\mathcal{L}\|) = \|\nabla_{\boldsymbol{w}}\mathcal{L}\|^2\Delta t - \|\nabla_{\boldsymbol{w}}\mathcal{L}\|b\sqrt{N}\Delta t(1+\psi\Delta t) - \frac{b^2N\psi\Delta t^2}{2} \geq 0 \qquad (55)
\end{aligned}$$

It can be clearly seen that $Q(0) < 0$ and Q is an upward facing parabola. Therefore, one of the roots are negative and the other positive. Since $\|\nabla_{\boldsymbol{w}}\mathcal{L}\|$ cannot be negative, the above condition can only be satisfied iff:

$$\begin{aligned}
\|\nabla_{\boldsymbol{w}}\mathcal{L}\| &\geq \frac{b\sqrt{N}\Delta t(1+\psi\Delta t) + \sqrt{b^2N\Delta t^2(1+\psi\Delta t)^2 + 2b^2N\psi\Delta t^3}}{2\Delta t} \\
\|\nabla_{\boldsymbol{w}}\mathcal{L}\| &\geq b\sqrt{N}\frac{1+\psi\Delta t + \sqrt{1+4\psi\Delta t + \psi^2\Delta t^2}}{2}
\end{aligned}$$

Again, using the relation: $\quad (1+x)^n \approx 1+nx \quad$ when $|x| << 1$

$$\sqrt{1+4\psi\Delta t+\psi^2\Delta t^2} \approx 1 + \frac{1}{2}(4\psi\Delta t + \psi^2\Delta t^2) \approx 1 + 2\psi\Delta t$$

Therefore, $\quad \|\nabla_{\boldsymbol{w}}\mathcal{L}\| \geq b\sqrt{N}(1+\frac{3}{2}\psi\Delta t)$

$$\implies \frac{b}{\|\nabla_{\boldsymbol{w}}\mathcal{L}\|} \leq \frac{1}{\sqrt{N}}\frac{1}{(1+\frac{3}{2}\psi\Delta t)} \qquad (56)$$

□

Tightness of Bound: One of the key inequalities used for proving Theorem A.8 is the dot product inequality. It is well known that the dot product inequality is weaker for high dimensions, specifically because two random vectors in high-dimensional space are more likely to be orthogonal, i.e. their dot product is ≈ 0. This is also reflected in Fig. 8 where as the number of dimensions increases, the bound in Theorem A.8 becomes a weaker. Therefore, even though we have a sufficient condition that produces a decrease in $\mathcal{L}$, it may not accurately reflect the conditions under which $\mathcal{L}$ can decrease almost always. To derive these conditions and derive a more stronger bound, we turn to the probably approximately correct framework and concentration inequalities. The concentration inequality for dot product in high dimensions states that for unit vectors $\hat{X}, \hat{Y}$ in N-dimensional space, $\left\langle \hat{X}, \hat{Y} \right\rangle$ concentrates around $\frac{1}{\sqrt{N}}$. Therefore, we can assume that with very high probability, $\left\langle \hat{X}, \hat{Y} \right\rangle \leq \frac{\sqrt[4]{N}}{\sqrt{N}} = \frac{1}{\sqrt[4]{N}}$. Incorporating this into inequality 51:

$$\begin{aligned}\sum_i \lambda_i \alpha_i^2 \Delta t + 2b\alpha_i\beta_i(1-\lambda_i\Delta t) + \lambda_i b^2 \Delta t \beta_i^2 &= \sum_i \left(b\beta_i(b\beta_i\lambda_i\Delta t + 2\alpha_i(1-\lambda_i\Delta t))\right) \\ &\leq \frac{1}{\sqrt[4]{N}}\sqrt{\sum_i b^2\beta_i^2}\sqrt{\sum_i (b\beta_i\lambda_i\Delta t + 2\alpha_i(1-\lambda_i\Delta t))^2}\end{aligned} \tag{57}$$

Following the rest of the proof as above, we can add a correction factor of $\sqrt[4]{N}$ in the final inequality. Therefore,

$$\|\nabla_{\boldsymbol{w}}\mathcal{L}\| \geq b\sqrt[4]{N}\left(1 + \left(1 + \frac{\sqrt[4]{N}}{2}\right)\psi\Delta t\right) \tag{58}$$

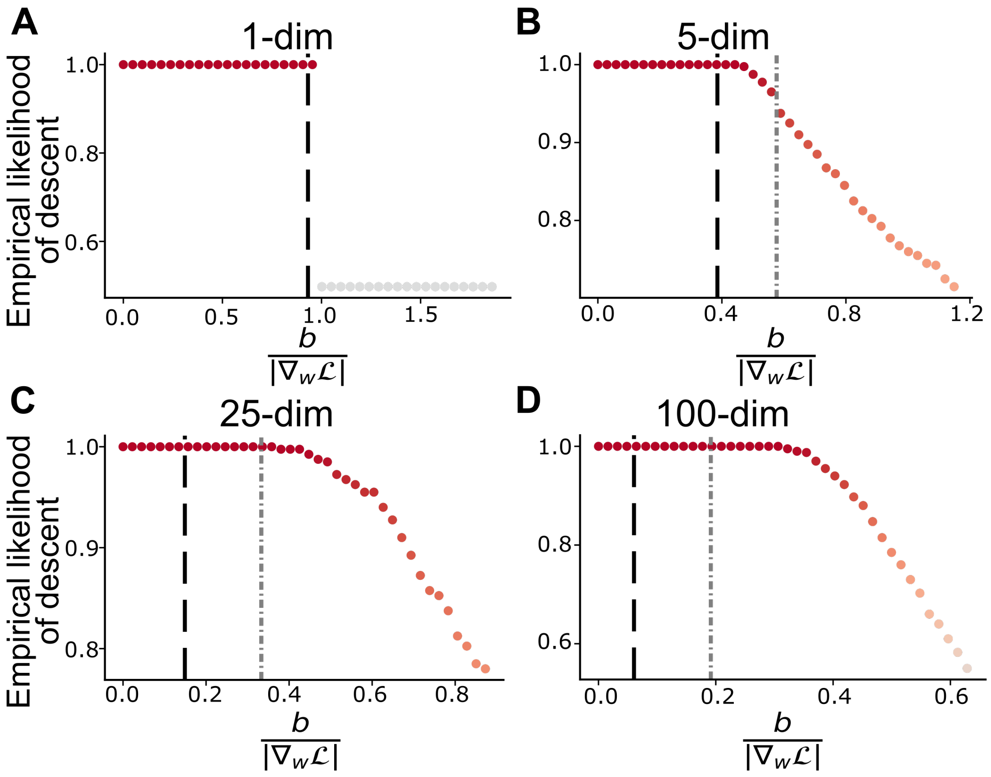


Figure 8: Empiricial verification of Theorem A.8 for increasing dimensionality of parameter space. Black dashed line indicates the bound in Theorem A.8 and Gray dashed dotted line indicates the probably approximately correct bound that offers a stronger condition for higher dimensions. (A) 1 dimensional parameter space; (B) 5-dimensional parameter space where the probably approximate bound satisfies the loss decrease condition with $\approx 95\%$ probability; (C) 25 dimensional parameter space where the black line is a weak bound; (D) 100 dimensional parameter space where the gray line is a stronger bound for decrease in $\mathcal{L}$.

