Unsupervised Domain Adaptation for Deconvolution of Spatial Transcriptomics Spots

William Ma
Department of Electrical and Computer Engineering
McGill University
Montréal, Québec, Canada
July 24, 2023

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

Master of Science.

Abstract

Interest in local intercellular communication within tissues in fields such developmental biology, neuroscience, and oncology have motivated the development of spatially resolved transcriptomics technologies. These technologies come with trade-offs; for *in situ* capture methods such as 10x Genomics Visium, for example, the gene expression captured from a single spot may come from multiple cells, and thus computational methods for integrating single-cell data to deconvolve cell type proportions have gained interest. To date, these have required sample-matched datasets from each modality, whereas in most cases these are not available. Taking inspiration from and building on CellDART, a method that uses domain adaptation to address this problem by treating reference scRNA-seq data as source domain data, we further investigate the performance of 3 domain adaptation methods (ADDA, DANN, and Deep CORAL) and a Bayesian method (RCTD) on 3 different pairs of source and target datasets (dlPFC, PDAC, and a "gold standard" mouse cortex dataset). We used metrics across 3 levels of evaluation: (a) performance on source domain scRNA-seq derived "pseudo-spots", (b) performance in mapping inputs from both domains to a common distribution, and (c) performance on real spatial transcriptomics spots. By integrating these 3 datasets and 3 levels of metrics, we ensured that our results were robust and broadly applicable. Upon performing hyperparameter tuning for all three models across all three sets, and evaluating the models, we were unable to conclusively determine whether any method performed better or worse. We found that CellDART remained a good performer, that it was difficult to train DANN, and that ADDA's performance is limited by not remaining consistent for specific samples. Future directions include improving on CellDART, integrate cycle-consistent loss for ADDA, and implementing unsupervised domain adaptation validation methods.

Abrégé

L'intérêt pour la communication intercellulaire locale au sein des tissus dans des domaines tels que la biologie du développement, les neurosciences et l'oncologie a motivé le développement de technologies transcriptomiques spatiales. Ces technologies s'accompagnent de compromis; pour les méthodes de capture in situ telles que 10x Genomics Visium, par exemple, l'expression génique capturée à partir d'un seul point peut provenir de plusieurs cellules, et c'est pourquoi les méthodes computationnelles d'intégration des données de cellules uniques pour déconvoluer les proportions de types cellulaires ont suscité un intérêt croissant. Jusqu'à présent, ces méthodes nécessitaient des ensembles de données correspondant à des échantillons de chaque modalité, ce qui n'est pas le cas dans la plupart des cas. En s'inspirant de et s'appuyant sur CellDART, une méthode qui utilise l'adaptation de domaine pour résoudre ce problème en traitant les données scRNA-seq de référence comme des données de domaine source, nous avons étudié les performances de 3 méthodes d'adaptation de domaine (ADDA, DANN et Deep CORAL) et d'une méthode bayésienne (RCTD) sur 3 paires différentes d'ensembles de données source et cible (dlPFC, PDAC et un ensemble de données de cortex de souris « étalon-or »). Nous avons utilisé des mesures à trois niveaux d'évaluation: (a) performance sur des « pseudo-points » dérivés de scRNA-seq dans le domaine source, (b) performance dans la mise en correspondance des entrées des deux domaines avec une distribution commune, et (c) performance sur de véritables points de transcriptomique spatiale. En intégrant ces trois ensembles de données et ces trois niveaux de mesure, nous nous sommes assurés que nos résultats étaient robustes et largement applicables. Après avoir effectué le réglage des hyperparamètres pour les trois modèles dans les trois ensembles et évalué les modèles, nous n'avons pas été en mesure de déterminer de manière concluante si l'une ou l'autre des méthodes était plus performante ou moins performante que l'autre. Nous constatons que CellDART reste

performant, qu'il est difficile d'entraîner DANN et que les performances d'ADDA sont limitées par le fait qu'elles ne restent pas cohérentes pour des échantillons spécifiques. Les orientations futures comprennent l'amélioration de CellDART, l'intégration d'un fonction objectif cohérente avec le cycle pour ADDA et la mise en œuvre de méthodes non supervisées de validation de l'adaptation de domaine.

Acknowledgements

I would like to thank my supervisor, Professor Amin Emad, for his guidance, feedback, and support as well as his engagement in the work that went into this thesis. I would also like to thank the Department of Electrical Computer Engineering at McGill and the Natural Sciences and Engineering Research Council of Canada for funding my research through the Graduate Excellence Fellowships and Canada Graduate Scholarships–Master's, respectively. I would finally like to thank my family, friends, and lab partners for their unwavering support and help throughout my degree.

Table of Contents

Abstract i
Abrégé ii
Acknowledgements iv
Table of Contents v
List of Tables vii
List of Figures x
List of Abbreviations xii
Contribution of Authors xiv
1. Introduction 1
2. Literature Review 4
2.1 Overview 4
2.2 Molecular biology background 4
2.2.1 Gene expression and intercellular interactions 4
2.2.2 Transcriptomics 5
2.2.3 Applications of spatial transcriptomics 9
2.2.4 Domain discrepancy in transcriptomics 10
2.3 Computational background 12
2.3.1 Integration of single-cell and spatial transcriptomics data 12
2.3.2 Domain adaptation 13
3. Methodology 21
3.1 Overview 21
3.2 Data and data processing 22
3.2.1 Datasets, cleaning, and pre-processing 22
3.2.2 Data splitting and marker gene selection 25
3.2.3 Pseudo-spot generation 27
3.2.4 Log transforms and normalization 29
3.3 Evaluation 30
3.3.1 Source domain performance 31
3.3.2 Variance between source and target 31

3.3.3 Target domain performance 35
3.4 Models and training 39
3.4.1 Reimplemented CellDART 43
3.4.2 ADDA 45
3.4.3 DANN 48
3.4.4 CORAL 51
4. Results 54
4.1 Overview 54
4.2 Overall performance 54
4.2.1 Performance on source domain data 54
4.2.2 Evaluation of domain adaptation 55
4.2.3 Evaluation of deconvolution of real spots 57
4.3 Selected hyperparameters 59
4.4 Adversarial training results 61
4.5 Effect of domain adaptation 64
4.6 Comparison with original CellDART 66
5. Discussion and Future Work 68
5.1 Analysis of results 68
5.1.1 Gradient reversal layer causes poor adversarial training stability 69
5.1.2 Performance of ADDA wholly a result of initialization of target encoder 70
5.1.3 Remark on domain discrimination performance 71
5.2 Future work 72
6. Conclusion and Summary 75
References 76

List of Tables

Table 3.1: Mappings from source scRNA-seq data to target spatial transcriptomics regions used to evaluate cell type proportions for spatial PDAC data. 37

Table 3.2: Mappings of scRNA-seq source dataset cell types to target spatial transcriptomic cell types for the mouse cortex dataset. Where multiple classes were merged, individual cell types are separated by a comma. For the source cell type, abbreviations used by the paper authors are used; for definitions, refer to the original paper {/Peng, 2019 #240/}. Greyed out cells indicate no mapping to the other set. 37

Table 3.3: Common data hyperparameters used for all model types. 41

Table 3.4: Specifications of CellDART. Where not specified, configurations values are either default PyTorch values or hyperparameters sets in Table 3.5; hyperparameter placeholders are indicated with an underline. 43

Table 3.5: Reimplemented CellDART hyperparameters validated. α, as used in CellDART, is a scaling factor for the discriminator classification loss Ld, like λ in DANN. αLR is a scaling factor for the discriminator's learning rate against the main learning rate, with 5 being reported as a good value by CellDART's authors. 44

Table 3.6: Specifications of ADDA. Where not specified, configurations values are either default PyTorch values or hyperparameters sets in Table 3.7; hyperparameter placeholders are indicated with an underline. 46

Table 3.7: Validated ADDA hyperparameters. de is the dimension of the embedding at the output of the encoder. αLR is a scaling factor for the discriminator's learning rate vs. the main learning rate. Kd is the discriminator loop factor. 47

Table 3.8 Specifications of DANN. Where not specified, configurations values are either default PyTorch values or hyperparameters sets in Table 3.9; hyperparameter placeholders are indicated with an underline. 50

Table 3.9: DANN hyperparameters validated. de is the dimension of the embedding at the output of the encoder. λ is a scaling factor for the discriminator classification loss Ld. α is a scaling factor for the gradient as it passes through the GRL. Higher values of α mean larger encoder gradients relative to those passed from the discriminator. 51

Table 3.10: CORAL specifications. Where not specified, configurations values are either default PyTorch values or hyperparameters sets in Table 3.11; hyperparameter placeholders are indicated with an underline. 52
Table 3.11: CORAL hyperparameters validated. de is the dimension of the embedding at the output of the encoder, while (λe, $\lambda logits$) are CORAL loss weighting parameters for the encoder output and the predictor output, respectively. 52
Table 4.1: Test performance of each model on synthetic scRNA-seq pseudo-spots. Values are mean cosine distance across bootstraps, lower values being better; standard deviation across 5 bootstraps are shown in braces. 54
Table 4.2: RF50 scores of each model by dataset using embeddings on test samples from both source and target, except for PDAC, where training samples were used. Values are expressed as mean accuracies of the random forest model on a holdout set of 20%. Standard deviation across 5 bootstraps is expressed in braces. A value closer to 0.5 is better. 55
Table 4.3: miLISI scores between source and target test samples from both source and target. except for PDAC, where training samples were used. Values are expressed as the mean miLISI across 5 bootstrap models, while standard deviation is indicated in braces. A value closer to 2 is better. 55
Table 4.4: Performance on real spatial transcriptomics spots. Standard deviation, where available, is indicated in branches. The best performing deep UDA methods are indicated with an underline, while the best including the RCTD baseline is indicated in **bold**. 58
Table 4.5: CellDART hyperparameters selected via random search. α, as used in CellDART, is a scaling factor for the discriminator classification loss Ld, like λ in DANN. αLR is a scaling factor for the discriminator's learning rate against the main learning rate. 59
Table 4.6: ADDA hyperparameters selected via random search. ADDA hyperparameters. de is the dimension of the embedding at the output of the encoder. αLR is a scaling factor for the discriminator's learning rate vs. the main learning rate. Kd is the discriminator loop factor. 59
Table 4.7: DANN hyperparameters selected via random search. de is the dimension of the embedding at the output of the encoder. λ is a scaling factor for the discriminator classification loss Ld. α is a scaling factor for the gradient as it passes through the GRL.

Higher values of α mean larger encoder gradients relative to those passed from the discriminator. 60
Table 4.8: CORAL hyperparameters selected via random search. de is the dimension of the embedding at the output of the encoder, while (λe, $\lambda logits$) are CORAL loss weighting parameters for the encoder output and the predictor output, respectively. 61
Table 4.9: Performance of CellDART and ADDA on source pseudo-spots comparing pretraining only and after domain adaptation stage. Values are cosine distance, where lower is better. . 64
Table 4.10: Performance of CellDART and ADDA comparing RF50 between source and target data before and after domain adaptation stage. Values are expressed as discrimination accuracies, where lower is better. Test samples were used for both sets, except for PDAC where training samples were used as target samples. 64
Table 4.11: Performance of CellDART and ADDA comparing miLISI scores between source and target data before and after domain adaptation stage. Test samples were used for both sets, except for PDAC where training samples were used as target samples. Perp.=perplexity, higher is better. 65
Table 4.12: Performance of CellDART and ADDA comparing miLISI scores on real spots before and after domain adaptation stage. Test samples were used for both sets, except for PDAC. 65
Table 4.13: Comparison of test performance between our reimplementation of CellDART with the original code, using the selected hyperparameters shown in Table 4.5. In each dataset and metric, the better performing score of the two is bolded and underlined. Standard deviation is shown in braces. 66

List of Figures

Fig. 2.1: Comparison of mouse cortex pseudo-spots generated from scRNA-seq and real 10x Genomics Visium spots after independent log transforms and standardization, shown using PCA and UMAP. The scRNA-seq GEx was obtained from {/Tasic, 2018 #242/} and the spatial dataset was downloaded from the 10x Genomics database {/10x Genomics, #266/} via the scanpy (v. 1.8.2) Python package {/Wolf, 2018 #235/}. Both datasets were gathered separately. SC=scRNA-seq, ST=spatial transcriptomics. 11

Fig. 2.2: Diagram of the DANN architecture showing the adversarial aspect introduced by the GRL. Forward propagation is shown with orthogonal arrows, while backpropagation is shown with curved arrows. α is a scaling parameter for the encoder gradient relative to the discriminator gradient. 16

Fig. 3.1: Overall methodology, excluding hyperparameter tuning, showing processing, marker selection, pseudo-spot generation, data splitting, and when each dataset is used when training or under each level of evaluation. The dlPFC datasets for both source and domain are shown for this example, where a single slide each is held out as validation and test. The x-axes (genes) of the GEx heatmaps before and after marker selection are not to scale, as the actual number of selected genes was much lower than the total number. The neural networks shown are a simplification and differ depending on the specific model architecture used. The architectures weights for the top and bottom models are the same between source and target, indicated by the dashed lines; in reality, this is also differs depending on the specific model used. 21

Fig. 4.1: Comparison of embeddings produced using CellDART and CORAL. Target slide was test slide 151675, and source data were the test set of pseudo-spots. 57

Fig. 4.2: Accuracy of discriminator and encoder of ADDA; it was not possible to separately measure the accuracy of the encoder during evaluation and so is left out of the validation subfigure. 62

Fig. 4.3: Performance shown as cosine distance (lower=better) of ADDA on Spotless "gold standard 1" mouse cortex spots at each epoch of adversarial training when extending to 100 epochs, performance before training is shown at epoch "-1", while the 20th epoch (epoch 19) that was the final model is indicated with a vertical line. 63

Fig. 4.4: Accuracy at each iteration (end of epoch for validation) on domain discrimination task for final DANN model on Spotless “gold standard 1” mouse cortex dataset. Selected epoch 372 indicated with a vertical bar. Note that as training of the encoder of the encoder and discriminator were not performed separately, we were unable to individually track their accuracies. 63
Fig. 4.5: Potential mode collapse in the generator of DANN. This can be seen by the accuracy for both domains steadily increasing over training, indicating overfitting of the discriminator. 64

List of Abbreviations

2D: 2-Dimensional

ADDA: Adversarial Discriminative Domain Adaptation

ANOVA: Analysis of Variance

AUROC: Area Under the Receiver Operating Characteristic Curve

BCE: Binary Cross-Entropy

CAF: Cancer-Associated Fibroblast

CCIN: Cell-Cell Interaction Network

cLISI: Cell Type Local Inverse Simpson's Index

CORAL: CORrelation Alignment

CyCADA: Cycle-Consistent Adversarial Domain Adaptation

DANN: Domain-Adversarial Neural Network

dlPFC: Dorsolateral Prefrontal Cortex

DNA: Deoxyribonucleic Acid

DSTG: Deconvoluting Spatial Transcriptomics Data through Graph-Based Convolutional Networks

ELU: Exponential Linear Unit

ESC: Embryonic Stem Cell

FISH: Fluorescence *In Situ* Hybridization

FOV: Field of View

FPR: False Positive Rate

GAN: Generative Adversarial Network

GCN: Graph Convolutional Network

GEx: Gene Expression

GRL: Gradient Reversal Layer

GRN: Gene Regulatory Network

iLISI: Integration Local Inverse Simpson's Index

ISS: *In Situ Sequencing*

KL-Divergence: Kullback–Leibler Divergence

LISI: Local Inverse Simpson's Index

MDD: Major Depressive Disorder

MERFISH: Multiplexed Error-Robust Fluorescence *In Situ* Hybridization

miLISI: Median Integration Local Inverse Simpson's Index

mRNA: Messenger Ribonucleic Acid

MSc: Master of Science

PCA: Principal Component Analysis

PDAC: Pancreatic Ductal Adenocarcinoma

RCTD: Robust Cell Type Decomposition

ReLU: Rectified Linear Unit

RNA: Ribonucleic Acid

RNA-seq: Ribonucleic Acid Sequencing

ROC: Receiver Operating Characteristic

scRNA-seq: Single-Cell Ribonucleic Acid Sequencing

seqFISH: Sequential Fluorescence *In Situ* Hybridization

snRNA-seq: Single-Nucleus Ribonucleic Acid Sequencing

STARmap: Spatially-Resolved Transcript Amplicon Readout Mapping

TCR: T Cell Receptor

TF: Transcription Factor

TME: Tumour Microenvironment

TPR: True Positive Rate

UDA: Unsupervised Domain Adaptation

UMAP: Uniform Manifold Approximation and Projection

Contribution of Authors

This thesis represents the culmination of my work toward the completion of my Master of Science (MSc) degree in the Department of Electrical and Computer Engineering at McGill University. All the content within this thesis is my contribution under the supervision of Professor Amin Emad. My contributions to each chapter are as follows:

1) Introduction: In this chapter, I briefly state the background and motivation in cell type deconvolution of spatial transcriptomics data by integrating scRNA-seq as source domain data, as well as the objectives of the thesis.
2) Literature Review: In this chapter, I explain the biology background necessary to understand the context of and challenges we seek to address in this work, then review the literature embodying the computational applications in this field and that we sought to apply.
3) Methodology: In this chapter, I explain the protocols I used to conduct my investigation, including the datasets and data processing, evaluation of models, the models used, and training.
4) Results: In this chapter, I report the results of my investigation, and comment on any particularly interesting results.
5) Discussion and Future Work: In this chapter, I comment on the implications of the results, interesting insights learned, and possible future directions.
6) Conclusion and Summary: In this chapter, I summarize the findings of my investigations and their implications in the context of the objectives outlined in Chapter 1.
7) Conclusion and Summary: I summarize the findings of my investigations and their implications in the context of the objectives outlined in Chapter 1.