B Experimental details

All experiments were implemented using PyTorch 1.10.2 (license: `https://github.com/pytorch/pytorch/blob/master/LICENSE`). For all experiments, the direction of bias was chosen to be along the direction of vector $\overrightarrow{1}$ for simplicity. However, this choice is not a necessity for our experimental results and merely a design choice. Note that our theoretical results are generic enough for any bias direction. Furthermore, for Figures 3-7, torch DoubleTensor was used, which allowed for float64 level of precision (resolution $\approx$ 2.22e-16). Implementation details pertaining to each figure are presented below

B.1 Figure 1

A VGG-16 (initialized with ImageNet-pretrained weights) was trained to perform object recognition on the CIFAR-10 dataset using the Negative Log Likelihood loss function. Each update entailed computing the gradient over the entire dataset, instead of using the more commonly used stochastic gradient descent. This procedure is more commonly known as the full-batch gradient descent. Some amount of bias and variance was added artificially to the computed gradient before making updates. The amount of variance and bias were chosen to be a function of the gradient norm, such that at each step the ratio of the net variance and/or bias to the gradient norm was constant. We plot the performance degradation/improvement corresponding to this fixed variance and bias ratio in the colorplot. Note that we performed a sweep over different variance and bias ratios in order to generate the 2d colorplot. The standard SGD optimizer from PyTorch was used and the optimizer step was called at the end of each epoch to use the full-batch gradient for updates. Hyperparameter values for training are as follows: epochs = 50, learning rate = 0.02, weight decay = 0.0, momentum = 0.9, batch size = 5000. Each bias and variance configuration was run for 3 seeds and accuracy values were averaged over these 3 seeds for plotting. The training was run on an RTX8000 gpu and accelerated using the ffcv library (Github repo: `https://github.com/libffcv/ffcv`, License: `https://github.com/libffcv/ffcv/blob/main/LICENSE`) and each run (1 seed) took approximately 10 minutes.

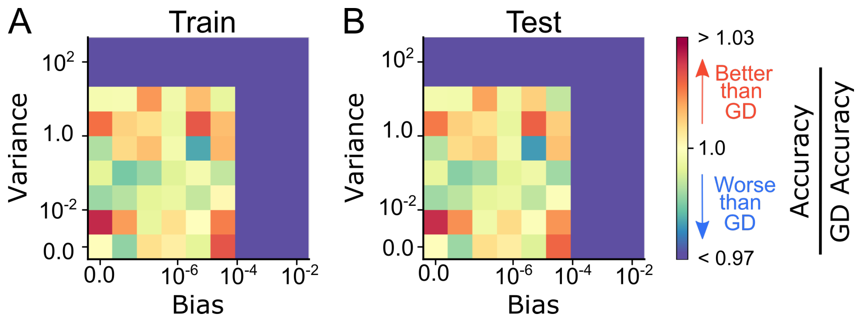


Figure 9: Figure 1 from main text plotted in Spectral colormap. Train and test accuracy of a VGG-16 network trained for 50 epochs (to convergence) on CIFAR-10 using full-batch gradient descent (with no learning rate schedule) with varying amount of variance and bias (as a fraction of the gradient norm) added to the gradient estimates. These results (avg of 20 seeds) indicate that excessive noise and bias harms learning, but a small amount can aid it.

B.2 Figure 3

We ran multiple repeats to sample different noise vectors, typically 20. However, to ensure that the mean of the sampled noise vectors was 0, we artificially sampled an additional noise vector such that the sum of all sampled vectors was 0. This procedure ensured that the bias in gradient estimates was 0 and we could observe the impact of variance, independent of the bias. We observed that this is a feature of finite sampling and if we increase the number of noise vector samples, the sample mean reduces. However, in line with time and compute budget we added this artificial sampling technique.

(A) The teacher network was set to be a single linear layer mapping the input to output. For the student networks with varying width in the hidden layer, the first layer projecting the input to the latent space is kept fixed and initialized such that the variance of the input is preserved. The weights from the hidden units to output are learned using gradient descent. Decreasing the width below the input dimension also proportionally reduces the variance and therefore trace of the hessian reduces. This results in no change in the $\Delta\mathcal{L}$ when width is less than the teacher width.

(B,C) The teacher network was set to have 4 hidden layers with ReLU non-linearity. The student networks were allowed to have varying depth and width in each layer. The width of all layers was set to the same value. For each configuration, a linear and a ReLU student network was run.

All these experiments were ran on CPU and took a couple of hours. For each configuration, 20 different teacher initializations and 20 different student initializations were used, thereby indicating results averaged over different datasets and ANN initialzations.

B.3 FIGURE 4

Similar to Figure 3A, these experiments were ran on a linear teacher and student network to validate the relationship between $\Delta\mathcal{L}$ and bias. For each experiment some bias value was set and the bias vector was artificially added to the gradient vector. Each experiment entailed setting the teacher weights to a random configuration and using different such configurations, typically 20. For each teacher configuration, we ran multiple seeds, typically 20, to simulate multiple student network initializations. For each seed, we trained a control network for 10 epochs in order to efficiently sample parameter configurations exhibiting wide range of gradient norm. All these experiments were ran on CPU and took around an hour.

B.4 FIGURE 5

The teacher network was a VGG-19 configuration trained on the CIFAR-10 dataset to achieve 92.6% accuracy.

(A) For each student network configuration, ImageNet-pretrained and random initialization weights were used. The last layer was re-initialized to have 10 output units as opposed to the typical 1000 units. Each experiment entailed training the student network to match the teacher outputs for 10 different seeds, i.e. student initializations. For each seed, the student was trained for 5 epochs to sample a wide range of gradient norm. For each update step, 20 random vectors were sampled to add noise to gradient. The norm of the vectors was chosen such that the variance in gradient estimates is 20. Similar to Figure 3, the artificial sampling correction was used to ensure an unbiased gradient estimate.

(B) Downsampling indicates reducing the width of network by the downsampling factor. The number of channels in the convolutional layers was reduced by the downsampling factor and the number of units in the linear layers (except the final output layer) were reduced by the downsampling factor. Since these are alterations to the VGG configurations, no ImageNet pretrained weights were available. Hence, only random initializations were used. Same procedure as above was used for each experiment.

Each experiment was run on an RTX8000 GPU and required 10 GB RAM. One experiment took around an hour and multiple experiments were run in parallel on the institute cluster. Note that there was no observed relationship in $\Delta\mathcal{L}$ and gradient norm in any of the experiments.

B.5 FIGURE 6

Same procedure as above but did not require random sampling of noise vectors. Instead, a constant vector along $\vec{1}$ was added to the gradient such that the amount of bias in gradient estimates is $b = 0.02$.

B.6 FIGURE 7

All experiments were run on a linear network setting. For each experiment, the Hessian of the (MSE) loss function was changed by altering the covariance matrix of the input. To compute the empirical likelihood of descent, we sampled different student network initializations (here 100) and a weight update step was performed with respective variance or bias values. For each such update, we counted the number of updates that led to a decrease in the loss and finally divided this number by the total number of updates performed in the experiment, thereby yielding the empirical likelihood of descending the loss landscape. We also ran this process over multiple teacher network initializations (here 10). It is worth noting that for the case of noisy gradients, we followed similar suit as described in experimental procedure for Fig. 3 to ensure that the amount of noise added was zero-mean and the mean decrease in loss was measured.

(A,B) Similar noise vector sampling procedure was employed and the amount of noise to gradient norm ratio was fixed for all experiments. These experiments followed similar suit as Figure 3.

(C) Similar procedure as Figure 4. Although, here the bias to gradient norm ratio was fixed for all experiments.

These experiments were performed on CPU and took less than an hour.

C Detailed Discussion on Related works

C.1 Relation to methods and findings from Raman *et al.*

Our study is significantly motivated and influenced by the setup used by Raman et al. (2019). Despite these strong links, there are certain distinctions in our approach that we elaborate on in this section. In their setup, Raman *et al.* assume the weight update step can be decomposed into a term aligned with the gradient, a term orthogonal to the gradient, and synapse-intrinsic noise. They proceed to derive the optimal network size, specifically width, beyond which learning on the training set is impeded. Instead, in our work, we analyze the role of different architectural choices, like depth, width, and nonlinearity, in mitigating the impact of variance and bias in gradient approximations on both training and generalization performance. Notably, our setting can be used to understand both the impact of intrinsic noise (see extensions to Theorem 2.2) as well as gradient approximation noise on the network performance.

C.2 Relationship to stochastic gradient descent and dropout

Deep learning pipelines rarely use the true gradient of the loss function over the entire dataset to make weight updates. Instead, it is common practice to add noise in different forms, e.g. mini-batch stochasticity (Bottou, 2012), dropout (Srivastava et al., 2014), dropconnect (Wan et al., 2013) and label noise (Blanc et al., 2020; HaoChen et al., 2021; Damian et al., 2021). Although the noise structure may differ from assumptions in our work, our analytical results can be extended to incorporate the specifics in each case and understanding its impact on learning dynamics.

For stochastic gradient descent (SGD), the noise in gradient (under small learning rate) is unbiased. Although SGD does not exhibit white noise structure, several analytical and empirical studies have used white noise structure to model SGD dynamics, and concluded that the SGD noise acts as a regularizer against converging to sharp minima (Foret et al., 2020; Chaudhari et al., 2019; Yao et al., 2018; Ghorbani et al., 2019; Smith et al., 2021). Furthermore, it has been shown that anti-correlated noise injection improves generalization (Orvieto et al., 2022). Moreover, some studies have leveraged Langevin dynamics to understand the learning trajectory under noisy weight updates (Murray & Edwards, 1992; Rögnvaldsson, 1994). Taken together, these results are in agreement with our analysis of variance in gradient estimates and therefore indicate that our work has far reaching consequences in understanding gradient-based learning. Similarly, dropout can be thought of as adding noise to the credit assignment pathway. However, dropout involves turning off a random subset of units in the network and each such configuration yields an approximation to the gradient, characterized by some variance and bias with respect to the true gradient. For a truly unbiased estimate of the gradient, one must compute an estimate of the gradient across all configurations, something quite uncommon. Empirical results in the deep neural networks have demonstrated that dropout leads to better generalization, albeit at the cost of slightly worse training performance. This result indicates that our analysis of bias in gradient estimates is also relevant to understanding how different training strategies affect gradient-based learning dynamics.

C.3 A NOTE ON PRACTICAL APPLICATIONS

We believe that our work is an initial step towards understanding the impact of gradient approximations on learning and generalization and the role of network architecture choice in mitigating such impacts. Notably, we believe that our work could provide guidelines for neuromorphic chip design by suggesting possible network architecture choices as well as choice of nonlinearity when the properties of approximation, i.e. variance and bias in gradient estimates, are characterized. Furthermore, we believe that our work scopes out the knowledge landscape in computational neuroscience for biologically-plausible learning rules and future work could aim to characterize popular learning rules using this framework. Similar characterizations could be carried out in biological agents while they learn to perform a task and therefore understand what regime of variance and bias exists in biological circuits. In doing so, we believe we can better formulate learning rules as well as make testable predictions about learning in the brain.

Part V

Discussion and Conclusions

9 Discussion

In the previous chapters, we have explored specific examples of how neuroscience and AI can mutually inform each other. In Part II, we compared representations in brains and ANNs, and found that both biological and artificial systems have scale-free representations. Inspired by this observation, we used a neuroscience-inspired metric, α-ReQ, to quantify the representation geometry of ANNs, and demonstrated that this metric is indicative of downstream performance for ANNs trained using self-supervised learning (SSL). Next, in Part III, within such SSL models, we investigated the learning dynamics of feature learning. Using tools from deep learning theory, we provided a normative understanding of how certain biological mechanisms of learning affect the feature learning dynamics; the mechanisms we examined include orthogonality constraints on neural activities (thought to be implemented by inhibitory neurons [Giridhar et al., 2011, Duong et al., 2023]) and multiple views for exploring novel objects [Antunes and Biala, 2012]. We leveraged this understanding to improve the compute and sample-efficiency of SSL pipelines. Finally, in Part IV we compared learning mechanisms in the brain and ANNs, and analytically and empirically characterized how approximating the gradient could affect learning and generalization in connectionist systems. Moreover, we studied the impact of architectural design choices like size of the network and the activation sparsification non-linearity on mitigating the negative impacts of gradient approximation while still preserving the benefits of noisy optimization, thereby providing a potential normative explanation of certain architectural motifs in the brain.

Taken together, these examples demonstrate that experimental observations in neuroscience can guide the design of AI systems, while ANNs can serve as a controlled experimental platform for gaining a normative understanding of mechanisms underlying biological intelligence. In this chapter, I will present a broader perspective on the NeuroAI framework (Figure 1.1), discussing some important caveats and challenges that act as roadblocks in applying this framework. Finally, I will explore the broader implications of the findings presented in this thesis and outline potential directions for future research.