1. Introduction

The fundamental mechanism by which cells in a multicellular organism differentiate into different cell types, perform their respective functions, and interact is through variations in the activity of individual genes (and their protein products), referred to as gene expression (GEx) [1, 2]. High throughput technologies such as single-cell RNA sequencing (scRNA-seq) and single-nucleus RNA sequencing (snRNA-seq), which provide GEx of individual cells, have led to leaps in our understanding of immunology [3], oncology [4], and developmental biology [5], paving the way for myriad promising applications in human health. Despite these advancements, a major blind spot of single-cell transcriptomics is that the locations of individual cells are lost. This has motivated the development of spatial transcriptomic methods [6].

One example where spatial transcriptomics shows potential is in cell-cell interaction networks (CCINs) and their role in the development of biological tissues. Cell differentiation is regulated by transcription factors (TFs) [7] through both internal gene regulatory mechanisms (represented by gene regulatory networks or GRNs) and external crosstalk between cells in their environment [8] (represented by CCINs. The integration of these networks is necessary to comprehensively understand the biological mechanisms involved in cell differentiation, which provides a basis for the study of pregnancy, birth complications, and treatments, in addition to laying the groundwork for stem cell research. Significant progress has been made in understanding internal GRNs using bulk and single cell RNA sequencing (RNA-seq) technologies [5, 9-11], however without spatial information, their use in constructing CCINs along with other determinants of cell fate is limited, especially with respect to how cells organize themselves into tissues [12-15]. Another key area where spatial transcriptomics shows great potential is in understanding of the tumour microenvironment (TME) to elucidate structural factors in cancer. A recent area of focus in TME

research that can be elucidated using spatial transcriptomics is the role of stroma (a distinct tissue region that forms in aggressive cancers such as pancreatic ductal adenocarcinoma (PDAC) [16]), which has been implicated in enhanced angiogenesis (the formation of new blood vessels), immune escape, treatment resistance, and metastasis in these cancers [17-20].

Spatial transcriptomics methods, however, come with their own trade-offs when compared with methods like scRNA-seq. Image-based technologies such as multiplexed error-robust fluorescence *in situ* hybridization (MERFISH) [21] and sequential fluorescence *in situ* hybridization (seqFISH) [22] can capture GEx at single-cell resolution but for only a limited number of genes (typically in the 100s for seqFISH); these genes must be known *a priori* and thus have limited use in discovery of new pathways in GRNs and CCINs. On the other hand, RNA-seq-based technologies such as 10x Genomics Visium [23] and the emerging Stereo-seq [24] can measure GEx for all genes (~20,000 in humans). The measurements provided by these technologies, however, correspond to spots, not cells, which in the case of Visium comprise of tens of cells of (potentially) different types, and even in emerging higher-resolution technologies like Stereo-seq, spots will still potentially measure expression from multiple different cells. Compared with scRNA-seq, whose specificity allows for the identification and measurement of GEx at a single cell resolution [25] but have spatial organization as a blind spot, these methods fall short in providing a comprehensive cell-specific view of the tissue.

For a more comprehensive picture of cell-cell interactions and to identify cell type-specific factors controlling cell development and especially differentiation, integration of the two modalities has been of interest and has shown potential in revealing insights that would otherwise require targeted experiments [26]. Of interest to us is the integration of cell-type information from scRNA-seq and snRNA-seq with RNA-seq-based spatial transcriptomics using novel

computational methods, where a single-cell reference set is used to infer cell type proportions within spatial transcriptomics spots. The problem thus far has been largely framed around integrating datasets sourced from the same tissue sample, and thus integrating independently collected out-of-distribution scRNA-seq data remains a challenge [14]. A recent method that attempts to address this shortcoming as well as the inherent differences of the two modalities is CellDART [27], which uses unsupervised domain adaptation (UDA), an area of machine learning involving the transfer of learned knowledge to new contexts where labelled data is limited.

Although the authors of CellDART [27] had demonstrated the potential of UDA, the analysis, training, and performance were characteristic of a singular proof-of-concept rather than a rigorous and in-depth analysis of UDA as a whole. Building on their work, we aimed to further explore and evaluate the use of UDA in cell type deconvolution by leveraging advancements developed in the fields of computer vision and natural language processing [28-31]. The main contributions of our work are the development and assessment of 3 deep UDA models. We focused on cases where single-cell source data was collected separately from the target spatial data. To do this, we compared our methods against a baseline non-UDA and non-deep learning model [32] found in a benchmarking study to be a top performer when using source and target data collected from the same source [33] and finally CellDART as our UDA baseline. We additionally investigated whether domain adaptation is a crucial factor in their performance, firstly by evaluating their performance in the source domain, secondly by evaluating each model's ability to map each domain to a common distribution, and thirdly by evaluating their performance on real spatial transcriptomics data. To ensure that our results were applicable, we tested using three pairs of datasets.

2. Literature Review

2.1 Overview

In this chapter, we first explain the biology background necessary to understand the context of and challenges we seek to address in this work, then review the literature embodying the computational applications in this field. In Section 2.2, we begin by explaining GEx and the role it plays in the human body. We then provide an overview of relevant transcriptomics technologies used to measure it, the advantages and drawbacks of each, and finally the applications of spatial transcriptomics. In Section 2.3, we provide an overview of techniques used to integrate single-cell and spatial transcriptomics data, as well as a background on UDA, which we used in this work.

2.2 Molecular biology background

2.2.1 Gene expression and intercellular interactions

Deoxyribonucleic acid (DNA), ribonucleic acid (RNA), and proteins are the 3 types of macromolecules carrying the sequential information used to build the structures and perform the functions necessary for life in biological systems. The central dogma of molecular biology states that this information cannot flow out of proteins; the information encoding its structure must come from DNA or RNA [34]. In multicellular organisms, the genome, which is all genetic information including the more than 20,000 protein-coding genes in humans, are encoded in DNA [35], and is identical and shared between nearly every cell in the body; in humans, this has been completely sequenced [36, 37]. Proteins, on the hand, form most of the structure and perform most of the functions of a cell. The general understanding is that the main processes by which genes become proteins occur in 2 steps: (a) *transcription*, whereby genes in DNA are copied into individual

messenger RNA (mRNA) strands, and (b) *translation*, through which a the sequence of nucleotides in mRNA is used to chain together amino acids into proteins [1, 2].

For individual cells to present and behave differently (despite sharing a common genome), not all genes corresponding to various functions are expressed equally or spatially across the cells within the body or over time. The regulation of a cell's GEx is primarily controlled at the point of transcription through the modulation of the number of mRNA transcripts present and thus how many of the corresponding proteins will be produced [1]. The factors controlling transcription themselves naturally have their origins in genes, and how each gene affects and regulates one another can be modelled as GRNs. As proteins are the primary functional and structural components of cells, this process therefore controls the differentiation of stem (undifferentiated) cells into various types of somatic (forming the body) cells [38] as well as regulating the function of individual cells of the same type in different tissues [39] and playing a key role in their dysfunction in the case of many diseases [40].

In addition to the regulation of GEx within a cell, inter-cellular interaction also plays a role. Differentiation and organization of embryonic stem cells (ESCs) into different germinal layers, organs, tissues, and somatic cell types requires substantial and complex CCINs to signal to cells when, where, and into what they differentiate [41]. Similarly, dysfunction of genes within CCINs play a key role in diseases such as in cancer [8, 16-20, 42].

2.2.2 Transcriptomics

2.2.2.1 A brief overview of transcriptomics

Transcriptomics, or the measurement of RNA transcripts present in a cell or tissue, has, over the last few decades, seen great improvements in both the capabilities of available technologies and the discoveries and advancements they have produced. In recent years, this has been in no

small part due to the supplantation of microarray-based technologies (which uses arrays of spots each corresponding to known transcripts) by RNA-seq and its derivatives [43, 44].

One particularly fruitful form of RNA-seq has been scRNA-seq. In contrast with bulk RNA-seq, which measures the sum of all transcripts in a sample tissue, scRNA-seq and other single-cell technologies measure RNA on a per-cell basis. The ability to differentiate GEx from individual cells using scRNA-seq and snRNA-seq data has resulted in innumerable discoveries in diverse biological and biomedical contexts. Unsupervised clustering methods are often used to group cells in a sample by their GEx profiles; these clusters can then be used to consistently identify and classify individual cells by cell type based on known expression of genetic markers for an associated cell type, or to even identify different cell states within a given cell type [45].

Knowledge of how different cell types each in different cell states interact in terms of GEx has allowed for a better understanding of the complex regulatory networks and interactions in the immune system [3]. This includes the discovery of the diversity, dynamic nature, and trajectories of dysfunctional T cells within tumours which presents opportunities for better immunotherapy [46], as well as shedding light on T cell responses based on the great variety of T cell receptors (TCRs) that may target antigens [47]. In addition, analysis of heterogeneity of GEx across cell types in the TME allows for the identification of new, specific drug targets, and analysis of heterogeneity across patients in concert with clinical profiling can allow for targeted combination therapies [4]. Another area in which scRNA-seq has proven useful has been in analysis leveraging cell type-specific pathways to determine the regulatory networks, including GRNs and CCINs, driving embryonic development, such as in lungs [5].

2.2.2.2 Spatial transcriptomics

While single-cell technologies have demonstrated their usefulness in the discovery of GRNs and CCINs, local effects are also important when analyzing co-expression of signaling molecules (ligands) and their receptors in cell-cell communication. Ligand-receptor pairs may only interact within the same compartments, and within a limited distance. These effects are not considered with scRNA-seq, and thus have motivated the development of spatially resolved transcriptomics, or spatial transcriptomics [6, 14, 15, 48]. The most common methods can be roughly organized into three major categories: (a) fluorescent *in situ* hybridization (FISH), (b) *in situ* sequencing (ISS), and (b) *in situ* capture, also known as spatial barcoding [49].

FISH-based approaches include seqFISH [21] and MERFISH [22], and use specialized nucleic acids to target and bind with a specific transcript within a histological section. A histological (tissue) section is a thin slice of tissue fixed on a microscopy slide to be stained, imaged, and used for any additional processing and data collection [50, 51]. Once mounted, they remain together during any imaging or transcriptomic analysis, so we use the term slide to refer to both. An example of spatial transcriptomics slides is shown in Fig. 3.1 for 10x Genomics Visium (discussed later in this section). These probes are fluorescent, and through microscopy imaging, GEx can be localized at a subcellular level. Although they provide high spatial resolution and can be multiplexed for thousands of genes in the case of MERFISH, they are still limited compared to RNA-seq based technologies which do not require transcripts to be set beforehand.

ISS-based approaches include spatially-resolved transcript amplicon readout mapping (STARmap) [52], as well as the original ISS method [53]. Broadly speaking, transcripts are fixed and then reverse-transcribed from mRNA to a special form of DNA made of fluorescent markers, allowing the sequence to be directly imaged. Unlike FISH-based approaches, variations in

sequences can be detected, but, due to potential crowding and the requirement of targets to be pre-selected, this method is limited in terms of the number of genes that can be measured. A more recent approach, STARmap, binds the transcribed sequences to a polymer added to the tissue called a hydrogel; this preserves the structure of the tissue and the positions of in the next step when the tissue is stripped from the gel. A transparent structure is left behind, allowing for increased sensitivity during imaging and thus allows for more unique transcripts to be sequenced.

In contrast to the first two classes of spatial transcriptomics methods, *in situ* capture or RNA-seq-based methods involve first capturing the position of transcripts, then sequencing transcripts afterward using existing bulk RNA-seq. The original method was named "Spatial Transcriptomics" by Ståhl *et al.* [50] and will hereafter be referred to as the Ståhl method to distinguish it from the field of spatial transcriptomics itself. The basic premise of each of these methods roughly involves applying a grid of spots or beads to a tissue, each spot containing probes with a spot-specific barcode, which binds to transcripts. When RNA-seq is performed, these barcodes can be recovered and thus transcripts can be associated with a particular spot. While this and derivative methods provide many of the advantages of scRNA-seq, namely whole-genome coverage, measurements are constrained by the structure of the grid of spots, with each spot potentially containing multiple or no cells, and some cells' GEx split across multiple spots. Improvements in resolution have been made using advancements in capture technology; while the Ståhl method allows for up to ~1,000 spots with ~30-70 cells each, 10x Genomics' Visium allows for up to ~5,000 spots with ~1-10 cells each [23], and, more recently, Stereo-seq has demonstrated cellular to potentially sub-cellular resolution for up to ~10^{15} spots [24]. Despite these increases in resolution, spots are still not necessarily matched with single cells, presenting a challenge when studying inter-cellular differences and interactions within a tissue sample.

2.2.3 Applications of spatial transcriptomics

2.2.3.1 Developmental Biology

The differentiation of ESCs into somatic cells, and their organization in tissues, are driven both by internal GRNs as well as external CCINs. As single-cell transcriptomic methods allow for identification of cell types, interactions between cell types may be studied by analyzing expression of ligand-receptor pairs, crucial to cell signaling, as well as by integration and augmentation of other molecular biology data [48]. This process has shed light on regulatory networks in the embryonic development of mouse lungs [5] and the development of the interface between the placenta and uterus [54].

One example where spatial transcriptomics shows potential is in CCINs and their role in tissue development. Differential GEx and therefore cell differentiation is regulated by transcription factors (TFs) [7] through both internal gene regulatory mechanisms and external crosstalk between cells in their environment [8], represented by CCINs and GRNs, respectively, and the integration of both is necessary to understand those biological mechanisms. This provides a basis for the study of pregnancy, birth complications, and treatments, in addition to laying the groundwork for stem cell research. By identifying candidate interventions for influencing cell development, better protocols for differentiating stem cells into desired cell types can be developed and used in therapies including regenerative medicine [55].

2.2.3.2 Tumour microenvironment

The physical layout and structure of tumours has long been known to be a key factor in how they form, resist immune responses, and grow. For example, tumour growth is facilitated in part by how it encourages angiogenesis (the formation of new blood vessels). While this has been known since at least the 1990s [42], the underlying mechanisms have, until recently, remained

elusive. One recent area of focus in TME research has been in the role of the stroma, a structural region serving as the "support" of a tumour present in certain highly aggressive tumours such as PDAC [16], how cell-cell communication influences its development [17], and how signalling within the stroma influences cancer-enhancing factors such as angiogenesis [19]. In addition, it has been demonstrated that cancer-associated fibroblasts (CAFs) in the stroma contribute to tumours' ability to suppress an immune response and reinforce tumour cell proliferation and metastasis [18, 56]. Furthermore, it has been shown that, due to both the structure and signalling environment within the stroma, immune cells become dysfunctional, giving the tumour free reign in its growth and malignancy [20]. It is both the structural and cell-signaling aspects of the stroma which make spatial transcriptomics promising in cancer research.

2.2.3.3 Neuroscience

An extension of the developmental biology applications of spatial transcriptomics, as discussed in Section 2.2.3.1, is specifically in the development of the mammalian brain. In particular, the roles of GRNs and CCINs in regulating the formation of cerebral cortex functions and architectures, the analysis of different cell types in how they are physically connected, and the interactions in the local area around amyloid plaques in Alzheimer's are all areas where advancements in spatial transcriptomics technologies show promise over non-spatial techniques such as scRNA-seq and snRNA-seq [6].

2.2.4 Domain discrepancy in transcriptomics

GEx can be highly variable across individuals, technologies, within the same tissue or disease state, across research groups, and even when repeated within the same research groups due to slightly different conditions and protocols; these are known as batch-effects [4, 57, 58]. These are particularly problematic when differences in distribution of data across different batches

confounds results, leading to spurious type I errors; these effects can be pernicious and difficult to account for [59]. To help combat these issues, a body of research on methods and best practices in correcting batch-effects in transcriptomics has emerged [60-62].