9.1 Is the framework sufficient?

Returning to the NeuroAI framework discussed in the introduction of this thesis, the work I have presented thus far touched upon the learned representations and learning rule axes of this framework. The same philosophy can be extended to understand synergies between brains and ANNs for the other framework pillars. Therefore, at this point, a reader might wonder: *is this synergy universal?* As such, can we apply this philosophy readily and uncover multiple aspects of intelligence? Can we leverage these insights to uncover shared motifs of intelligence and improve AI systems in the process?

A key consideration of widely adopting the NeuroAI framework lies in the open-endedness and subjectivity of this framework. A researcher adopting this framework has the flexibility to decide the abstraction level at which they want to operate at, select the scope of the synergy they are interested in exploring, and subsequently investigate the potential benefits that a particular observation might offer to support computations underlying intelligence. Let us try to understand the challenges that come with every step of this process.

First, let us consider the choice of the abstraction level. Brains are complex systems, performing computations at various scales, ranging from whole-brain to individual brain regions and neuronal populations to individual neurons to subcellular compartments within neurons. While using connectionist models like ANNs to model neural computations, the researcher is faced with choosing the appropriate level of abstraction of neural computations they in-tend to model or draw parallels with. For instance, the term "neural representations" could correspond to very different physical substrates at each level of abstraction. For studying whole brain activity, using methods like functional brain imaging, representations correspond to average population activity of a large number of neurons [Schrimpf et al., 2018]. For neuronal ensemble or single cell activity, studied with electrophysiology or optical imaging, representations could correspond to average activity of a few proximal neurons or single neurons, depending on the recording technique [Azabou et al., 2023]. For cellular or subcellular activity, representations could correspond to membrane potentials at different dendritic compartments within a single neuron [Guerguiev et al., 2017]. It is hard to imagine a property that would be preserved across representations at all these scales, and therefore could be leveraged to improve AI systems or even build ANN models to study the utility of such properties.

Understanding computations at each abstraction level requires deciding on modeling choice followed by significant engineering efforts to build reasonable models that can replicate the computations of interest. Consequently, the uncovered shared principles of intelligence across biological and artificial intelligence systems will be a function of the abstraction level being modeled. As an example, consider the three chapters in this thesis. Part II leverages property of representations at the level of single neurons in the mouse visual cortex, while Part III also leverages observations about organism-level behavior. Despite its more theoretical flavor, Part IV discusses a problem with credit assignment in the brain, and can potentially pave the way for future studies aimed at understanding the role of dendritic compartments within a single neuron in solving the credit assignment problem. While each chapter lies at the intersection of neuroscience and AI in its own right, their implications are relevant for slightly different audiences. As a result, communicating shared principles of intelligence require sufficient context that clearly indicate the level of abstraction those principles are valid for, as well as explanations of limitations. Unifying these principles across computational levels will require ongoing efforts and represents a long-term goal of the NeuroAI framework.

A greater challenge is in selecting the scope of the synergy to be explored. Nervous systems in organisms are products of genetic, epigenetic and cultural evolution [Jablonka and Lamb, 2006] — optimization processes that take place across generations. Therefore, it is often unclear whether certain properties of neural systems exist to support computations underlying learning during the organism's lifetime (development), are products of evolutionary adaptation to some archaic constraints (evolution), or are simply idiosyncrasies of the system (unrelated to function). Consider the example of linguistic communication abilities, often used as a proxy for reasoning abilities. It is unclear whether there are genetic predispositions or transfer of epigenetic information that enable humans to specifically learn a language, or whether it is learned by leveraging mechanisms useful for lifetime learning. Moreover, it is unclear how mechanisms underlying language understanding descended from previous non-linguistic communication systems that existed in other species of the *homo genus* [Donald, 2019]. Of course, there is also cultural transmission of linguistic information, especially during early childhood where language learning ability in humans peaks [Gómez and Gerken, 2000]. As a result, studies aiming to draw motivation from how humans process language to build better language understanding in AI systems face a significant challenge of identifying neural mechanisms that are actually useful for language learning, per se.

A final challenge that is shared across many interdisciplinary fields is to convince researchers working in the base fields of the utility of an intermingling exercise. Traditionally, neuroscientists sought mechanistic models of understanding computations in the brain – building up explanations of behavior by combining components at molecular, cellular, and systems level of understanding. This is in contradiction to the promise of the NeuroAI framework, which can offer normative explanations and needs to work at a certain level of abstraction, often ignoring specific biological details at other levels [Saxe et al., 2021]. On the other hand, AI researchers sought scalable solutions that can be used to solve existing problems, which are often of a targeted and specific nature. Unfortunately, biological inspiration might not present the path for designing the most scalable solutions, whereas large-scale engineering efforts have resulted in solutions [Gershman, 2024]. As a result, the burden of proof of the utility to combine the two fields rests on those seeking to explore this avenue. While doing so to understand the shared principles of intelligence is a noble goal in itself, science is done by people and people are embedded in society. Hence, it is imperative to carefully think of the potential offerings that this framework offers to existing scientific disciplines and industry endeavors [Luppi et al., 2024].

While the challenges outlined here might paint a gloomy picture for widely adopting the NeuroAI framework, it is worth emphasizing that the emergence of connectionist models as the forefront models of intelligence is fairly recent [Doerig et al., 2023]. As a result, the NeuroAI research program is still in its nascent stages. Like every other nascent field, it will mature with time and carve out its own niche. With this optimistic perspective, I will discuss potential broader implications of the findings presented in this thesis, and how they might shape the trajectory of the field in the near future.

9.2 Representations in brains and ANNs

Our findings in Part II highlight that learned representations in ANNs, especially those trained using SSL, are scale-free, i.e., they have heavy-tailed eigenspectra that follow a powerlaw. In Part III, we studied the learning dynamics of SSL and showed that features learned in an SSL pipeline are those that best match the data similarity kernel, that is in turn explicitly defined by the data augmentations. Intuitively, the network learns a feature space that matches the structure of the world, i.e. distance between two data points in the feature space roughly matches how semantically different those data points are [Zhai et al., 2024]. Consequently, a powerlaw in the eigenspectrum of learned features reflects the heavy-tailed nature of the stimulus space. Moreover, our results show that the tail of the eigenspectrum gets "heavier" (i.e. representations become more high rank) as learning proceeds. A natural question that arises at this point is: *what is being learned in the tail?*