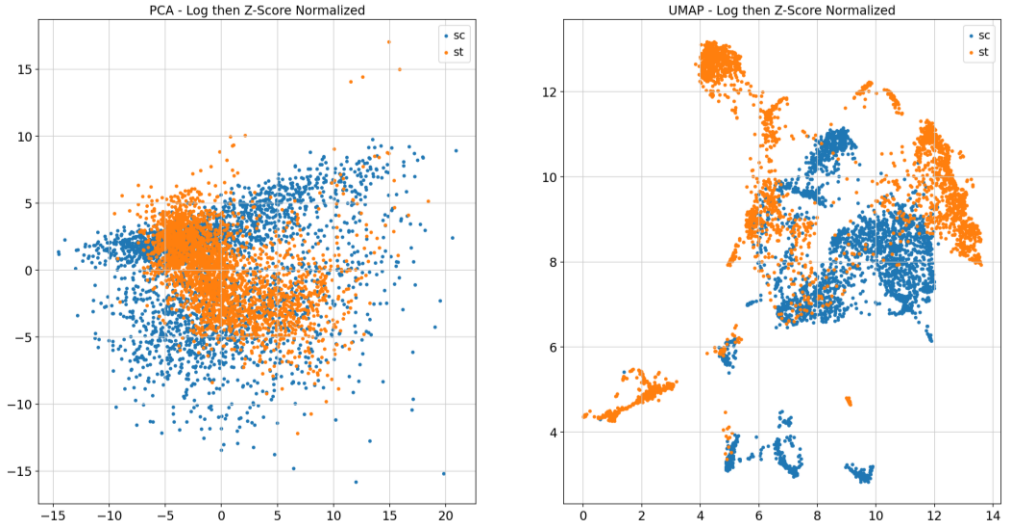

Fig. 2.1: Comparison of mouse cortex pseudo-spots generated from scRNA-seq and real 10x Genomics Visium spots after independent log transforms and standardization, shown using PCA and UMAP. The scRNA-seq GEx was obtained from [63] and the spatial dataset was downloaded from the 10x Genomics database [64] via the scanpy (v. 1.8.2) Python package [65]. Both datasets were gathered separately. SC=scRNA-seq, ST=spatial transcriptomics.

Of particular interest to us in our work are the differences in distribution between GEx data between those derived from each of spatial transcriptomics and single-cell transcriptomics, which can be compounded by batch effects when using independently collected datasets [14]. To demonstrate these differences, we generated pseudo-spots from scRNA-seq by randomly sampling cells from different cell types ($n_{mix} = 8$; see Section 3.2.3 for pseudo-spot generation process) and aggregating their scRNA-seq profiles. Even after separately preprocessing, log-transforming, and normalizing the real spatial spots and generated pseudo-spots, their joint embedding using uniform manifold approximation and projection (UMAP) [66] and principal component analysis (PCA) (Fig. 2.1) showed clear separation. Furthermore, when using 80% of the data as a training set, using as *sci-kit learn*'s (v. 1.1.2) `RandomForestClassifier` and `SVC` [67] as models with the default settings, these models achieved 100% accuracy in predicting the origin domain on the 20%

holdout data, indicating the need for transfer learning techniques such as UDA for any model trained on one technology to be able to generalize to another [28].

2.3 Computational background

2.3.1 Integration of single-cell and spatial transcriptomics data

Most techniques which integrate scRNA-seq and spatial transcriptomics data broadly fall into one of two categories: (a) mapping-based techniques, whereby individual single-cell samples are mapped to a set of coordinates on the spatial transcriptomics slide, and (b) deconvolution techniques, which break down spots by their constituent cell types and/or their individual contributions to the GEx [14, 33].

Mapping-based approaches are used when spatial GEx data has single-cell resolution but are only available for a finite, predetermined set of genes, such as when derived from imaging-based methods of spatial transcriptomics [21, 22, 49, 52, 68]. By mapping scRNA-seq cells and/or their GEx to locations on a slide, the effective number of genes that are spatially resolved is increased [14]; this can be thought of as data imputation. One approach is to map both single-cell sets into a shared space and batch correcting them using a tool such as Harmony [69]. Harmony does this by first reducing both sets to the same number of dimensions using PCA, then clustering the data in this space under two constraints, both using a method referred to by the Harmony authors as local inverse Simpson's index (LISI). The first is to quantify the mixing or integration of two datasets, using integration LISI (iLISI), which is a measure of the effective number of different datasets surrounding a particular cell. The second is to, also using LISI, minimize the number of cell types dominant within a cluster; they refer to this measure as cell type LISI (cLISI). As we use iLISI as a method to evaluate how well our models integrated single-cell and spatial transcriptomics data, we further explain and discuss LISI in Section 3.3.2.

Deconvolution techniques come in several approaches. One approach is to fit a distribution over the gene transcript counts of the scRNA-seq data and create a model of cell type proportions based on that, then apply it to the spatial GEx data. Robust cell type decomposition (RCTD), for example, uses a Bayesian hierarchical model, modelling gene counts as a random variable arising from a Poisson distribution as a prior, conditioned on a log-linear function based on cell type, among other factors [32]. This model is notable in that, in a recent analysis of 11 deconvolution methods using datasets including actual cell types as ground truth as well as synthetic data (the analysis pipeline and datasets are referred to as Spotless), it was one of the top two performing methods and the most stable and scalable of the two. It should be noted, however, that datasets used in this analysis were sample-matched i.e., both datasets came from the same biological sample, and thus may not fully capture a method's out-of-distribution performance [33].

Deep-learning based methods include "deconvoluting spatial transcriptomics data through graph-based convolutional networks" (DSTG) [70] and CellDART [27], the former of which we will briefly describe here, and the latter of which we describe in Section 2.3.2. DSTG generates pseudo-spots (the aggregates of single cells simulating *in situ* capture spots). Then, it constructs a heterogenous graph integrating the real and pseudo spots connecting them based on similarity. Finally, it trains a graph convolutional network (GCN) which propagates cell types along the graph in its output and is trained to predict cell type proportions. In this way, the cell type proportions of real spots are imputed via the graph.

2.3.2 Domain adaptation

All the techniques mentioned above apply varying degrees of regularization. Regularization is a body of techniques that seek to reduce a model's variance or generalization error (i.e., the relative loss in performance when comparing the training data and real world data [71]). In

addition, many methods seek to reduce mismatch using "batch effect correction" techniques. These, however, ignore the fact that technologies such as scRNA-seq and *in situ* capture technologies are inherently different in how data are sampled and collected, despite both being based on RNA-seq. As we explored in Section 2.2.4, there can be a great discrepancy in the distribution of GEx data between those originating from single-cell and spatial transcriptomics methods. As a result, cell type deconvolution can be considered an out-of-distribution problem, where spatial transcriptomics data (for which we seek to infer cell type proportions) are the *target* domain and are of a different distribution than the single-cell data used to train a model (the *source* domain).

To specifically address the issue of out-of-distribution inference in machine learning, a body of transfer learning methods are used. These methods seek to model how humans learn, applying a "far-transfer" of related skills and techniques acquired over a lifetime to new problems and contexts. While supervised transfer learning requires the use of at least some labels of target domain data for training, unsupervised domain adaptation or UDA addresses this issue by training using only the labels of source domain data and uses target domain data only as inputs to compare the source domain data against [28]. We explore three such UDA methods: (a) CORrelation Alignment (CORAL) [72], specifically its application to deep neural networks called Deep CORAL [31], (b) Domain-Adversarial Neural Network (DANN) [30], and (c) Adversarial Discriminative Domain Adaptation (ADDA) [29], each of which we describe in this section. We further discuss CellDART [27], a UDA method that specifically addresses cell type deconvolution using single-cell data as source domain data.

Deep CORAL works by incorporating a loss function called CORAL into the overall training cost function, scaled by a coefficient λ. CORAL loss is simply the distance between the latent

feature covariance matrices of the source and target embeddings produced by a model prior to their activation functions. This loss is described in Equation 2.1, where $\|\cdot\|_F^2$ is the squared Frobenius norm, while the equation describing both C_S and C_T (the source and target covariances) is given in Equation 2.2. In this equation, $\mathbf{1}$ is a vector consisting of all 1s and so $(\mathbf{1}^{\mathrm{T}}\boldsymbol{Z})$ is the sum of $\boldsymbol{Z}$, an embedding matrix of $n_{spots} \times d_{embedding}$, over its rows [31, 72].

$$L_{CORAL} = \frac{1}{4d_{embedding}^2}\|C_S - C_S\|_F^2 \tag{2.1}$$

$$C = \frac{1}{n-1}\left(\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{Z} - \frac{1}{n}(\mathbf{1}^{\mathrm{T}}\boldsymbol{Z})^{\mathrm{T}}(\mathbf{1}^{\mathrm{T}}\boldsymbol{Z})\right) \tag{2.2}$$

By contrast, other methods such as DANN and ADDA use an adversarial approach to ensure embeddings produced in both domains are indistinguishable. Adversarial networks were first proposed by Goodfellow *et al.* and used in generative adversarial networks (GANs) by training a generator network G to generate new "fake" data, using random Gaussian noise as inputs, while a discriminator network D learns to tell apart the "fake" data from "real" data by attempting to classify them. After training, G can be used to generate "fake" data similar to the "real" data. This is done by setting the optimization criteria for the composite model $D \circ G$ such that the weights of D are optimized to classify "fake" and "real" data well, while G is optimized to fool the discriminator resulting in poor performance on the same task [73]. We specify how this adversarial training is achieved further in this section while comparing DANN and ADDA. Regardless, adversarial training is not a simple optimization problem. As the weights of each of G and D each are changing in training, and as both networks have different goals, the objectives for each continuously changes. Furthermore, it is not the goal for any one subnetwork to converge, as that would mean one subnetwork has "won" and the other subnetwork has "lost"; in both cases, the generator fails in its role in producing good "fake" data. Ideally, both networks should demonstrate

mediocre but balanced performance in the classification metric on both "fake" and "real" data, and the overall system should achieve a metastable state during training between all these components and losses.

In both DANN and ADDA, instead of generating "fake" data based on a noise input, the generator, which we refer to as an "encoder" E, learns to generate embeddings from input data from both the source and target domains, while the discriminator D learns to classify the embeddings derived from each of the source and target domain, $\boldsymbol{Z}_S$ and $\boldsymbol{Z}_T$ respectively. This task is auxiliary to the primary prediction task, where a predictor P is used and $P \circ E$ is trained end-to-end using traditional supervised learning on source input data and labels via backpropagation [29, 30].

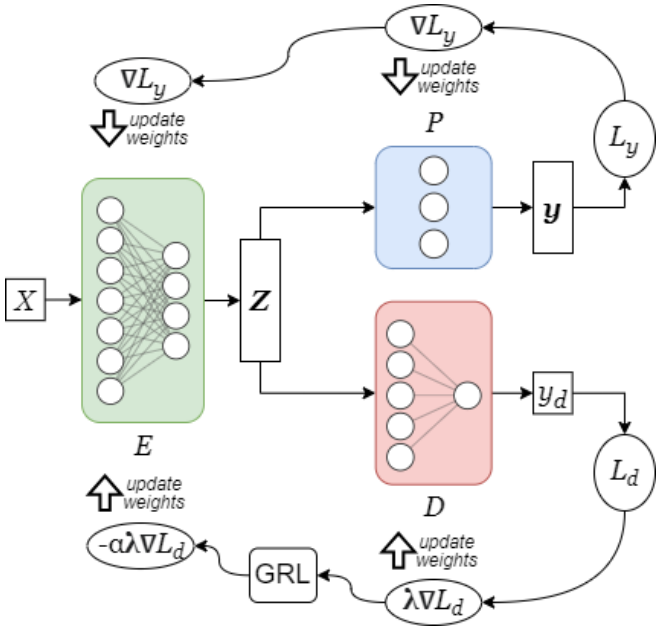

Fig. 2.2: Diagram of the DANN architecture showing the adversarial aspect introduced by the GRL. Forward propagation is shown with orthogonal arrows, while backpropagation is shown with curved arrows. α is a scaling parameter for the encoder gradient relative to the discriminator gradient.

DANN achieves the adversarial aspect through a gradient reversal layer (GRL) between the encoder and discriminator, as shown in Fig. 2.2. This layer is not active during a forward pass, but during backpropagation, the GRL negates the gradient of the loss function passed from

discriminator to the encoder. This causes the encoder to maximize the loss of the discriminator, while at the same time the discriminator is optimized for minimum loss in the domain classification task. The entire model is trained at the same time by combining both losses using Equation 2.3. In this equation, $\boldsymbol{X}_{S_i}$ and $\boldsymbol{Y}_{S_i}$ are input and label vectors of sample i out of n_S samples from the source domain, $\boldsymbol{X}_{T_i}$ is the input vector of sample i out of n_T samples in the target domain, y_{d_S} and y_{d_T} indicate membership in either the source or target domain, respectively, L_y is the loss function for the prediction task, L_d is a classification loss (typically binary cross-entropy (BCE); see Equation 3.10 in Section 3.4), and λ is a weight factor for the discrimination task.

$$L(\boldsymbol{X}_S, \boldsymbol{X}_T, \boldsymbol{Y}_S) = \frac{1}{n_S}\sum_{i=1}^{n_S} L_y(P \circ G(\boldsymbol{X}_{S_i}), \boldsymbol{Y}_{S_i}) + \lambda\left(\frac{1}{n_S}\sum_{i=1}^{n_S} L_d(D \circ G(\boldsymbol{X}_{S_i}), \mathrm{y}_{d_S}) + \frac{1}{n_T}\sum_{i=1}^{n_T} L_d(D \circ G(\boldsymbol{X}_{T_i}), \mathrm{y}_{d_T})\right) \quad (2.3)$$

As opposed to the GRL-based approach of DANN, ADDA uses an approach more directly modeled after GANs [73] using 3 stages of training [29]:

1) First, a pretraining phase is performed using only source domain data, training the predictor and encoder together as a single traditional supervised learning task, with predicted labels $\widehat{\boldsymbol{Y}}_S = P \circ E_S(\boldsymbol{X}_S)$ (more on encoder E_S's subscript following) and ground truth labels $\boldsymbol{Y}_S$. This is the only point where training on the prediction task is done.
2) Then, in the adversarial phase, the weights of the encoder E_S and prediction head P are frozen, while two new modules, E_T and D, are initialized. E_T, the target encoder, is initialized with, but does not share, the weights of frozen source encoder E_S, while D is the discriminator. In this phase, E_T is equivalent to the generator in a GAN, producing "fake"

embeddings $\boldsymbol{Z}_T = E_T(\boldsymbol{X}_T)$ from target input data that mimic "real" embeddings $\boldsymbol{Z_S} = E_S(\boldsymbol{X}_S)$ produced from source input data, while discriminator D classifies the embeddings as originating from either the target or source domain. For domain classification criterion L_d, ADDA follows GANs in alternating training of the target encoder and discriminator. When the discriminator is trained, the weights of both source and target encoders E_S and E_T are frozen. The discriminator learns to classify the embeddings derived from both source and target data, predicting domain labels $\widehat{\boldsymbol{y}}_d = D \circ E(\boldsymbol{X})$, where E is the encoder matching the domain of input data $\boldsymbol{X}$ and $\widehat{\boldsymbol{y}}_d$ are the predicted domains for the input data. During iterations where E_T is trained, all other modules' weights are frozen, and the domain discrimination task proceeds with only the target data, producing labels $\widehat{\boldsymbol{y}}_{d_T} = D \circ E_T(\boldsymbol{X}_S)$. To optimize the weights of E_T such that D misclassifies its embeddings, the binary ground truth labels are flipped by the equation $\boldsymbol{y}'_d = \mathbf{1} - \boldsymbol{y}_d$, where $\boldsymbol{y}'_d$ are the now incorrect labels.

3) Finally, during inference, the target encoder E_T feeds embeddings to predictor P such that it predicts target domain labels $\widehat{\boldsymbol{Y}}_T = P \circ E_T(\boldsymbol{X}_T)$.

2.3.2.1 CellDART

As far as we are aware, at the time of writing, the only method that explicitly leverages UDA techniques while using scRNA-seq as reference data is CellDART, integrating datasets of the two different modalities collected by different research groups. While the authors claim to use ADDA, upon closer inspection of the paper as well as the provided source code, their method architecturally more closely resembles DANN as shown in Fig. 2.2, using a single encoder for both source and target [30]. Training uses ADDA's strategy of alternating training the encoder and discriminator [29], where in one step only the discriminator head is trained, and, in the other the

discriminator weights are frozen while the domain labels are flipped according to $\boldsymbol{y}'_d = \mathbf{1} - \boldsymbol{y}_d$. In the second step, the overall loss is calculated the same as DANN in Equation 2.3 while simultaneously training on the prediction task, but with the weights of the discriminator frozen. Additionally, similar to DSTG [70], they simulated *in situ* capture spots as pseudo-spots to form their source domain dataset [27]. As we reimplemented CellDART as part of our investigation, we explore and discuss its specifics in Section 3.4.1.

Using the expected locations of each of 10 excitatory neuron cell types originating from an scRNA-seq dataset [74] to evaluate performance, CellDART showed favourable results when compared to 6 other methods, 2 of which were RCTD [32] and DSTG [70], previously discussed. Furthermore, CellDART was not one of the 11 methods evaluated in the Spotless analysis that showed RCTD as a consistent top performer [33]. As CellDART was the only known deconvolution method to leverage UDA in addition to its reported performance, it was important for us to explore, investigate, and compare against, as we detail further in this work.

2.3.2.2 Remark on the current state of benchmarking cell type deconvolution

Due to the nature of the task of UDA and the nascent field of cell-type deconvolution of spatial transcriptomics data, reference datasets with ground truth are difficult to obtain, and evaluation of performance is often done by using some kind of proxy for accuracy or using synthetic data. DSTG and many other models, for example, use pseudo-spots [33, 70, 75]. In the UDA setting, while this tells us valuable information on how well a model is performing in the source domain, it does not tell us anything about how well a model performs in the target domain. This is especially salient as we intend for our methods to be able to be used to integrate disparate datasets gathered at different times under different conditions by different researchers. An approach often used, in addition to evaluating using synthetic data, is to evaluate based on a correlation metric between

the expression of known marker genes for given cell types, and the proportions of the cell types predicted by the model [27, 32, 75, 76]. A final approach sometimes used is to visually validate regional distributions of cell types in plots and figures [27, 75, 77].

While not an exhaustive list, and while most methods rely on multiple different evaluation metrics, this still makes evident an issue pervasive in this field: few methods are actually evaluated based on their performance in their intended role in a direct and robust manner. Marker-based metrics require existing research on the expression of specific genes, which limits their potential in improving beyond those markers and truly leveraging the whole genome, and therefore could only ever be used as a sanity check. Spotless, a benchmarking pipeline, seeks to partially remedy this by creating synthetic "pseudo-Visium" spots in its "gold standard" datasets by leveraging the increased resolution of Stereo-seq fields of view (FOVs) to assign cell types to spots. An FOV for Stereo-seq is like a slide but corresponds to a "zoomed-in" subregion. They then spatially downsampled the high-resolution Stereo-seq spots to 10x Genomics Visium-sized spots to obtain relatively accurate cell type proportions, which can be used as a ground truth. A downside is that at Visium-like resolutions, each of the 7 FOVs only has $3 \times 3 = 9$ spots [33].

3. Methodology

3.1 Overview

In this chapter, we explain the protocols we used to conduct our investigation. In Section 3.2, we go over each dataset used, processing and cleaning of data, the segregation of each dataset into training, validation, and test sets, and the generation of pseudo-spots. In Section 3.3, we detail the evaluation protocols we set to be able to accurately measure models' performance in both domains. Finally, in 3.4, we elaborate on the implementation of training of the models we investigated. The overall workflow, excluding hyperparameter tuning, is shown in Fig. 3.1.

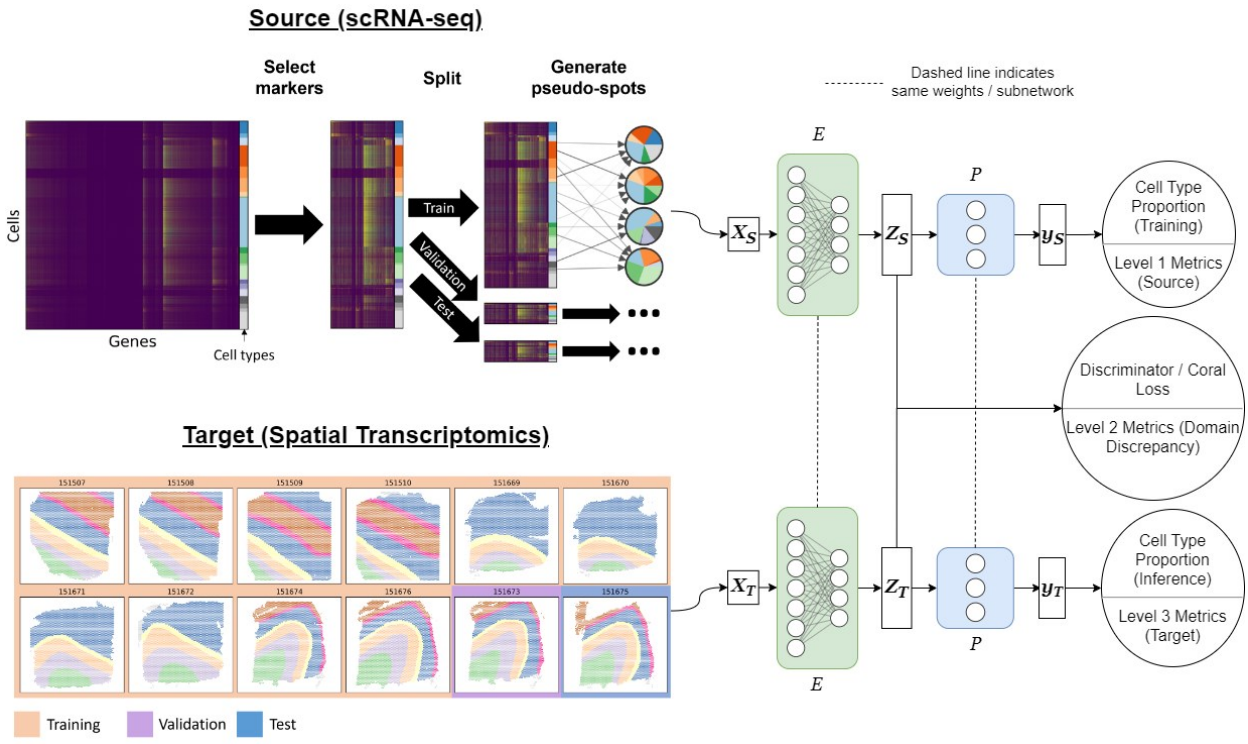

Fig. 3.1: Overall methodology, excluding hyperparameter tuning, showing processing, marker selection, pseudo-spot generation, data splitting, and when each dataset is used when training or under each level of evaluation. The dlPFC datasets for both source and domain are shown for this example, where a single slide each is held out as validation and test. The x-axes (genes) of the GEx heatmaps before and after marker selection are not to scale, as the actual number of selected genes was much lower than the total number. The neural networks shown are a simplification and differ depending on the specific model architecture used. The architectures weights for the top and bottom models are the same between source and target, indicated by the dashed lines; in reality, this is also differs depending on the specific model used.

3.2 Data and data processing

As our investigation focused on unsupervised domain adaptation, each experiment required two datasets: a labelled source dataset, and an unlabelled target dataset. The raw source datasets used were single-cell GEx matrices of $N_{cells} \times N_{genes}$, with values as raw integer transcript counts, with cell type labels for each cell. The raw target datasets used were spot-level GEx from multiple cells of $N_{spots} \times N_{genes}$, also containing raw integer transcript counts. Each dataset also contained varying amounts of metadata for each cell/spot, which guided data processing and evaluation. A basic overview of data processing, excluding log-transforming, normalization, and preprocessing, is shown for both spatial and single-cell data in Fig. 3.1.

3.2.1 Datasets, cleaning, and pre-processing

In general, for each dataset, we first imported and converted raw data from each dataset into the AnnData format using the *anndata* (v. 0.8.0) [78] and *scanpy* (v. 1.8.2) [65] Python packages. Using *scanpy*, we performed quality control by first filtering out genes found in fewer than 3 samples. Then, for single-cell source data, we filtered out cells with fewer than 200 genes. Where possible, we also filtered out cells where 5% or more of the total counts were mitochondrial. As our goal in UDA was to make predictions in the target domain regardless of the sample quality, we did not filter out spatial transcriptomics spots nor genes, instead relying on single-cell quality control, which would later be reflected in the target data after marker selection (see Section 3.2.2). Finally, we performed library size normalization, whereby we scaled the counts such that, for each sample, the total transcript counts across all genes was 10,000. Further details on pre-processing specific to each dataset, along with information on each dataset itself, are explored in the following subsections.

3.2.1.1 Dorsolateral prefrontal cortex

Human dorsolateral prefrontal cortex (dlPFC) datasets were used by the CellDART authors to evaluate and compare the performance of their model against other methods [27]. As such, we used the same datasets and evaluation methods to reproduce and compare their results.

The source dataset came from a transcriptomic study of the dlPFC in major depressive disorder (MDD) [74]; we, as with the CellDART authors [27], only used samples that were obtained from the healthy controls. After filtering out the non-control samples, preprocessing, and filtering out 2,130 genes, the dataset composed of snRNA-seq GEx profiles for 35,212 nuclei in the form of transcript counts for 27,932 genes. Each sample had been additionally labelled into one of 26 cell type clusters, with 10 being excitatory neurons, each labelled based on genetic markers, with one or more of the 6 mammalian neocortex layers [79] they could be expected to be found in.

The target dataset composed of a total of 12 slides of dlPFC tissue [80], which we obtained via the spatialLIBD package [81]. Each slide's data contained between 3,460 to 4,789 (inclusive) 10x Genomics Visium spots (mean: 3,973), each with corresponding GEx transcript counts for 25,615 genes along with 2-dimensional (2D) coordinates on the plane of the slide. Each spot had also been annotated by its location in one of the 6 mammalian grey matter neocortex layers [79], with some spots labelled as white matter if they were not a part of the neocortex. The 12 slides along with their neocortex layer is shown in the bottom left of Fig. 3.1.

3.2.1.2 Pancreatic ductal adenocarcinoma

We elected to use a PDAC dataset by Moncada *et al.* [26], previously used in several spatial transcriptomics deconvolution analyses [75, 77, 82]. This consisted of, in part, Ståhl method-derived data from 2 slides, each from a separate PDAC tumour, referred to as PDAC A and PDAC B, with 428 and 224 spots respectively. Each spot contained corresponding transcript count data

for 19,725 genes along with 2D coordinates on the plane of the slide and were estimated by the authors to contain ~30-70 cells each. In their analysis, Zhou *et al.* [82] used annotations of spots by region (cancer region, pancreatic tissue, duct epithelium, stroma, and interstitium) from the data-generating paper [26]. These regions had been annotated by the original authors by leveraging sample-matched scRNA-seq data and subsequently validated using independent annotations of images of the corresponding slides. We were unable to find these annotations in the dataset except in the figures of the dataset's accompanying paper, so we manually transferred and verified these annotations from said figures.

In other analyses, the accompanying sample-matched scRNA-seq data provided by Moncada *et al.* [26] was usually also the source dataset used to evaluate spatial transcriptomics deconvolution methods [75, 77, 82]. As we intended to investigate UDA where sample-matched data are not available, for our source dataset, we instead used an independent PDAC scRNA-seq dataset originally produced by Peng *et al.* [83] and then processed by Chijimatsu *et al.* [84]. As the data were already preprocessed, no samples or genes were filtered out when we performed quality control and the full complement of 41,964 cells and 16,999 genes were available, along with 10 different cell types.

3.2.1.3 Mouse cortex and Spotless gold standards

An issue inherent in cell type deconvolution is the lack of reliable ground truth for cell type proportions to evaluate our methods against. For all previously discussed target datasets, no cell type proportions are provided; instead, all evaluation had to be done using metadata (layers for dlPFC, regions for PDAC). To remedy this issue among a host of evaluation related issues, Sang-aram *et al.* presented Spotless, a spatial transcriptomics cell type deconvolution pipeline, alongside 54 "silver standard" and 3 "gold standard" datasets [33]. In particular, the "gold standard" datasets

consists of high-resolution seqFISH [22] (gold standards 1-2) or STARmap [52] (gold standard 3) data. Individual cells were isolated *in silico* and their cell types identified. Then, GEx and cell types were aggregated in 55 μm "spots" to simulate 10x Genomics Visium-like GEx data, along with cell type proportion labels. We used "gold standard 1", which was derived from a single mouse cerebral cortex slide and consisted of 7 individual fields of view (FOVs), which are sub-regions of a slide that are captured in a single spatial transcriptomic array. A disadvantage of this method was that, because of the aggregation, each FOV consisted of merely 9 "spots" arranged 3×3; otherwise, GEx data were available for 10,000 genes and cell type proportions for 17 cell types.

While Spotless [33] used sample-matched data as a source; i.e., the source dataset used was holdout samples, we again opted to use an independent scRNA-seq GEx dataset to investigate the case where matched single-cell data are not available. We used data collected from the mouse primary visual cortex (VISp) and anterior lateral motor Cortex (ALM). After pre-processing, the GEx transcript counts data for 4,915 genes were filtered out, leaving 40,853 genes across 25,720 cells, of which 22,277 had cell type information [63].

3.2.2 Data splitting and marker gene selection

To be able to evaluate, perform validation, and compare performance in both the source and target domain, we randomly split the single-cell data for each source dataset into training, validation, and test sets with proportions of 80%, 10% and 10% respectively. We stratified using cell type to maintain consistent proportions across splits, and thus filtered out cell types with fewer than 10 cells to ensure that at least 1 sample would be present in each of the splits. This resulted in a reduction of 29 to 28 cell types for mouse cortex. To avoid data leakage, this splitting was performed after initial preprocessing but prior to any marker selection, pseudo-spot generation, or

normalization, among other analyses. Additionally, the same splits were used for all said downstream analyses to maintain consistency and remove sources of variation.

For the spatial data, as we did not want to disrupt the spatial structure of each slide (or FOV for the Spotless mouse cortex data), we elected to hold out 2 whole slides/FOVs in each target dataset as our validation and test datasets. We randomly selected the validation and test slide, while eliminating certain samples from contention due to poor suitability for evaluation (4 sections in the dlPFC set did not have any spots in layer 1 of the neocortex, while 4 FOVs in Spotless's "gold standard 1" contained relatively few of the total cell types in the dataset). As our PDAC dataset only contained 2 sections, each with slightly different but not exclusive annotations, we opted not to hold out a test sample, instead relying on the fact that our models were trained in an unsupervised manner to limit data leakage.

As we wanted to be able to directly compare our methods against CellDART, we followed the example of the CellDART authors [27] and performed feature selection by identifying characteristic genes for each cell type in the source dataset. This was done on our part to reduce the dimensionality of the data and thus avoid the "curse of dimensionality" while training our models, as the datasets we used contained GEx data for ~20,000 genes, often in excess of the number of samples. For detailed information on the characteristics of each dataset, refer to Section 3.2.1. Using *scanpy*'s `rank_genes_groups` function [65], we first ranked the top characteristic genes for each cell type in the training split of the source data. The process performed by this function with the parameters we used is as follows:

1) For each gene, the Mann–Whitney U (MWU) [85] test was used to find the significance of the enrichment of that gene's expression, or *p*-value, for each cell type, in a one-vs-rest (comparing GEx for a given gene within and without a cell type) manner. The MWU test is

a non-parametric significance test of the null hypothesis that two sets of sample values are drawn from equal distributions. This provides, for every cell type and every gene, a *p*-value indicating that gene's enrichment for a cell type.

2) To correct for multiple discoveries for a given cell type, the Benjamini-Hochberg method [86] was used to calculate the false discovery rate (FDR) for each gene from *p*-values.
3) The FDRs for each cell type were ranked by significance (lowest to highest, with lowest being most significant.)

Then, the union of the top n_{markers} across all cell types was used as the selected features of each source dataset, with n_{markers} being a hyperparameter. As CellDART [27] used $N_{\text{markers}} = 20$ for brain data, we opted to investigate performance for $N_{\text{markers}} = 20, 40,$ and 80. We then used the intersection of the selected features with the genes in the matching target dataset as the final set of genes.

3.2.3 Pseudo-spot generation

To train our models to predict cell type proportions, we, like other methods, chose to aggregate GEx and cell type proportions into pseudo-spots. Since we sought to compare against CellDART, we, as with feature selection, once again used CellDART's method [27] as a starting point. This involved two additional hyper-parameters, n_{mix} and n_{spots}. The method used is as follows for a given GEx matrix $\boldsymbol{X}$ of shape $N_{cells} \times N_{genes}$, cell type vector $\boldsymbol{y}$ of length N_{cells}, and hyperparameters n_{mix} and n_{spots}:

1) Convert $\boldsymbol{y}$ into a one-hot matrix $\boldsymbol{Y}$ of $N_{cells} \times N_{\text{cell types}}$, where $N_{\text{cell types}} = |\{y_i\}_{i \in \{1,\dots N_{cells}\}}|$ (# of unique values of y_i).
2) For $i = 1 \dots n_{\text{spots}}$,

a. draw, with replacement, n_{mix} samples from $\boldsymbol{X}$ and their corresponding samples in $\boldsymbol{Y}$, forming samples $(\boldsymbol{X_k}, \boldsymbol{Y_k})_{\text{k}\in\{1,\dots n_{\text{mix}}\}}$,

b. randomly sample n_{mix} values from distribution $U_{[0,1]}$, normalize the values such that their sum is 1, and multiply each drawn sample $(\boldsymbol{X_k}, \boldsymbol{Y_k})$ exclusively with one of the sampled values,

c. and sum the $\text{k} \in \{1, \dots n_{\text{mix}}\}$ samples together elementwise along n_{mix} to form i^{th} elements $\boldsymbol{X_{S_i}}$ and $\boldsymbol{Y_{S_i}}$ of pseudo-spot matrices $\boldsymbol{X_S}$ and $\boldsymbol{Y_S}$. As each sampled cell $\boldsymbol{Y_k}$'s elements sum to 1 due to its one-hot encoding, and the uniformly sampled fractions are also normalized to sum to one, naturally the pseudo-cell type proportion of $\boldsymbol{Y}_{S_i}$ will also sum to 1. This also preserves the library size normalization on the GEx counts performed in the pre-processing stage.