Let us start by taking a closer look at our findings regarding the behavior of the tail and the body of the eigenspectrum (here we use "body" to refer to the top few eigendirections). Our findings from Part II indicate that this information is indeed task-specific: the slope of the eigenspectrum decay is indicative of the downstream task performance. On the other hand, our findings from Part III indicate that the features corresponding to larger singular values of the data similarity kernel are learned earlier during training, while smaller singular values are learned later. Given that the body of the eigenspectrum, i.e. eigenvectors corresponding to the strongest singular values of the data similarity kernel, is learned very early during training, the information being learned by the body would be the coarse semantic structure of the stimulus space indicating coarse-level distinctions in the world. Examples of such distinctions may include distinguishing between animate and inanimate objects, or differentiating between outdoor and indoor background scenes. In contrast, fine-grained semantic information, potentially indicative of object identities or configurations (such as information necessary for interacting with objects), would correspond to smaller singular values in the data similarity kernel, and therefore, are learned later during training. If this fine-grained semantic information about the stimulus space is encoded in the tail of the eigenspectrum of the feature covariance matrix, then networks that fail to learn the correct information geometry in the tail would correspond to worse performance on downstream tasks. Indeed, our results in Part III demonstrate that representations learned later in training that have a “heavier” tail correspond to good performance on the object recognition task. Therefore, for high-dimensional, scale-free representations, the tail of the eigenspectrum, i.e. directions in the representation space that correspond to lower variance, encode information that is useful for downstream tasks.

While this finding may seem paradoxical to the reader at first glance, there are certain benefits to using such a scale-free representation encoding strategy. Firstly, a scale-free representation, characterized by a powerlaw decay in the eigenspectrum, allows the network to learn different features at varying scale. As a result, two inputs with slight differences will end up being represented similarly, given that the difference between the inputs is in a feature that is encoded in the tail. For instance, let us consider two images of dogs, with slight differences in their fur colors in the image. The network might represent both the images similarly, attributing the slight differences in fur color to differences in lighting conditions. This property allows the network to learn representations that are robust to certain noise statistics in the inputs. Conversely, if such a difference is significant, the two inputs will be sufficiently distinctly represented and potentially lead to different downstream behavior. In our previous example, assume that the dog’s fur color in one of the images is beige, and in the other, it is white. In this case, the network might represent the two images differently, either indicating two different dogs or difference in the state of the same dog (clean vs dirty), resulting in downstream effects of this difference in the representation space. Therefore, a scale-free representation enables the agent to navigate the fundamental tradeoff between efficiency and robustness, wherein the agent needs to capture relevant information about the stimulus space without being overly sensitive to observation noise (see [Stringer et al., 2019] for a detailed discussion on the relationship between the slope of the power law and the efficiency-robustness tradeoff).

At this point, an astute reader (potentially *Ipcha Mistabra*?) might question the universality of such scale-free representations. As such, *do all representations need to be scale-free*? The answer to that is *not necessarily*! Note that, in the previous paragraph, we were considering a system that learns features that reflect the semantic structure of the stimulus space. Such systems are committed to learning representations that are generic enough to support a large variety of downstream behavior. However, this requirement might not necessarily hold for functionally specialized systems that care about supporting only a few related behaviors [DiCarlo et al., 2012, Sorscher et al., 2022]. For instance, the face recognition area (fusiform face area) does not need to learn features to discriminate the color of dog fur. Although such systems still need to navigate the efficiency-robustness tradeoff to some extent, they might not need scale-free representations to do so and may instead rely on other strategies.

Another question that might arise from the Ipcha Mistabra is *whether this distinction between the body and tail is even necessary.* Specifically, one might argue that the distinction between the body and tail of eigenspectrum is artificial, and we do not need tools to characterize them separately. There are two key points to argue in favor of this distinction. First, the result that the tail of the eigenspectrum contains information task-relevant information, more so than the body, is key to explaining why low-dimensional projections of high-dimensional representations are not useful to understand a system's behavior. It is important to note that low-dimensional projections, as done currently in most neuroscience and AI studies, preserve the information in the body of the eigenspectrum (directions with highest variance) while losing the information in the tail of the eigenspectrum (directions with low variance). While previous systems neuroscience studies have shown this apparently-paradoxical nature of high-dimensional representations in the brain [Gao et al., 2017], recent evidence from large language model studies suggest the same for large-scale ANN systems [Friedman et al., 2023].

Second, the distinction between body and tail has significant consequences on the topic of comparing representations across different intelligent systems. Specifically, common methods comparing representations across different intelligent systems compare differences in the kernel space of the network (i.e. how are different stimuli represented in each representation space) [Kornblith et al., 2019], and because these methods rely on the eigenspectrum body of the representation space, they are more sensitive to coarse differences [Cloos et al., 2024]. Consequently, such metrics of representation space similarity will not be indicative of differences in the downstream task behavior, compared to metrics that consider the entire eigenspectrum (instead of just the body of the eigenspectrum). This point is empirically

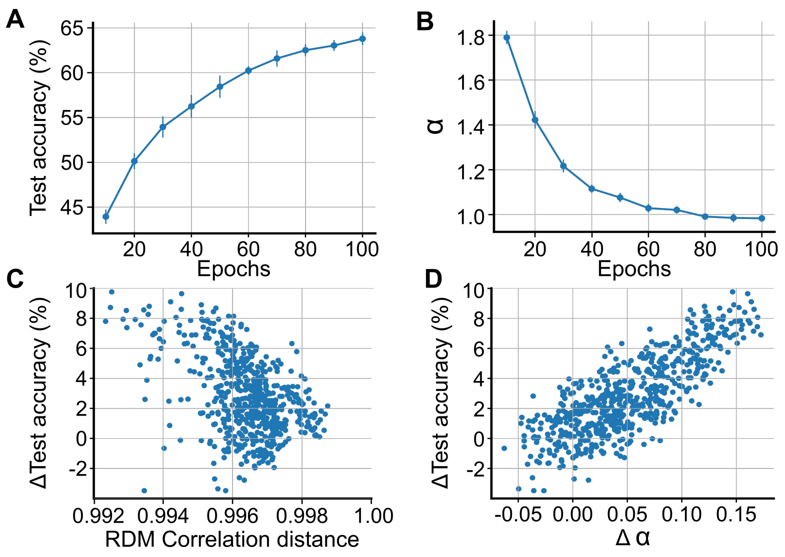


Figure 9.1: Metrics of representation similarity that capture the tail of the eigenspectrum better reflect model behavior, compared to those that capture only the body of the eigenspectrum. (A) Image classification accuracy of a ResNet-18 model pretrained using the BarlowTwins loss function on the Imagenet-100 dataset, averaged over 5 seeds. (B) Spectral decay coefficient, α, of the learned representations across training, averaged over 5 seeds. (C) Pairwise difference in representations, measured using correlation between respective Representation Dissimilarity Matrices (RDM), and performance across models trained using the BarlowTwins loss function (Pearson Correlation coefficient, $\rho = -0.474$). Each point indicates the difference in representation and downstream task performance between two models chosen from the set of all ResNet-18 models trained for at least 40 epochs. (D) Same as C, but using difference in α as a proxy to measure the difference in the eigenspectrum tail of the representations (Pearson Correlation coefficient, $\rho = 0.797$). Error bars indicate standard deviation.

demonstrated in Figure 9.1, wherein we compare the representation space of a ResNet-18 network trained using a SSL loss function, specifically BarlowTwins [Zbontar et al., 2021], on the Imagenet dataset for 100 epochs with another network trained for fewer epochs. While the representation similarity metric (here, the RSA score), is very similar for networks trained for 50 epochs or more, the downstream object recognition performance is indeed different. Extending this to recent studies claiming a convergence of representation spaces [Huh et al., 2024], it's worth noting that since these studies use common representation similarity metrics, they may be simply capturing similarities in the eigenspectrum body and not necessarily the tail. Further investigation is needed to understand whether this convergence is limited only to the coarse structure of the learned semantic stimulus space or applies to the entire representation space. Taken together, the distinction between the body and tail of eigenspectrum is an important consideration in the field of NeuroAI when comparing scale-free representation spaces, e.g. ANNs and the brain.

9.3 Credit assignment in brains and ANNs

While ANNs traditionally rely on gradient descent to optimize their parameters, Part IV demonstrated that effective learning and generalization is possible even with approximate gradients. This finding is particularly intriguing when considering the biological brain, which lacks the computational machinery to compute exact gradients [Lillicrap et al., 2020]. It is important to keep in mind that the brain's credit assignment problem, i.e. the challenge of determining which neurons or synapses contributed to a particular outcome, has been solved through a co-evolutionary process. Specifically, the architecture and the credit assignment algorithms it employs have most likely evolved together, with advancements in one enabling improvements in the other. Therefore, understanding the neural circuitry, including the neuronal morphology and connectivity structure among neurons, is thought to be crucial for understanding the mechanisms of biological credit assignment.

Current proposals for credit assignment algorithms often overlook the specific architectural properties of the brain. However, recent advances in connectomics, such as the detailed mapping of the fly brain [Schlegel et al., 2024], offer valuable insights into the architecture of the neural circuitry of a system. These insights about the architecture design space can in turn help us characterize the nature of gradient approximations that would enable learning and generalization in an equivalent computational model. Researchers in the field of NeuroAI are aptly situated to leverage these advances in systems neuroscience to develop credit assignment algorithms that are tailored to the specific architectural properties of different neural systems. It is, however, important to recognize that the co-evolution of algorithms and hardware may lead to diverse credit assignment strategies across different species. Organisms with few levels of hierarchical processing, for example, might employ distinct algorithms from those that have multiple levels of hierarchical information processing. For instance, simpler organisms with fewer hierarchical layers may rely on more local learning rules with access to some global credit signals in order to optimize a behavior-related objective function [Bellec et al., 2020]. On the contrary, complex organisms with deep hierarchies may leverage local learning rules to optimize different objective functions at different hierarchical stages, leading to distinct functional specialization properties [Illing et al., 2021, Payeur et al., 2021].

From an AI perspective, the computational and memory cost of training large-scale ANNs using end-to-end gradient descent can be prohibitive [Raffel et al., 2020, Ding et al., 2022]. Layer-wise local learning rules, which lead to some approximation of the true gradient, offer a potential alternative to end-to-end backpropagation with the promise of alleviating the computational burden. Recent advances in parameter and memory-efficient training, such as LoRA [Hu et al., 2022] and GaLore [Zhao et al., 2024], have shown promising results in fine-tuning large models without computing the exact gradient for the entire model. By drawing inspiration from the brain, it might be possible to discover novel credit assignment algorithms that are both efficient and effective. Rather than focusing solely on better gradient approximation algorithms, it may be promising to explore architecture-aware algorithms that leverage the specific properties of a given neural network and connect them to the structure of the loss landscape and its true gradients. Such advances in understanding the loss landscape could lead to better gradient approximation algorithms for the specific network architecture. This shift in perspective could lead to significant advances in training large-scale AI systems.

Taken together, the interplay between credit assignment algorithms and neural architecture is a fundamental aspect of both brains and AI systems. By understanding the principles underlying biological credit assignment and applying such architecture-aware credit assignment algorithms to AI, we can potentially develop more efficient and cheaper learning systems.

10

Conclusion

In this dissertation, I aimed to present evidence in favor of the continuing synergy between neuroscience and AI. Insights from neuroscience about learning in the brain can help improve the design of AI systems. At the same time, ANNs can be used as in-silico models of the brain, thereby acting as a testbed to develop normative explanations for the computational advantages of experimental observations in neuroscience. I also outlined a NeuroAI framework for discovering these synergies at various levels of analysis, along with specific examples of how such synergies can be explored under the pillars of learned representations and learning rule. Finally, I presented limitations and challenges regarding the universal adoption of this framework, yet presented a positive outlook regarding how the present thesis may impact the future research directions in the field of NeuroAI.

While I do not hope to convince a reader to become a strong proponent of NeuroAI at this juncture, I hope that the reader shares my optimism about this synergistic relationship between neuroscience and AI to some degree. A general skepticism about the extent to which this synergistic relationship can shape the fields of neuroscience and AI is understandable, and indeed necessary. I believe such skepticism is an essential component of the scientific process, and such discourse will only help the field of NeuroAI progress towards its goal of strengthening the exchange of ideas across its two base disciplines and accelerating advances in each of them. As AI becomes an integral technology in science and society, I believe that it is likely that the synergistic relationship between the fields of neuroscience and AI will also continue to evolve, leading to groundbreaking discoveries over the next century of science.

Publications

1. **A. Ghosh***, K.K. Agrawal*, S. Sodhani, A. Oberman†, B.A. Richards†, *Harnessing small projectors and multiple views for efficient vision pre-training*, 38th Conference on Neural Information Processing Systems (NeurIPS), Vancouver, Dec 2024.

2. R. Chua, **A. Ghosh**, C. Kaplanis, B.A. Richards, D. Precup, *Learning Successor Features the Simple Way*, 38th Conference on Neural Information Processing Systems (NeurIPS), Vancouver, Dec 2024.

3. R. Pogodin*, J. Cornford*, **A. Ghosh**, G. Gidel, G. Lajoie, B.A. Richards, *Synaptic Weight Distributions Depend on the Geometry of Plasticity*, 12th International Conference on Learning Representations (ICLR), Vienna, May 2024 [**spotlight**].

4. P. Li*, J. Cornford*, **A. Ghosh**, B.A. Richards, *Learning better with Dale's law: a spectral perspective*, 37th Conference on Neural Information Processing Systems (NeurIPS), New Orleans, Dec 2023.

5. Z. Chorghay, V.J. Li, A. Schohl, **A. Ghosh**, E.S. Ruthazer. *The effects of the NMDAR co-agonist D-serine on the structure and function of the optic tectum.* Scientific Reports, Aug 17 2023;13(1):13383.