This method forms pseudo-spot matrices $\boldsymbol{X_S}$ and $\boldsymbol{Y_S}$ of shapes $n_{\text{spots}} \times N_{genes}$ and $n_{\text{spots}} \times N_{cell\ types}$, respectively, which could now be considered as our source dataset. For our part, we performed this process separately for each of the training, validation, and testing splits. We also chose to fix $n_{\text{spots}} = 100{,}000$, or 5 times as many as used in CellDART, as any arbitrary number of pseudo-spots could be generated. Increasing n_{spots} acts as a form of data augmentation, where few data are randomly transformed to increase the effective size of the dataset, acting as a form of regularization to prevent overfitting [87]. Furthermore, as we held out slides/FOVs for validation and evaluation, as opposed to CellDART which treated each section as individual datasets with no holdout samples, we combined all training slides. This resulted in larger training sets (40,450 samples for dlPFC vs. a mean of 3,973 spots per slide for CellDART) requiring a higher number of pseudo-spots. We thus found that 100,000 was good a value for n_{spots},

augmenting the source dataset by more than two-fold for dlPFC and more for PDAC and mouse cortex. For validation and test sets, we generated 25,000 pseudo-spots each.

Pseudo-spot generation is shown as part of the scRNA-seq processing pipeline in Fig. 3.1. We hereafter denote our labelled source pseudo-spot datasets and unlabeled target datasets with the subscripts "T" and "S", respectively.

3.2.4 Log transforms and normalization

While the process of log-transforming raw count data is controversial from a statistical point of view and especially for analysis of variance (ANOVA) [88], it is still common practice for transforming from long-tailed data, such as transcript counts, into a better fit for linear models, such as the inputs of neural networks [89]. Furthermore, we aimed to minimize deviation from CellDART, which used the log-transform, to be able to fairly and directly compare their domain adaptation method [27]. We transformed all input matrices elementwise as shown in Equation 3.1, where ln is the natural logarithm.

$$X'_{ij} = \ln\left(1 + X_{ij}\right) \tag{3.1}$$

After log transformation, we used *sci-kit learn*'s (v. 1.1.2) `StandardScaler`, which performs standardization [67], to normalize our input data by feature across samples. The equation for each the standardization of each element is as shown in Equation 3.2, where i and j index samples and features respectively, X_{ij} is a single element of an input matrix, $\overline{\boldsymbol{X_j}}$ is the mean of the elements for the vector of samples corresponding to the feature, $s(\boldsymbol{X_j})$ is the sample standard deviation of the elements of that feature vector, and X'_{ij} is the standardized value for the element. This linearly transforms the data such that the mean is 0 and standard deviation is 1.

$$X'_{ij} = \frac{X_{ij} - \overline{\boldsymbol{X_j}}}{s\left(\boldsymbol{X_j}\right)} \tag{3.2}$$

As CellDART uses min-max scaling, we additionally used *sci-kit learn*'s `MinMaxScaler`, which linearly scales the data for each feature such that the minimum value is 0, and maximum value is 1 [67], to ensure that we could evaluate CellDART as intended by its authors. The transformation is expressed in Equation 3.3, where i and j index samples and features respectively, X_{ij} is an element of a matrix, $X_{j_{min}}$ is the minimum value of the elements of a vector of samples for feature j, $X_{j_{max}}$ is the maximum value, and X'_{ij} is the element's normalized value.

$$X'_{ij} = \frac{X_{ij} - X_{j_{min}}}{X_{j_{max}} - X_{j_{min}}} \tag{3.3}$$

For both scaling methods, the statistics ($\overline{\boldsymbol{X_j}}$, $s(\boldsymbol{X_j})$, $X_{j_{min}}$, and $X_{j_{max}}$) were calculated only using training samples, while the transform was applied to all sets (training, validation, and test) to avoid data leakage. For spatial data, where there were multiple histological sections or FOVs, we concatenated the sections/FOVs within a given split prior to transformation.

3.3 Evaluation

To mitigate the pitfalls in evaluating cell-type deconvolution we remarked on in 2.3.2.2, we used multiple approaches to cover as many blind spots that each metric would have on its own and multiple datasets to address the others' shortcomings. In doing so, we aimed to show that our methods could work well in many contexts in both the source and target domains. We organized our metrics into 3 levels: (a) synthetic data or source domain/pseudo-spot performance, (b) invariance of the models between target and source domains, and (c) performance using real spots. Our choice of datasets (dlPFC, PDAC, and mouse cortex; see Section 3.2.1) was also guided by their suitability in these three levels of evaluation. The model configurations and contexts in which each level of metric is used is shown in Fig. 3.1.

3.3.1 Source domain performance

The first level was straightforward: we simply evaluated the cell type proportion labels of our generated pseudo-spots. We used the mean sample-wise cosine distance as our distance metric, as shown in Equation 3.4, where $\boldsymbol{Y_{true_i}}$ and $\boldsymbol{Y_{pred_i}}$ are the matching i^{th} samples of the ground truth and predicted cell type proportion matrices, respectively, while $\|\boldsymbol{Y_{true_i}}\|$ and $\|\boldsymbol{Y_{pred_i}}\|$ are the magnitudes (Euclidean lengths, or L2 norms) of those vectors. The cosine distance is based on the angle between the two vectors, with 0 being 0°, 2 being 180°, and 1 being orthogonal. Each element of $\boldsymbol{Y_{true_i}}$ and $\boldsymbol{Y_{pred_i}}$, being proportions, has a range of $[0,1]$, and so there cannot be a negative component to a sample's output vector. This means that the angle between the vectors is at most orthogonal, and it follows that the range of the distance metric is restricted to $[0,1]$, with 0 being exactly equal and 1 being most different. We used this evaluation on each of the training, test, and validation pseudo-spot splits; for more details on how pseudo-spots were generated, refer to Section 3.2.2. This first level indicates a given model's performance in the source domain.

$$D_C(\boldsymbol{Y_{pred}}, \boldsymbol{Y_{true}}) = \frac{\sum_{i=1}^{n_{spots}} \left(1 - \frac{\boldsymbol{Y_{true_i}} \cdot \boldsymbol{Y_{pred_i}}}{\|\boldsymbol{Y_{true_i}}\| \cdot \|\boldsymbol{Y_{pred_i}}\|}\right)}{n_{spots}} \tag{3.4}$$

3.3.2 Variance between source and target

For the second level, we compared the latent representations generated for each of the target and source datasets to evaluate the variance when moving from the source to target domain. We used the intermediate embeddings produced by a model's encoder just prior to feeding forward into a prediction head. For a given encoder E, predictor P, and input matrix $\boldsymbol{X}$, the embeddings matrix can be represented by $\boldsymbol{Z} = E(\boldsymbol{X})$, and thus the final predictions $\widehat{\boldsymbol{Y}} = C(\boldsymbol{Z})$ can be obtained. At this second level of evaluation, we sought to measure how well integrated were the sets of representations $\boldsymbol{Z_S} = E(\boldsymbol{X_S})$ and $\boldsymbol{Z_T} = E(\boldsymbol{X_T})$, the source and target representation sets

respectively, using 2 metrics: (a) a use case of LISI called iLISI, as defined and used in Harmony [69], and (b) RF50, both of which we have defined below.

For a given sample i and its corresponding vector $\boldsymbol{Z_i}$ in a space $\mathcal{Z}$, LISI's purpose is to quantify the effective number of classes of other samples $\boldsymbol{Z_{k\neq i}}$ that dominate its local neighbourhood [69]. Simpson's Index λ is a diversity metric that gives the probability that two random samples drawn from a set will be of the same class, given by Equation 3.5, where R is the number of discrete classes in the set and p_r is a given class's proportion within that set [90]. If the problem is reframed under the geometric distribution, where p is the probability of an event occurring on a given try, and $\frac{1}{p}$ is the mean number of tries for at least one success, it then follows that the inverse Simpson's Index is the mean number of independent draws of two samples required to obtain samples of the same class at least once. When the proportions of all classes are equal, $p_r = \frac{1}{R}$ and so $\lambda = \frac{1}{R}$ as well, while when only 1 class is present when there should be multiple, $\lambda = 1$. Equivalently, a perfectly balanced set will have $1/\lambda = 1$, whereas when one class dominates, $1/\lambda \to 1$. For these reasons, the inverse Simpson's Index is often used as a measure of the effective number of classes within a set, although it is in reality a measure of the number of *dominant* classes [91], which we discuss later in this section.

$$\lambda = \sum_{r=1}^{R} p_r^2 \tag{3.5}$$

To define a neighbourhood, LISI takes after t-distributed stochastic neighbour embedding (t-SNE) in using a Gaussian kernel centered around a query point $\boldsymbol{Z_i}$ [92]. For a query sample i, the contribution $p_{k|i}$ of a sample $k \neq i$ toward its class proportion p_r is weighted by the Gaussian probability density function (pdf) $f(x) = \frac{1}{\sigma_i}\varphi\left(\frac{x}{\sigma_i}\right)$, where $\varphi(z)$ is standard normal distribution of $\mu = 0$ and $\sigma^2 = 1$, using its distance $\|\boldsymbol{Z_k} - \boldsymbol{Z_i}\|$ from the query point $\boldsymbol{Z_i}$ as input. To account for

how dense or sparse the neighbourhood surrounding a sample i is, instead of fixing the variance σ_i^2, a fixed value of perplexity is chosen instead.

To help explain the intuition behind fixing perplexity, we explain it in the context of diversity scores [93]. If we define a discrete random variable X with N possible outcomes and their corresponding proportions, in this case $p_{k|i} = f(x_{k|i})\ \forall\ k \neq i$ where $x_{k|i} = \|\boldsymbol{Z_k} - \boldsymbol{Z_i}\|$, then the perplexity $K \leq N$ is the reciprocal of the geometric mean of all proportions, each weighted by itself. For our case, the perplexity K_i for the kernel around a query point i is calculated using Equation 3.6. While the Simpson's Index λ can be thought of as the self-weighted *arithmetic* mean of proportions [90], $\frac{1}{K}$ is the self-weighted *geometric* mean. These are both proportions if all proportions had to be equal, just that the arithmetic uses the *sum* of proportions weighted by themselves, while the geometric mean is measured and constrained by the *product* of proportions weighted by themselves. This means that, while Simpson's index λ can be useful for an observer as an *expectation* of the class proportion for a given sample, a *geometric* mean gives a better estimate of the average proportion *relative* to the population as proportions are a *relative* value. Their inverses, $\frac{1}{\lambda}$ and K, are special cases of the Hill diversity numbers N_q with orders 2 and 1, respectively [90]. N_2 is also called Simpson's diversity and used as a "dominance metric", such as when used to quantify mixing with iLISI. On the other hand, N_1, related to Shannon entropy [94], is used as a "true" diversity metric [91]. We can then say that the perplexity K_i for a point i gives a true unbiased estimate for the effective number of samples in the neighbourhood of $\boldsymbol{Z_i}$. By fixing perplexity instead of standard deviation, each kernel can be a different width in response to how dense or sparse the neighbourhood surrounding a sample is. To calculate iLISI, we adapted the implementation in the Python package *harmonypy* [95], itself a Python port of Harmony [69]. We concatenated sample-wise the embedding matrices $\boldsymbol{Z_S}$ and $\boldsymbol{Z_T}$ using target and source domain as

classes, while we also randomly undersampled the majority class to eliminate any imbalance. For the final score, we used the median value across all samples, referred to as median-iLISI (miLISI).

$$K_i = \frac{1}{\prod_{k=1}^{n_{samples}-1} p_{k|i}^{p_{k|i}}}, k \neq i \tag{3.6}$$

Our second metric, RF50, measured how well a classifier can discriminate between the two classes. We concatenated sample-wise the embedding matrices $\boldsymbol{Z_S}$ and $\boldsymbol{Z_T}$ using target and source domain as classes, then split the combined set, stratified by domain, into training and test sets of 80% and 20% of the total set. We found the first 50 (or the width of the embedding, whichever lesser) principal components using the RF50 training set and applied the transform to both sets. Then, we used the RF50 training set to train a random forest classifier using the default settings of the *imbalanced-learn* (v. 0.10.1) Python package's `BalancedRandomForestClassifier`, which randomly undersamples the majority class for every bootstrap model [96]. Finally, we calculated the average accuracy across all classes ("macro" accuracy) using the RF50 test set; the closer the score to 1, the worse the domain invariance of the encoder was, while the closer to 0.5 (i.e., random), the better.

Finally, as we were evaluating how well an encoder can provide domain-invariant representations to the predictor, this required a consistent definition of a prediction head across all models for comparable results. Due to backpropagation, the training of an encoder is not independent of the predictor and could vary otherwise. For every model and dataset, a predictor head C was a single fully connected layer with $N_{cell\ types}$ neurons, followed by a softmax activation function [71], which produced an output vector $\boldsymbol{Y_i}$ with values in the range $(0,1)$ and also summing to 1, consistent with cell type proportions. By keeping the predictors as shallow as possible, a predictor is only capable of learning proportions that are monotonic transforms of a latent vector $\boldsymbol{Z_i}$, because single-layer perceptrons can only learn first-order mappings [97]. This

means that an encoder E must map all samples from a manifold in GEx space to a representation $\boldsymbol{Z_i}$ in a latent space $\mathcal{Z}$, where meaningful variations in cell type proportions only occur linearly in that space [71]. We did this to reduce potential variability in the latent distributions from the encoder, allowing us to evaluate the representations specifically as they pertain to the classification task. For further details on the architecture of the models, see Section 3.4.

To ensure that any good result in the evaluation of variance between domains was meaningful, we specifically sought single-cell source domain data obtained by different research groups from the target spatial transcriptomics data (see 3.2.1 for further details on each dataset). If a given method performed well both on pseudo-spots in the source domain (level 1 performance) and demonstrates invariance across source and domain data (level 2 performance), then we could by proxy conjecture that our model's performance in the source domain would likely generalize to the target domain.

3.3.3 Target domain performance

For performance in the target domain, we faced the same issues as other methods [27, 32, 33, 70, 75, 77], where most real datasets did not have direct ground truth data. We therefore used a different approach for each of the three datasets.

Each spot in the dlPFC 10x Genomics Visium target dataset [80] had been annotated with one of the 6 mammalian neocortex layers [79]. Furthermore, of the 26 cell types present in the source dataset, there were 10 distinct excitatory neuron groups, each having been labelled with whichever neocortex layer(s) where it would be expected to be found. CellDART's authors [27] used the predicted cell type proportion of each of these 10 excitatory neuron groups to separately calculate receiver operating characteristic (ROC), using whether that excitatory cell type is expected to be found in a spot/layer as the ground truth. The ROC shows the increase in the false positive rate

(FPR) as the true positive rate (TPR) (also referred to as sensitivity or recall) increases when different detection thresholds are used for a set of continuous predictions; the equations for TPR and FPR are shown in Equations 3.7 and 3.8, respectively. In our case, given a set of predictions of proportions, ROC can be thought of as the trade-off between correctly predicting a cell type's abundance in the layer(s) it would be expected to be versus incorrectly predicting that cell type's abundance elsewhere. The area under the ROC (AUROC) can thus serve as a heuristic, at least for the cell types evaluated, how well the predicted proportions match the 2D structure of the slide.

$$TPR = \frac{TP}{TP + FN} \tag{3.7}$$

$$FPR = \frac{FP}{FP + TN} \tag{3.8}$$

Although this metric applied to the dlPFC set relies on markers and only works with a subset of cell types, this target spatial transcriptomics dataset is the largest and most complete out of our datasets, with the most spots per slide and most slides in total (see Section 3.2.1.1). It also uses the 10x Genomics Visium technology, which is the technology with the resolution (1-10 cells per spot) where cell type deconvolution is most useful and the number of spots per slide (up to ~5,000) [23] where datasets are large enough for deep learning methods, and it therefore represents the main use case for our work. We therefore included this dataset to be able to have a metric of how well it performs in this use case, and so, for evaluating true performance across different models, we relied on stronger metrics applied to our other datasets.

As with for dlPFC, we also only had region annotations for our spatial PDAC data, but due to the regions representing wholly different tissues, we were able to map most cell types in the source domain dataset to regions in the target domain dataset using information from their respective

papers [26, 83] as well as the literature [16-18, 98, 99]. The mappings used are shown in Table 3.1, where cell types were mapped to up to 2 regions.