6. **A. Ghosh**, Y.H. Liu, G. Lajoie, K.P. Körding, B.A. Richards, *How gradient estimator variance and bias impact learning in neural networks*, 11th International Conference on Learning Representations (ICLR), Kigali, Rwanda, May 2023.

7. K.K. Agrawal*, A.K. Mondal*, **A. Ghosh***, B.A. Richards, *α-ReQ: Assessing representation quality by measuring eigenspectrum decay*, 36th Conference on Neural Information Processing Systems (NeurIPS), New Orleans, Dec 2022.

8. Y.H. Liu, **A. Ghosh**, B.A. Richards, E. Shea-Brown, G. Lajoie, *Beyond accuracy: generalization properties of bio-plausible temporal credit assignment rules*, 36th Conference on Neural Information Processing Systems (NeurIPS), New Orleans, Dec 2022.

9. R.H. Eyono, E. Boven, **A. Ghosh**, J. Pemberton, F. Scherr, C. Clopath, R.P. Costa, W. Maass, B.A. Richards, C. Savin, K.A. Wilmes, L.Y. Prince, *Current State and Future Directions for Learning in Biological Recurrent Neural Networks: A Perspective Piece.* Neurons, Behavior, Data analysis, and Theory, p.35302, April 2022.

---

*,† Equal contribution

Bibliography

[Agrawal et al., 2022] Agrawal, K. K., Mondal, A. K., Ghosh, A., and Richards, B. (2022). α-req: Assessing representation quality in self-supervised learning by measuring eigenspectrum decay. *Advances in Neural Information Processing Systems*, 35:17626–17638.

[Antunes and Biala, 2012] Antunes, M. and Biala, G. (2012). The novel object recognition memory: neurobiology, test procedure, and its modifications. *Cognitive processing*, 13:93–110.

[Assran et al., 2023] Assran, M., Duval, Q., Misra, I., Bojanowski, P., Vincent, P., Rabbat, M., LeCun, Y., and Ballas, N. (2023). Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15619–15629.

[Azabou et al., 2023] Azabou, M., Arora, V., Ganesh, V., Mao, X., Nachimuthu, S., Mendelson, M., Richards, B., Perich, M., Lajoie, G., and Dyer, E. (2023). A unified, scalable framework for neural population decoding. *Advances in Neural Information Processing Systems*, 37.

[Ba et al., 2016] Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

[Bahdanau et al., 2014] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[Balestriero et al., 2023] Balestriero, R., Ibrahim, M., Sobal, V., Morcos, A., Shekhar, S., Goldstein, T., Bordes, F., Bardes, A., Mialon, G., Tian, Y., et al. (2023). A cookbook of self-supervised learning. *arXiv preprint arXiv:2304.12210*.

[Bardes et al., 2022] Bardes, A., Ponce, J., and LeCun, Y. (2022). Vicreg: Variance-invariance-covariance regularization for self-supervised learning. In *International Conference on Learning Representations*.

[Barlow et al., 1961] Barlow, H. B. et al. (1961). Possible principles underlying the transformation of sensory messages. *Sensory communication*, 1(01):217–233.

[Bellec et al., 2020] Bellec, G., Scherr, F., Subramoney, A., Hajek, E., Salaj, D., Legenstein, R., and Maass, W. (2020). A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature communications*, 11(1):3625.

[Boyd and Vandenberghe, 2004] Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

[Bredenberg and Savin, 2024] Bredenberg, C. and Savin, C. (2024). Desiderata for normative models of synaptic plasticity. *Neural Computation*, 36(7):1245–1285.

[Bubeck et al., 2015] Bubeck, S. et al. (2015). Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357.

[Chen et al., 2020] Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

[Cloos et al., 2024] Cloos, N., Li, M., Siegel, M., Brincat, S. L., Miller, E. K., Yang, G. R., and Cueva, C. J. (2024). Differentiable optimization of similarity scores between models and brains. *arXiv preprint arXiv:2407.07059*.

[DiCarlo et al., 2012] DiCarlo, J. J., Zoccolan, D., and Rust, N. C. (2012). How does the brain solve visual object recognition? *Neuron*, 73(3):415–434.

[Ding et al., 2022] Ding, N., Qin, Y., Yang, G., Wei, F., Yang, Z., Su, Y., Hu, S., Chen, Y., Chan, C.-M., Chen, W., et al. (2022). Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models. *arXiv preprint arXiv:2203.06904*.

[Doerig et al., 2023] Doerig, A., Sommers, R. P., Seeliger, K., Richards, B., Ismael, J., Lindsay, G. W., Kording, K. P., Konkle, T., Van Gerven, M. A., Kriegeskorte, N., et al. (2023). The neuroconnectionist research programme. *Nature Reviews Neuroscience*, 24(7):431–450.

[Donald, 2019] Donald, M. (2019). Key cognitive preconditions for the evolution of language. *Handbook of Cognitive Archaeology*, pages 287–295.

[Dosovitskiy et al., 2021] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

[Duong et al., 2023] Duong, L., Lipshutz, D., Heeger, D., Chklovskii, D., and Simoncelli, E. P. (2023). Adaptive whitening in neural populations with gain-modulating interneurons. In *International Conference on Machine Learning*, pages 8902–8921. PMLR.

[Failor et al., 2021] Failor, S. W., Carandini, M., and Harris, K. D. (2021). Visuomotor association orthogonalizes visual cortical population codes. *BioRxiv*, pages 2021–05.

[Friedman et al., 2023] Friedman, D., Lampinen, A., Dixon, L., Chen, D., and Ghandeharioun, A. (2023). Interpretability illusions in the generalization of simplified models. *arXiv preprint arXiv:2312.03656*.

[Gao et al., 2017] Gao, P., Trautmann, E., Yu, B., Santhanam, G., Ryu, S., Shenoy, K., and Ganguli, S. (2017). A theory of multineuronal dimensionality, dynamics and measurement. *BioRxiv*, page 214262.

[Gershman, 2024] Gershman, S. J. (2024). What have we learned about artificial intelligence from studying the brain? *Biological Cybernetics*, pages 1–5.

[Ghosh et al., 2024] Ghosh, A., Agrawal, K. K., Sodhani, S., Oberman, A., and Richards, B. A. (2024). Harnessing small projectors and multiple views for efficient vision pretraining. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

[Ghosh et al., 2023] Ghosh, A., Liu, Y. H., Lajoie, G., Kording, K., and Richards, B. A. (2023). How gradient estimator variance and bias impact learning in neural networks. In *The Eleventh International Conference on Learning Representations*.

[Giridhar et al., 2011] Giridhar, S., Doiron, B., and Urban, N. N. (2011). Timescale-dependent shaping of correlation by olfactory bulb lateral inhibition. *Proceedings of the National Academy of Sciences*, 108(14):5843–5848.

[Gómez and Gerken, 2000] Gómez, R. L. and Gerken, L. (2000). Infant artificial language learning and language acquisition. *Trends in cognitive sciences*, 4(5):178–186.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*, volume 1 (2). MIT Press.

[Grill et al., 2020] Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al. (2020). Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284.

[Guerguiev et al., 2017] Guerguiev, J., Lillicrap, T. P., and Richards, B. A. (2017). Towards deep learning with segregated dendrites. *Elife*, 6:e22901.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[Hornik et al., 1989] Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.

[Hu et al., 2022] Hu, E. J., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al. (2022). Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

[Huh et al., 2024] Huh, M., Cheung, B., Wang, T., and Isola, P. (2024). The platonic representation hypothesis. *arXiv preprint arXiv:2405.07987*.

[Illing et al., 2021] Illing, B., Ventura, J., Bellec, G., and Gerstner, W. (2021). Local plasticity rules can learn deep representations using self-supervised contrastive predictions. *Advances in neural information processing systems*, 34:30365–30379.

[Ioffe and Szegedy, 2015] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 448–456.

[Jablonka and Lamb, 2006] Jablonka, E. and Lamb, M. J. (2006). *Evolution in four dimensions, revised edition: Genetic, epigenetic, behavioral, and symbolic variation in the history of life*. MIT press.

[Jumper et al., 2021] Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. (2021). Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Kong et al., 2022] Kong, N. C., Margalit, E., Gardner, J. L., and Norcia, A. M. (2022). Increasing neural network robustness improves match to macaque v1 eigenspectrum, spatial frequency preference and predictivity. *PLOS Computational Biology*, 18(1):e1009739.

[Kornblith et al., 2019] Kornblith, S., Norouzi, M., Lee, H., and Hinton, G. (2019). Similarity of neural network representations revisited. In *International conference on machine learning*, pages 3519–3529. PMLR.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

[LeCun, 2022] LeCun, Y. (2022). A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62(1):1–62.

[LeCun and Bengio, 1995] LeCun, Y. and Bengio, Y. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.

[LeCun et al., 2015] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

[Lillicrap et al., 2020] Lillicrap, T. P., Santoro, A., Marris, L., Akerman, C. J., and Hinton, G. (2020). Backpropagation and the brain. *Nature Reviews Neuroscience*, 21(6):335–346.

[Luppi et al., 2024] Luppi, A. I., Achterberg, J., Schmidgall, S., Bilgin, I. P., Herholz, P., Sprang, M., Fockter, B., Ham, A. S., Thorat, S., Ziaei, R., et al. (2024). Trainees' perspectives and recommendations for catalyzing the next generation of neuroai researchers. *nature communications*, 15(1):9152.

[Marr, 1982] Marr, D. (1982). *Vision: A computational investigation into the human representation and processing of visual information.* MIT press.

[Nemirovskij and Yudin, 1983] Nemirovskij, A. S. and Yudin, D. B. (1983). *Problem complexity and method efficiency in optimization.* Wiley-Interscience.

[Oquab et al., 2024] Oquab, M., Darcet, T., Moutakanni, T., Vo, H. V., Szafraniec, M., Khalidov, V., Fernandez, P., HAZIZA, D., Massa, F., El-Nouby, A., et al. (2024). Dinov2: Learning robust visual features without supervision. *Transactions on Machine Learning Research.*

[Payeur et al., 2021] Payeur, A., Guerguiev, J., Zenke, F., Richards, B. A., and Naud, R. (2021). Burst-dependent synaptic plasticity can coordinate learning in hierarchical circuits. *Nature neuroscience*, 24(7):1010–1019.

[Raffel et al., 2020] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

[Richards and Kording, 2023] Richards, B. A. and Kording, K. P. (2023). The study of plasticity has always been about gradients. *The Journal of Physiology*, 601(15):3141–3149.

[Richards et al., 2019] Richards, B. A., Lillicrap, T. P., Beaudoin, P., Bengio, Y., Bogacz, R., Christensen, A., Clopath, C., Costa, R. P., de Berker, A., Ganguli, S., et al. (2019). A deep learning framework for neuroscience. *Nature neuroscience*, 22(11):1761–1770.

[Rosenblatt, 1958] Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization. *Psychological Review*, 65(6):386–408.

[Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

[Saxe et al., 2021] Saxe, A., Nelli, S., and Summerfield, C. (2021). If deep learning is the answer, what is the question? *Nature Reviews Neuroscience*, 22(1):55–67.

[Schlegel et al., 2024] Schlegel, P., Yin, Y., Bates, A. S., Dorkenwald, S., Eichler, K., Brooks, P., Han, D. S., Gkantia, M., Dos Santos, M., Munnelly, E. J., et al. (2024). Whole-brain annotation and multi-connectome cell typing of drosophila. *Nature*, 634(8032):139–152.

[Schrimpf et al., 2018] Schrimpf, M., Kubilius, J., Hong, H., Majaj, N. J., Rajalingham, R., Issa, E. B., Kar, K., Bashivan, P., Prescott-Roy, J., Geiger, F., et al. (2018). Brain-score: Which artificial neural network for object recognition is most brain-like? *BioRxiv*, page 407007.

[Sorscher et al., 2022] Sorscher, B., Ganguli, S., and Sompolinsky, H. (2022). Neural representational geometry underlies few-shot concept learning. *Proceedings of the National Academy of Sciences*, 119(43):e2200800119.

[Strang, 2022] Strang, G. (2022). *Introduction to linear algebra.* SIAM.

[Stringer et al., 2019] Stringer, C., Pachitariu, M., Steinmetz, N., Carandini, M., and Harris, K. D. (2019). High-dimensional geometry of population responses in visual cortex. *Nature*, 571(7765):361–365.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems December*, pages 6000–6010.

[Yuan et al., 2021] Yuan, L., Chen, Y., Wang, T., Yu, W., Shi, Y., Jiang, Z.-H., Tay, F. E., Feng, J., and Yan, S. (2021). Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 558–567.

[Zbontar et al., 2021] Zbontar, J., Jing, L., Misra, I., LeCun, Y., and Deny, S. (2021). Barlow twins: Self-supervised learning via redundancy reduction. In *International conference on machine learning*, pages 12310–12320. PMLR.

[Zhai et al., 2024] Zhai, R., Liu, B., Risteski, A., Kolter, J. Z., and Ravikumar, P. K. (2024). Understanding augmentation-based self-supervised representation learning via rkhs approximation and regression. In *The Twelfth International Conference on Learning Representations*.

[Zhao et al., 2024] Zhao, J., Zhang, Z., Chen, B., Wang, Z., Anandkumar, A., and Tian, Y. (2024). Galore: Memory-efficient llm training by gradient low-rank projection. In *Forty-first International Conference on Machine Learning*.