Table 3.1: Mappings from source scRNA-seq data to target spatial transcriptomics regions used to evaluate cell type proportions for spatial PDAC data.

Source domain cell type	**Target domain region**	
	Region 1	Region 2
Ductal cell type 2	Cancer region	
T cell		Stroma
Macrophage cell		
Fibroblast cell		
B cell		
Stellate cell	Pancreatic tissue	
Acinar cell		
Endocrine cell		
Ductal cell type 1	Duct epithelium	
Endothelial cell	Interstitium	

For the Spotless mouse cortex data [33], as ground truth labels were available, we were able to use cosine distance as with the level 1 evaluation on pseudo-spots (see Section 3.3.1). A challenge, however, was mapping the scRNA-seq source dataset's cell types [63] produced by a model to the cell type labels present in the Spotless dataset. Through careful inspection of clusterings of cell types in the source data and through thorough reading of the matching paper's discussion, we were able to map 25 out of 28 cell types from the scRNA-seq data to 13 out of 17 Spotless mouse cortex cell types. This, however, meant the merging of each into 10 cell types. The full mapping used is shown in Table 3.2.

Table 3.2: Mappings of scRNA-seq source dataset cell types to target spatial transcriptomic cell types for the mouse cortex dataset. Where multiple classes were merged, individual cell types are separated by a comma. For the source

cell type, abbreviations used by the paper authors are used; for definitions, refer to the original paper [83]. Greyed out cells indicate no mapping to the other set.

Source cell types	**Target cell types**
Astro, Doublet Astro Aqp4 Ex	Astrocytes deep, Astrocytes superficial
Batch Grouping, L5 PT, L5 IT, L6 CT, L6 IT, L6b, NP, High Intronic, Doublet VISp L5 NP and L6 CT	Excitatory layer 5/6
L2/3 IT	Excitatory layer II, Excitatory layer 3
L4	Excitatory layer 4
Endo, Peri, Doublet Endo	Endothelial, Choroid plexus
Macrophage	Microglia
Lamp5, Meis2, Pvalb, Serpinf1, Sncg, Sst, Vip	Interneurons, Interneurons deep,
Oligo	Oligodendrocytes, Oligodendrocyte progenitor cell
CR, SMC, VLMC	
	Ependymal, Neural Stem Cells, Neural progenitors, Neuroblasts

We strove to limit the amount of manual processing of cell types prior to evaluation to limit inductive bias on our part, only performing merging where necessary and only applying them at evaluation time. For source scRNA-seq cell types lacking a mapping to Spotless mouse cortex cell types, we merged these predictions at evaluation time into an “Other” column (meaning an erroneously predicted cell type) and added a corresponding feature to the ground truth matrix set to 0. For the predictions, we added a corresponding feature at the same index as that in the ground truth and set the values to 0, indicating that none of that cell type was predicted. As this method of evaluation directly used ground truth labels of cell type proportions, this was the strongest metric we used for evaluating level 3 performance, being the performance in the target domain on real spatial transcriptomics spots. On the other hand, the Spotless dataset was the smallest and least applicable to any potential real-world use (for details regarding this dataset, refer to Section

3.2.1.3), so evaluation using all 3 datasets was necessary to gain a comprehensive understanding of our and other methods' performance.

3.4 Models and training

We evaluated 5 models, among them 2 baselines and 3 of our own UDA models. For our baselines, we first tested RCTD, a hierarchical Bayesian model that uses a Poisson random variable [32] which was shown in the Spotless analysis to be simultaneously among the best performing, most consistent and stable, and most scalable methods [33]. Fitting Poisson distributions has been shown to work well for transcript-based count data [100, 101], so we used our data prior to pseudo-spot generation, log-transforming, and normalization (for data processing, refer to Section 3.2). We also tested two versions of CellDART, the first being the author's code [27] and the second being our reimplementation in PyTorch (v. 1.13.1) [102], to more effectively integrate it into our training pipeline and evaluate it fairly (see Section 3.4.1 for details).

Our three models are composed of fully connected subnetworks and are based on the ADDA [29], DANN [30], and Deep CORAL [31] UDA model architectures. As briefly mentioned in 3.3.2, we defined consistent model architecture components across all our models (as well as CellDART [27]) to be able to train and evaluate each in a fair and consistent matter. Each model consisted of at least two parts, an encoder E and a predictor P. Our adversarial models (ADDA and DANN, as well as our reimplementation of CellDART) also incorporated a discriminator D. The encoder's role was to map an $n_{spots} \times N_{genes}$ input matrix $\boldsymbol{X}$ to an $n_{spots} \times d_e$ $\boldsymbol{Z}$ matrix, where d_e is the dimension of the latent space $\mathcal{Z}$. The predictor's role was to map $\boldsymbol{Z}$ to $n_{spots} \times N_{\text{cell types}}$ cell type proportion matrix $\widehat{\boldsymbol{Y}}$. To be able to consistently compare the embedding from source and target domain data, $\boldsymbol{Z_S} = E(\boldsymbol{X_S})$ and $\boldsymbol{Z_T} = E(\boldsymbol{X_T})$ respectively, as discussed in Section 3.3.2, we fixed the predictor to be a single fully connected layer of $N_{\text{cell types}}$ neurons of input dimension

$d_{embedding}$ and no hidden layers, with a softmax output activation function to ensure each row (sample) $\widehat{\boldsymbol{Y}}_{\boldsymbol{i}}$ summed to 1 with all values in the range $(0, 1)$; this is also consistent and therefore a fair comparison with CellDART. Finally, as the discriminator's role was to classify source embeddings and target embeddings into their respective domains, for our models, the discriminator took embeddings matrix $\boldsymbol{Z}$ containing embeddings from one or both domains to an n_{spots} vector $\widehat{\boldsymbol{y}}_{domain} = D(\boldsymbol{Z})$, with input dimension $d_{embedding}$ and a single output neuron with a logistic activation function.

For the prediction task, the overall model M can be represented as $\widehat{\boldsymbol{Y}} = P \circ E(\boldsymbol{X})$. The loss function we used was Kullback–Leibler divergence (KL-divergence) [103], a common one-sided metric of the relative entropy of a model probability distribution Q against a reference distribution P, which in our case are, for every sample i, the rows of the cell type matrices $\widehat{\boldsymbol{Y}}_{\boldsymbol{i}}$ and $\boldsymbol{Y}_{\boldsymbol{i}}$ respectively. We present the sample-wise KL-divergence in Equation 3.9, with ln being the natural logarithm. For the KL-divergence across all samples i, the mean is taken.

$$L_{KL}\left(\widehat{\boldsymbol{Y}}_{\boldsymbol{i}}, \boldsymbol{Y}_{\boldsymbol{i}}\right) = \sum_{j=1}^{N_{\text{cell types}}} Y_{ij} \cdot \ln \frac{Y_{ij}}{\widehat{Y}_{ij}} \tag{3.9}$$

For the discrimination task, the overall model M_d can be represented as $\widehat{\boldsymbol{y}}_{domain} = D \circ E(\boldsymbol{X})$ trained against ground truth labels $\boldsymbol{y}_{domain}$, where each element y_{domain_i} was one of 0 and 1, For our models, we used the mean BCE across samples as the loss function of the discriminator, for which the element-wise formula is shown in Equation 3.10, where ln is the natural logarithm.

$$l_{BCE}(\hat{y}, y) = -y \ln \hat{y} - (1 - y) \ln(1 - \hat{y}) \tag{3.10}$$

In all of our own models, we used the AdamW optimizer [104], a variation on the commonly used [71] Adam [105]. AdamW corrects an issue where L2 norm regularization penalty term on the weights (often referred to as weight decay) was obliterated with large gradients. Outside of

pretraining, we defined an epoch to be based on one iteration over the target spatial transcriptomics dataset, as the randomly generated source pseudo-spots dataset could be of arbitrary size due to the n_{spots} hyperparameter. At every epoch, the pseudo-spot samples would be re-shuffled but would not iterate over the whole dataset, as we set $n_{spots} = 100{,}000$, much larger than any one spatial dataset. As potentially different random samples would be seen every epoch, this introduced a degree of limited data augmentation to our method. To reduce the variability caused by tuning batch sizes, we set a minimum alternative batch size of 512 for pretraining for CellDART and ADDA.

To perform hyperparameter tuning, we performed a random search, selecting a subset from the total set of possible hyperparameters we defined for each model. The number of hyperparameter combinations assessed, specific to each model and dataset, was dependent on available training resources. Using the reserved validation slide, we validated the performance using the reserved real-spot metric specific to each dataset that we defined in Section 3.3.3. For data inputs, we assessed the same common set of hyperparameters for all models, differing only across datasets; these are shown in Table 3.3, while model-specific hyperparameters are detailed in the following subsections. Ranges for n_{mix} were selected for average abundances of cells for each of the three spatial transcriptomics technologies used.

Table 3.3: Common data hyperparameters used for all model types.

Hyperparameter	**Dataset (if applicable)**	**Values (set)**
n_{mix}	dlPFC	[3, 5, 8, 10]
	PDAC	[30, 50, 70]
	mouse cortex	[5, 8, 10, 15]
$n_{markers}$		[20, 40, 80]
Normalization		[min-max, standard]

In the following subsections, we detail specific aspects of each of our models and how we trained them. Some commonalities and general rules we abided by include:

- For our adversarial models (DANN and ADDA), we used leaky ReLU for hidden layer activations, a variation of the rectified linear unit (ReLU) (a commonly used hidden activation [71]) that, instead of clamping negative inputs to 0, scales them by a factor of 0.01 [106]. This activation function was found by Radford $et\ al.$ to work well in GANs [107]. For CellDART, we used the same activation function of the original method, the exponential linear unit (ELU).
- While we initially only intended to use min-max normalization to evaluate CellDART in its default configuration, we found that CellDART universally performed better with min-max normalized data. We thus added scaler a hyperparameter to be tuned for all models.
- For our adversarial models (DANN and ADDA), we set the momentum parameter β_1 of the AdamW optimizer for the discrimination task to 0.5, as opposed to the default 0.9. This was found by Radford $et\ al.$ to reduce oscillations between the discriminator and generator and better achieve stability, as it allows gradients for each to react more quickly to changes [107].
- For all our models, we used batch normalization [108], commonly used to help stabilize parameter updates [71], after the hidden layer activations. We did not use batch normalization on the outputs of encoders, as suggested by Radford *et al.* for GANs, as they found that this resulted in instability between the generator and discriminator.
- Batch size and batch normalization momentum were fixed to 8 and 0.1 respectively for the mouse dataset due to the small number of samples within that dataset.

3.4.1 Reimplemented CellDART

We reimplemented, to the best of our abilities, CellDART in PyTorch, as it had originally been implemented in Keras [109]. We explain the implementation here; a detailed set of the specifications are shown in Table 3.4. Where not specified, configurations values are either default PyTorch values or are specified later in Table 3.5 as hyperparameters.

Table 3.4: Specifications of CellDART. Where not specified, configurations values are either default PyTorch values or hyperparameters sets in Table 3.5; hyperparameter placeholders are indicated with an underline.

Parameter type and component, if applicable		**Configuration**	**Value**
Model	Encoder	Layer sizes (input + hidden layers + output)	(N_{genes}, 1024, $\underline{d_e}$)
	Discriminator	Layer sizes (input + hidden layers + output)	($\underline{d_e}$, 32, 2)
		Hidden dropout rate	0.5
		Output activation	Softmax
	Predictor	Layer sizes (input + hidden layers + output)	($\underline{d_e}$, $N_{cell\ types}$)
		Output activation	Softmax
	Whole model	Batch normalization configuration	ϵ: 0.001
		Activation unless otherwise specified	ELU
Optimizer (Adam)		Optimizer configuration	(β_1, β_2): (0.9, 0.999) ϵ: 1×10^{-7}
Training	Pretraining	Epochs	10
	Adversarial	Iterations	15,000

Some aspects to note about this specification include:

- The CellDART authors encoded the domain labels $\boldsymbol{Y}_d$ in one-hot format, hence the 2 output neurons, softmax function, and categorical cross-entropy instead of BCE.

- The authors used the default Keras parameters for Adam [105] and batch normalization [108], which are different from PyTorch defaults; here we manually set them to Keras defaults.
- CellDART's original configuration was to train for 3000 iterations; as we, unlike CellDART, treated all training slides as part of one dataset, we increased the iterations to 15,000.
- As in the original implementation, instead of epochs, batches were randomly sampled, with replacement, from each of the source and target sets, and each were concatenated sample-wise together, masking out the target samples in the set for the prediction task.

CellDART did not use any form of early stopping or validation set during training, but we still performed hyperparameter tuning as with our models. A table of the hyperparameter values explored in each of the 3 datasets are shown in Table 3.5.

Table 3.5: Reimplemented CellDART hyperparameters validated. α, as used in CellDART, is a scaling factor for the discriminator classification loss L_d, like λ in DANN. α_{LR} is a scaling factor for the discriminator's learning rate against the main learning rate, with 5 being reported as a good value by CellDART's authors.

Hyperparameter	**Dataset (number of random search configurations)**		
	dfPFC (200)	PDAC (200)	Mouse cortex (200)
d_e	[32,64]		
α	[0.1, 0.6, 1.0, 2.0]		
α_{LR}	[1, 2, 5, 10]		
Learning rate	[0.01, 0.001, 0.0001]		
Batch size	[256, 512, 1024]		8 (fixed)
Batch norm momentum	[0.01, 0.1, 0.9, 0.99]		0.1 (fixed)

Note that PyTorch and Keras define momentum for batch normalization differently, where they are the complement of each other. Keras's default momentum is 0.99, hence 0.01 here. We additionally tried a range of values to explore the effects of changing this hyperparameter.

3.4.2 ADDA

As ADDA fully relies on the pretraining phase for the prediction task, we performed a more comprehensive pretraining on source pseudo-spots. We first increased the depth of the encoders, while adding dropout to the source encoder; specific configurations are shown in Table 3.6 and Table 3.7. We then extended the number of pretraining epochs from 10 to 200 and applied a one-cycle learning rate scheduler to maximize convergence [110], as implemented in PyTorch. This scheduler smoothly ramps the learning rate from $1/25^{th}$ to the full learning rate over the first 30% of training, then anneals the learning to $1/10{,}000^{th}$ of the maximum over the remaining 70%. The motivation is to first "warm up" any optimizers and batch normalization layers, then ramp up to the set learning rate to quickly seek out appropriate minima, before settling at a lower learning rate to fine-tune the model. We additionally performed early stopping by measuring the loss at the end of each epoch on the source validation set and saving the state at the best validated epoch, using that as the final model.

During adversarial training, at each alternation between training the discriminator and target encoder, we passed source and target and performed backpropagation and weight updates separately, one after another, to avoid initial batch effects confounding batch normalization. Leveraging separate batch normalization is important to prevent initially highly separable distributions causing the discriminator to overfit. This causes the encoder/generator to simply produce a single result the discriminator cannot generalize to, a condition called mode collapse [107]. Additionally, as the authors of ADDA state that due to its lack of weight sharing, the target encoder depends greatly on being pre-initialized to the source encoder weights [29], it thus clear and also for reasons we will further explore in Section 5.1.2 that ADDA depends heavily on its

starting conditions. We thus limited the number of adversarial training epochs to just 20, which we later show in Section 4.4 to be more than sufficient.

We performed initial testing to find configurations and ranges of values that appeared to work well in producing a stable adversarial model. To avoid the situation where compromised embeddings produced by the target encoder are used to train the discriminator, we disabled dropout in the target encoder. For the discriminator, we had to ensure that it would consistently be capable of, to use a chess analogy, "check" the encoder. To match the target encoder, we set the number of hidden layers and neurons in the discriminator to be the same as the encoder but in reverse. We further added two factors in training to help the discriminator react to new kinds of inputs as presented by the encoder. The first, α_{LR}, is a hyperparameter that scales up the learning rate of the discriminator relative to the target encoder; a similar hyperparameter was also used in CellDART [27]. Next, we added a "discriminator loop factor" K_d defining how many iterations the discriminator should be trained before the encoder's turn to train for a single iteration. A full list of the model specifications that we found worked well during initial testing and fixed during hyperparameter tuning are shown in Table 3.6, while hyperparameter values that we validated for are shown in Table 3.7.

Table 3.6: Specifications of ADDA. Where not specified, configurations values are either default PyTorch values or hyperparameters sets in Table 3.7; hyperparameter placeholders are indicated with an underline.

Parameter type and component, if applicable		**Configuration**	**Value**
Model	Source encoder	Layer sizes (input + hidden layers + output)	(N_{genes}, 1024, 512, $\underline{d_e}$)
		Hidden dropout rate	0.5
	Target encoder	Layer sizes (input + hidden layers + output)	(N_{genes}, 1024, 512, $\underline{d_e}$)
		Hidden dropout rate	0.0

	Discriminator	Layer sizes (input + hidden layers + output)	($\underline{d_e}$, 512, 1024, 1)
		Hidden dropout rate	0.5
		Output activation	Logistic
	Predictor	Layer sizes (input + hidden layers + output)	($\underline{d_e}$, $N_{cell\ types}$)
		Output activation	Softmax
	Whole model	Batch normalization configuration	ϵ: 0.001
		Activation unless otherwise specified	Leaky ReLU
Optimizer (AdamW)	Pretraining	Optimizer configuration	(β_1, β_2): (0.9, 0.999) ϵ: 1×10^{-7}
	Adversarial	Optimizer configuration	(β_1, β_2): (0.5, 0.999) ϵ: 1×10^{-7}
Scheduler	Pretraining	Scheduler type	One cycle
Training	Pretraining	Epochs	200
	Adversarial	Epochs	20

Table 3.7: Validated ADDA hyperparameters. d_e is the dimension of the embedding at the output of the encoder. α_{LR} is a scaling factor for the discriminator's learning rate vs. the main learning rate. K_d is the discriminator loop factor.

Hyperparameter	**Dataset (number of random search configurations)**		
	dfPFC (200)	PDAC (200)	Mouse cortex (1000)
d_e	[32, 64]		
α_{LR}	[1, 2, 5]		
Pretraining learning rate	[0.0001, 0.0002, 0.001]		
Adversarial learning rate	[0.01, 0.001, 0.0001]		
K_d	[2, 5, 10]		
Batch size	[512, 1024, 2048]	[256, 512, 1024]	8 (fixed)
Batch norm momentum	[0.01, 0.1]		0.1 (fixed)

To note, regarding the hyperparameters in Table 3.7: our initial testing for dlPFC and ADDA seemed to achieve stability more readily at larger batch sizes, so we increased the range of possible

batch sizes compared to other methods. This was not possible in PDAC, as 1,024 samples would be already larger than the entire dataset, so we left the batch size for that set as-is.

3.4.3 DANN

Like with ADDA, we trained using a discriminator that mirrored the structure of the encoder. On the other hand, due to the nature of training the whole model, including both the prediction and the domain discrimination task in a single pass, we encountered multiple issues. The first was that, during initial testing, we found DANN much more difficult than ADDA to achieve a stable state, as the encoder was responsible for both tasks at once. The second was how to deal with batch normalization.

To address the stability concerns as well as squaring two different training tasks, we defined conditions for stability of the model and tied them to early stopping as well as a learning rate scheduler. We defined stability by calculating the training accuracy over the course of the epoch, and then checking whether $0.5 < Accuracy_{D_{train}} < 0.6$. for both domains $D \in \{S, T\}$ Then, at the end of the epoch we checked using the validation set whether $0.3 < Accuracy_{D_{val}} < 0.8$. Both ranges are shifted upward as <50% accuracy on average would indicate predicting the opposite class of what is correct, which is undesirable, while the criterion for validation is looser to account for variance. If all four conditions are met, then we considered that epoch to be stable. Then, at the end of every epoch, we perform the pseudo-Python Algorithm 3.1, which depends on whether a stable epoch had been found previously, whether the current epoch is stable, and the current validation loss on the prediction task.

Algorithm 3.1

```
best_val_loss = +inf
stable_found = False
scheduler = StepLR(PATIENCE)
```

```
for i in i...epochs-1:
    ...
    # train model
    # check if stable
    ...
    # stable and have the best stable epoch so far
    if (stable and ((curr_val_loss < best_val_loss)
    # or found the first stable epoch
    or (not stable_found)):
        best_val_loss = curr_val_loss
        save_best_model()
        scheduler.reset_counter()

    # improving in the prediction task
    elif (curr_val_loss < best_val_loss)
    # but not stable but working toward it
    and (not stable) and (not stable_found):
    # and still early in training
    and (i < GRACE_PERIOD):
        # for tracking best score during grace period
        best_val_loss = curr_val_loss
        # give more time an extension for effort on prediction task
        scheduler.reset_counter()
        # save in case no stable model is ever found
        save_best_model()
    else:
        # steps down learning rate after PATIENCE steps
        scheduler.step()
```

Algorithm 3.1 performs a form of early stopping based on validation performance on the prediction task, but the score only counts if stability conditions are met. It also includes a scheduler to monitor both stability and the prediction score and scales the learning rate by a factor of 0.5 after 50 epochs of no stable score that improves over the previous stable score. We allowed for a grace period of 100 epochs where, so long as the validation score was improving, the counter would reset to give more time for the model to begin to converge (the grace period ends after the first stable epoch has been found). We chose the model saved at the stable epoch with the best source domain cell type proportion loss as the final model; if no such model existed, then we chose the best unstable epoch. During the hyperparameter random search, we also placed a constraint

that the top *stable* model on validation must be chosen, to eliminate unstable and potentially ungeneralizable configurations.

The second issue relates to batch normalization. The DANN algorithm originally did not incorporate batch normalization and performed weight updates for all tasks and both domains at once. As explained in Section 3.4.2, batch normalization is useful when training adversarially, as this prevents stability and mode collapse. We thus added two hyperparameters: the first is whether to perform the weight updates in a single pass, or iteratively for each of target and domain as in ADDA, where the prediction task naturally would only occur in the source pass. while the second is whether to perform the source pass or target pass first. We considered the second unlikely to have any real effect, and simply included it out of interest. The specifications of DANN are shown in Table 3.8, while the hyperparameters we validated are shown in Table 3.9.

Table 3.8 Specifications of DANN. Where not specified, configurations values are either default PyTorch values or hyperparameters sets in Table 3.9; hyperparameter placeholders are indicated with an underline.

Parameter type and component, if applicable		**Configuration**	**Value**
Model	Encoder	Layer sizes (input + hidden layers + output)	(N_{genes}, hidden layers, d_e)
	Discriminator	Layer sizes (input + hidden layers + output)	(d_e, hidden layers, 1)
		Output activation	Logistic
	Predictor	Layer sizes (input + hidden layers + output)	(d_e, $N_{cell\ types}$)
		Output activation	Softmax
	Whole model	Batch normalization configuration	PyTorch defaults
		Activation unless otherwise specified	Leaky ReLU
Optimizer (AdamW)	Adversarial	Optimizer configuration	(β_1, β_2): (0.5, 0.999)

Scheduler	Adversarial	Scheduler type	Step
Training	Adversarial	Epochs	500

Table 3.9: DANN hyperparameters validated. d_e is the dimension of the embedding at the output of the encoder. λ is a scaling factor for the discriminator classification loss L_d. α is a scaling factor for the gradient as it passes through the GRL. Higher values of α mean larger encoder gradients relative to those passed from the discriminator.

Hyperparameter	**Dataset (number of random search configurations)**		
	dfPFC (200)	PDAC (200)	Mouse cortex (200)
d_e	[32, 64]		
Encoder hidden layers (reversed for discriminator)	[(512,), (1024,), (512, 512), (512, 256), (256, 128),]		
α	[1, 2, 5]		
λ	[0.5, 1, 2]		
Hidden layers dropout	[0.1, 0.2, 0.5]		
Discriminator dropout factor	[0.5, 1]		
Adversarial learning rate	[2×10^{-3}, 1×10^{-3}, 2×10^{-4}, 1×10^{-4}, 2×10^{-5}, 1×10^{-5}]		
Batch size	[256, 512, 1024]	[128, 256, 512]	8 (fixed)
AdamW weight decay	[0.01, 0.1]		
Separate iterations for source and target data	[True, False]		
If separate iterations, whether first iteration should be source or target	[Source, Target]		

Note that we included a large variety of hyperparameters and ranges, as no pattern could be ascertained as to what values lead to stable training.

3.4.4 CORAL

An important consideration was how to incorporate CORAL loss. Deep CORAL enables the use of multiple hyperparameters λ_i, with i indexing the output, pre-activation, of a layer, using Equation 3.11, where L_Y is the loss on the prediction task, L is the overall loss, and t is the number

of layers in the network. In Deep CORAL, the authors suggested using the logits just prior to the softmax output activation as a starting point, and then adding deeper layers [31]. We performed initial testing to determine appropriate values of λ_{logits}. For our hyperparameter search, we then also included $\lambda_e = 0$ or $\lambda_e = \lambda_{logits}$ to control the CORAL loss at the output of the encoder. For the specifications of CORAL we used in this model, refer to Table 3.10; for the full set of random search hyperparameters, refer to Table 3.11.

$$L = L_Y + \sum_{i=1}^{t} \lambda_i L_{CORAL} \tag{3.11}$$

Table 3.10: CORAL specifications. Where not specified, configurations values are either default PyTorch values or hyperparameters sets in Table 3.11; hyperparameter placeholders are indicated with an underline.

Parameter type and component, if applicable		**Configuration**	**Value**
Model	Encoder	Layer sizes (input + hidden layers + output)	(N_{genes}, hidden layers hyperparameter, $\underline{d_e}$)
	Predictor	Layer sizes (input + hidden layers + output)	($\underline{d_e}$, $N_{cell\ types}$)
		Output activation	Softmax
	Whole model	Batch normalization configuration	PyTorch defaults
Optimizer (AdamW)		Optimizer configuration	PyTorch defaults
Scheduler		Scheduler type	One cycle
Training		Epochs	200

Table 3.11: CORAL hyperparameters validated. d_e is the dimension of the embedding at the output of the encoder, while (λ_e, λ_{logits}) are CORAL loss weighting parameters for the encoder output and the predictor output, respectively.

Hyperparameter	**Dataset (number of random search configurations)**		
	dfPFC (200)	PDAC (200)	Mouse cortex (1000)
d_e	[32,64]		

Encoder hidden layer sizes	[(1024, 512), (512, 256), (256, 128), (512, 256, 128)]		
Activation unless otherwise specified	[Leaky ReLU, ReLU]		
$(\lambda_e, \lambda_{logits})$	[(0, 50), (0, 100), (0, 200), (50, 50), (100, 100), (200, 200)]		
Hidden layers dropout	[0.1, 0.2, 0.5]		
Learning rate	[0.01, 0.001, 0.0001]		
Batch size	[256, 512, 1024]	[128, 256, 512]	8 (fixed)
AdamW weight decay	[0.1, 0.3, 0.5]		

As this was not an adversarial model, training Deep CORAL was uncomplicated, and so we took advantage of the one-cycle scheduler [110] we also used in the pretraining phase of ADDA. Because CORAL loss requires embeddings from both domains, we naturally trained the whole model end to end for both source and target at once. As both L_Y (KL-divergence) and L_{CORAL} were normalized, we did not scale the losses to account for fewer data in the prediction task.

4. Results

4.1 Overview

In this chapter, we report the results of our investigation, and comment on any particularly interesting results. In Section 4.2, we report overall performance results on the 3 levels of evaluation we defined in Section 3.3. In Section 4.3, we report the final hyperparameters for each model and dataset that were selected. We review and comment on adversarial training in Section 4.4 and domain adaptation in Section 4.5. Finally, in Section 4.6, we compare and validate our re-implementation of CellDART against its original implementation.

4.2 Overall performance

To obtain the final performances of each model, we performed 5 bootstraps of each model using the hyperparameters found, generating different sets of pseudo-spots and using different seeds for each. We report both mean and standard deviation values for the summary results. We first tackle all the results in summary at each level of evaluation as defined in Section 3.3. Due to its architecture, RCTD results are shown only for level 3 (target domain) predictions.

4.2.1 Performance on source domain data

Table 4.1: Test performance of each model on synthetic scRNA-seq pseudo-spots. Values are mean cosine distance across bootstraps, lower values being better; standard deviation across 5 bootstraps are shown in braces.

Source cosine distance (↓better)	**Dataset**		
Model	**dlPFC**	**PDAC**	**Mouse cortex**
Our CellDART	0.29581 (0.04825)	0.03794 (0.00311)	0.07141 (0.01079)
ADDA	**0.16617 (0.00098)**	**0.01225 (0.00006)**	**0.02749 (0.00011)**
DANN	0.17357 (0.00103)	0.10759 (0.07501)	0.08469 (0.00472)
CORAL	0.24273 (0.01216)	0.07106 (0.00225)	0.55217 (0.03186)

As shown in Table 4.1, on source domain data, ADDA performed the best out of all our methods plus CellDART. This is unsurprising, as ADDA's source encoder and predictor are only trained on the source prediction task. The rest of the methods perform at around the same level, except with CORAL on the mouse cortex set; while the other methods demonstrate distances below 0.1, CORAL achieves a cosine distance of ~0.55, failing in the prediction task. This could possibly indicate fitness issues in the calculation of covariance due to the small batches of size 8 used for this dataset.

4.2.2 Evaluation of domain adaptation

Table 4.2: RF50 scores of each model by dataset using embeddings on test samples from both source and target, except for PDAC, where training samples were used. Values are expressed as mean accuracies of the random forest model on a holdout set of 20%. Standard deviation across 5 bootstraps is expressed in braces. A value closer to 0.5 is better.

Source×target RF50 (↓better)	**Dataset**		
Model	**dlPFC**	**PDAC**	**Mouse cortex**
CellDART (ours)	0.9991 (0.00055)	0.99977 (0.00014)	0.99998 (0.00005)
ADDA	0.94986 (0.01622)	0.88372 (0.02075)	0.78032 (0.20845)
DANN	0.99474 (0.00421)	**0.66653 (0.03227)**	0.98954 (0.00615)
CORAL	**0.91612 (0.01835)**	0.99972 (0.00045)	**0.71642 (0.12817)**

Table 4.3: miLISI scores between source and target test samples from both source and target. except for PDAC, where training samples were used. Values are expressed as the mean miLISI across 5 bootstrap models, while standard deviation is indicated in braces. A value closer to 2 is better.

Source×target miLISI (↑better)	**Dataset**		
Model	**dlPFC (perplexity=30)**	**PDAC (perplexity=30)**	**Mouse cortex (perplexity=5)**
CellDART (ours)	1.0 (0.0)	1.33019 (0.1097)	1.02483 (0.02368)
ADDA	**1.25775 (0.03382)**	1.81931 (0.01604)	1.2893 (0.11881)
DANN	1.09193 (0.12809)	**1.84353 (0.01732)**	1.09244 (0.02474)

CORAL	1.23499 (0.06981)	1.0 (0.0)	**1.51268 (0.15324)**

In terms of our domain variance metrics, we note several items in Table 4.2 and Table 4.3; note also that while this was comparing test samples from source and target sets, as there was no test set in the PDAC target set, we compared test source data with target training data for that instead. The first is that CellDART fails in the domain invariance task, with near perfect (i.e., poor) scores in RF50 across all datasets and miLISI scores close to 1, indicating low local neighbourhood mixing, with the exception being PDAC with an effective 1.33 classes dominating each sample's local neighbourhood. This is reflected upon PCA visualization of the embeddings in Fig. 4.1, where CellDART's embedding shows almost linear separability in the first two principal components. The second is that CORAL performs the best on both tasks on the mouse cortex dataset, indicating that, for that dataset, it performed well at eliminating domain discrepancy in its embeddings, Furthermore, ADDA and DANN perform well in both metrics on the PDAC dataset, but DANN performs poorly on the other datasets. Both ADDA and CORAL perform well in these metrics on the dlPFC dataset, but neither perform exceptionally well, with RF50 still being able to discriminate most embeddings from both. Also of note is that ADDA had decent performance in the mouse cortex dataset. We must note, however, as the size of the test dataset was small (just 18 samples, 9 from source and 9 from target), we had to reduce the perplexity of the miLISI score relative to the other datasets, making the scores not directly comparable across datasets.

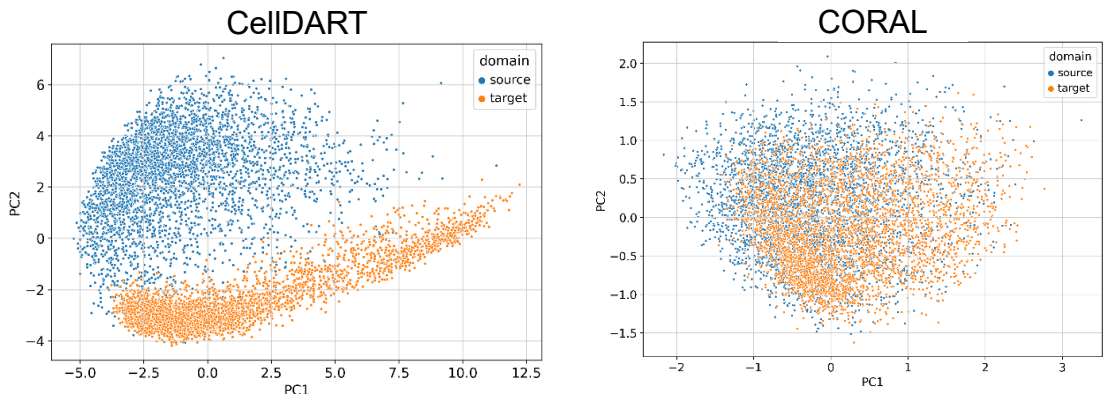

Fig. 4.1: Comparison of embeddings produced using CellDART and CORAL. Target slide was test slide 151675, and source data were the test set of pseudo-spots.

4.2.3 Evaluation of deconvolution of real spots

We now evaluate how these first levels of evaluation translate to real spot performance, as shown in Table 4.4. Firstly, as the best performer in the source domain was ADDA in all cases, and ADDA was one of top two performers in the domain discrepancy metrics in both the dlPFC and mouse cortex sets, one might expect that ADDA would be able to generalize well in the target task. In practice, while ADDA performed the best among the deep models tested in the mouse cortex (second including RCTD), it performed near the bottom of methods in the dlPFC task. It should also be noted that, while dlPFC is the most realistic dataset, its ground truths are also the least comprehensive, only measuring the location so one cell type, whose identities were determined from the enrichment of markers that would presumably also be in the input data. Secondly, we note that although CellDART was not a particularly well performing method in the source domain deconvolution task and was by far the worst in RF50 and miLISI, it was the only method to achieve greater than 50% AUC on the PDAC dataset, despite ADDA outperforming it on that dataset in both previous tasks. We point out, however, that CellDART was not much better, and these in fact could all be erroneous results as a result of data leakage, as the PDAC set did not have separate test and training samples for target data. We finally bring attention to CORAL's performance; although it performed best on dlPFC, consistent with its domain integration performance, it

performed the worst on real spots on the gold standard data, reflecting its poor performance while training on synthetic spots.

To evaluate RCTD, we used raw counts for the scRNA-seq reference data and passed in unscaled and pre-log-transformed data. As RCTD filters cell types with <25 scRNA-seq samples, when evaluating we filled in missing cell type proportions as 0. RCTD also did not generate predictions for some spots. For AUROC we set the predicted proportions for all cell types to 0 for missing spots. Filling values of 0s for predicted proportions has the effect of ensuring that, until FPR reaches 1 on the right edge of the ROC curve, the TPR will be short those samples, excluding that proportion of samples from the AUROC. For cosine distance, we set scores for samples that were missing to 1, representing complete orthogonality between predicted and ground truth cell type proportions.

We were not able to obtain intermediate nor bootstrap results, so Table 4.4 only includes the mean value for a single run using the target test sample. RCTD performed no better than a random predictor for dlPFC with an AUROC of ~0.5; we note also that none of the excitatory neuron cell types evaluated had fewer than 25 samples. The performance on the PDAC set was no better or worse than any other model, while for the Spotless "gold standard 1" mouse cortex sample, it performed the best by a margin of just over a standard deviation of the next best, ADDA.

*Table 4.4: Performance on real spatial transcriptomics spots. Standard deviation, where available, is indicated in branches. The best performing deep UDA methods are indicated with an underline, while the best including the RCTD baseline is indicated in **bold**.*

Target	**Dataset**		
Model	**dlPFC AUROC (↑better)**	**PDAC training AUROC (↑better)**	**Mouse cortex cosine distance (↓better)**
RCTD	0.49709	0.5	**0.12118**
CellDART (ours)	0.67013 (0.02787)	**0.51736 (0.00707)**	0.14755 (0.0764)
ADDA	0.65349 (0.01498)	0.49236 (0.00141)	0.13697 (0.01517)

DANN	0.64932 (0.05255)	0.49494 (0.00578)	0.14784 (0.04081)
CORAL	**0.6958 (0.03309)**	0.4949 (0.00523)	0.2383 (0.03621)

4.3 Selected hyperparameters

In this section are the selected hyperparameters of each model by dataset, after we performed tuning via random search based on the candidate hyperparameters in Section 3.4. We briefly comment on any interesting results.

Table 4.5: CellDART hyperparameters selected via random search. α, as used in CellDART, is a scaling factor for the discriminator classification loss L_d, like λ in DANN. α_{LR} is a scaling factor for the discriminator's learning rate against the main learning rate.

Hyperparameter	**Dataset (number of random search configurations)**		
	dfPFC (200)	**PDAC (200)**	**Mouse cortex (200)**
n_{mix}	3	50	10
$n_{markers}$	40	80	80
Normalization	Min-max	Min-max	Min-max
d_e	32	32	32
α	2	1	0.6
α_{LR}	10	5	5
Learning rate	0.0001	0.001	0.001
Batch size	512	256	8 (fixed)
Batch norm momentum	0.01	0.9	0.1 (fixed)

Table 4.6: ADDA hyperparameters selected via random search. ADDA hyperparameters. d_e is the dimension of the embedding at the output of the encoder. α_{LR} is a scaling factor for the discriminator's learning rate vs. the main learning rate. K_d is the discriminator loop factor.

Hyperparameter	**Dataset (number of random search configurations)**		
	dfPFC (200)	**PDAC (200)**	**Mouse cortex (1000)**
n_{mix}	3	50	5
$n_{markers}$	40	20	80

Normalization	Min-max	Min-max	Min-max
d_e	32	32	64
α_{LR}	1	2	1
Pretraining learning rate	0.001	2×10^{-4}	1×10^{-4}
Adversarial learning rate	2×10^{-4}	2×10^{-4}	2×10^{-6}
K_d	2	10	10
Batch size	2048	512	8 (fixed)
Batch norm momentum	0.01	0.01	0.1 (fixed)

Table 4.7: DANN hyperparameters selected via random search. d_e is the dimension of the embedding at the output of the encoder. λ is a scaling factor for the discriminator classification loss L_d. α is a scaling factor for the gradient as it passes through the GRL. Higher values of α mean larger encoder gradients relative to those passed from the discriminator.

Hyperparameter	**Dataset (number of random search configurations)**		
	dfPFC (200)	**PDAC (200)**	**Mouse cortex (200)**
n_{mix}	3	70	5
$n_{markers}$	80	40	20
Normalization	Min-max	Standard	Min-max
d_e	64	32	64
Encoder hidden layers (reversed for discriminator)	(512,256)	(512,256)	(256,128)
α	1	1	2
λ	2	1	0.5
Hidden layers dropout	0.2	0.2	10
Discriminator dropout factor	1	1	1
Adversarial learning rate	0.002	1×10^{-5}	0.1 (fixed)
Batch size	256	512	8 (fixed)
AdamW weight decay	0.1	0.1	0.1
Separate iterations for source and target data	False	False	True

If separate iterations, whether first iteration should be source or target	Target	Target	Target

Table 4.8: CORAL hyperparameters selected via random search. d_e is the dimension of the embedding at the output of the encoder, while (λ_e, λ_{logits}) are CORAL loss weighting parameters for the encoder output and the predictor output, respectively.

Hyperparameter	**Dataset (number of random search configurations)**		
	dfPFC (200)	**PDAC (200)**	**Mouse cortex (1000)**
n_{mix}	10	30	10
$n_{markers}$	80	80	20
Normalization	Min-max	Standard	Standard
d_e	32	64	64
Encoder hidden layer sizes	(256,128)	(512,256,128)	(256,128)
Activation unless otherwise specified	Leaky ReLU	Leaky ReLU	ReLU
(λ_e, λ_{logits})	(200,200)	(200,200)	(0,100)
Hidden layers dropout	0.2	0.5	0.5
Learning rate	0.01	0.001	1×10^{-4}
Batch size	512	512	8 (fixed)
AdamW weight decay	0.3	0.3	0.1

One notable aspect that we found was that for the dlPFC and PDAC sets, the weights ($\lambda_e, \lambda_{logits}$) for the CORAL loss were the maximum possible options along with the weight decay, whereas with the mouse cortex data, lower values of λ were selected.

4.4 Adversarial training results

As discussed previously, adversarial training is inherently a two-sided problem and is prone to reaching a state where it does not converge. We show, as an example of stable training, the accuracies of ADDA on the mouse cortex datasets, which was a performant model in all metrics and the top performer on the Spotless "gold standard 1" set, in Fig. 4.2. As is visible, although the

accuracy is noisy, due to the small batch sizes used, the accuracy during the training phase for all components (discriminator and encoder) on both datasets remain stable and do not individually converge across all 20 epochs.

Fig. 4.2: Accuracy of discriminator and encoder of ADDA; it was not possible to separately measure the accuracy of the encoder during evaluation and so is left out of the validation subfigure.

We further investigated the performance ADDA during training on the spotless gold standard dataset by testing its deconvolution performance at every epoch checkpoint shown in Fig. 4.3, including continued training for another 80 epochs (100 total), where the used model at the 20th epoch (epoch 19) is indicated with a vertical line. We found that its performance had reached its final equilibrium point in a single epoch. Interestingly, both the train and test performance decreased over the pretrained model (shown as epoch "-1"). This is in line with our earlier conjecture in Section 3.4.2, as well as the ADDA authors' comments [29], that as ADDA is highly dependent on pretraining and the target encoder's weights are not shared with the source encoder nor trained in any prediction task, that the target encoder is highly sensitive to initialization.

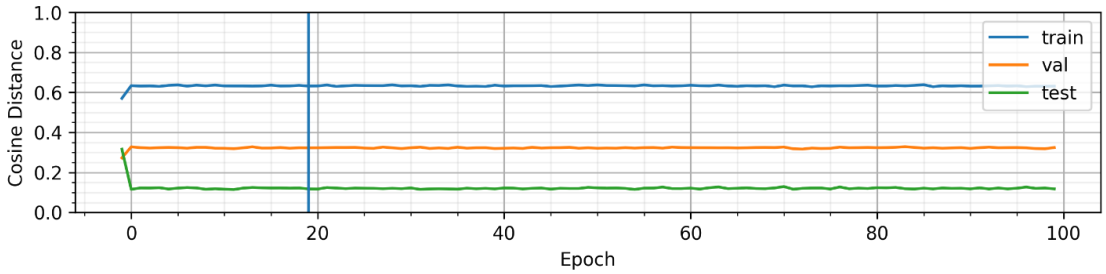

Fig. 4.3: Performance shown as cosine distance (lower=better) of ADDA on Spotless "gold standard 1" mouse cortex spots at each epoch of adversarial training when extending to 100 epochs, performance before training is shown at epoch "-1", while the 20th epoch (epoch 19) that was the final model is indicated with a vertical line.

An example of a model that had much more difficulty converging, at least in our investigation, was DANN. In Fig. 4.4, we show the accuracy at the discriminator on training and validation data across both sets separately; as neither discriminator nor encoder can be separated to show different results, this reflects both optimization tasks. This was the model trained using the hyperparameters that were eventually selected and was considered stable for the epoch chosen by our criteria from Section 3.4.3. Even so, large oscillations from predicting mostly source and target were seen. We generally had more difficulty training DANN than ADDA; out of 600 models trained across the 3 datasets (dlPFC, PDAC, Spotless mouse cortex), only 301 ever met the stability criteria at any epoch. An example of DANN that did not every reach stability is shown in Fig. 4.5; as accuracy steadily increased, this potentially is an example of modal collapse because of the discriminator overfitting.

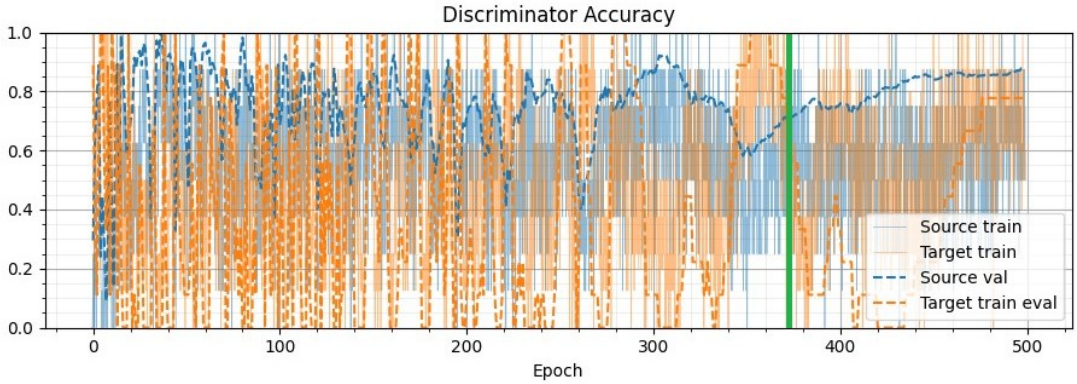

Fig. 4.4: Accuracy at each iteration (end of epoch for validation) on domain discrimination task for final DANN model on Spotless "gold standard 1" mouse cortex dataset. Selected epoch 372 indicated with a vertical bar. Note that as

training of the encoder of the encoder and discriminator were not performed separately, we were unable to individually track their accuracies.

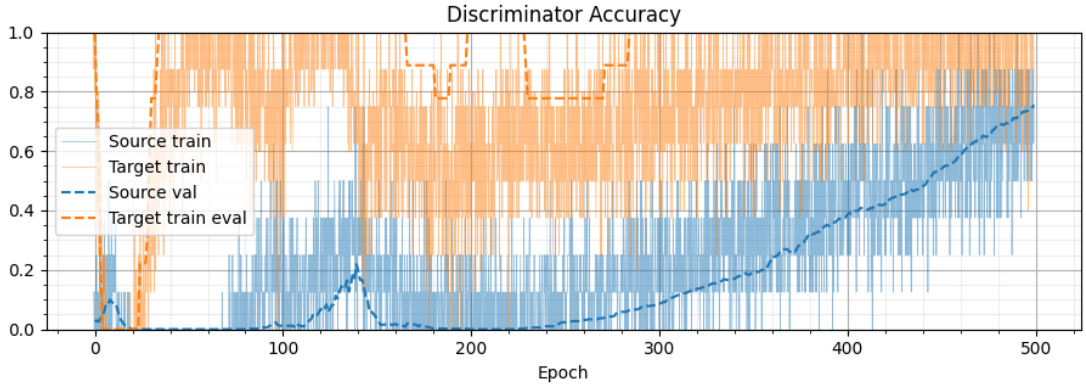

Fig. 4.5: Potential mode collapse in the generator of DANN. This can be seen by the accuracy for both domains steadily increasing over training, indicating overfitting of the discriminator.

4.5 Effect of domain adaptation

Table 4.9: Performance of CellDART and ADDA on source pseudo-spots comparing pretraining only and after domain adaptation stage. Values are cosine distance, where lower is better.

Model	**Stage**	**dlPFC**	**PDAC**	**Mouse cortex**
CellDART (ours)	Pretrain	0.20242 (0.00188)	0.00972 (0.00028)	0.04599 (0.00237)
	After UDA	0.29581 (0.04825)	0.03794 (0.00311)	0.07141 (0.01079)
ADDA	Pretrain	0.16617 (0.00098)	0.01225 (0.00006)	0.02749 (0.00011)
	After UDA	0.16617 (0.00098)	0.01225 (0.00006)	0.02749 (0.00011)

As both CellDART and ADDA had pretraining stages, we were also able to compare performance with just pretraining and after domain adaptation. Table 4.9 shows a decrease in source dataset performance for CellDART after domain adaptation due to the generalization task, while there was no change in performance for ADDA, which uses separate encoders for source and target dataset.

Table 4.10: Performance of CellDART and ADDA comparing RF50 between source and target data before and after domain adaptation stage. Values are expressed as discrimination accuracies, where lower is better. Test samples were used for both sets, except for PDAC where training samples were used as target samples.

Model	Stage	dlPFC	PDAC	Mouse cortex
CellDART (ours)	Pretrain	0.99226 (0.0032)	0.99998 (0.00003)	0.99922 (0.0006)
	After UDA	0.9991 (0.00055)	0.99977 (0.00014)	0.99998 (0.00004)
ADDA	Pretrain	0.99196 (0.00202)	0.99918 (0.00132)	1.0 (0.0)
	After UDA	0.94986 (0.01622)	0.88372 (0.02075)	0.78032 (0.20845)

Table 4.11: Performance of CellDART and ADDA comparing miLISI scores between source and target data before and after domain adaptation stage. Test samples were used for both sets, except for PDAC where training samples were used as target samples. Perp.=perplexity, higher is better.

Model	Stage	dlPFC (Perp.=30)	PDAC (Perp.=30)	Mouse cortex (Perp.=30)
CellDART (ours)	Pretrain	1.0 (0.0)	1.0 (0.0)	1.03059 (0.0212)
	After UDA	1.0 (0.0)	1.33019 (0.1097)	1.02483 (0.02368)
ADDA	Pretrain	1.0 (0.0)	1.00101 (0.00126)	1.00084 (0.00062)
	After UDA	1.25775 (0.03382)	1.81931 (0.01604)	1.2893 (0.11881)

In both miLISI and RF50 as shown in Table 4.10 and Table 4.11 respectively, CellDART and ADDA both showed near complete separability between source and target embeddings after only pretraining. After domain adaptation, while ADDA showed modest (in dlPFC) to good (on PDAC for miLISI) improvements, CellDART only showed any improvement in PDAC in miLISI, while demonstrating no improvement in RF50. This shows that ADDA's adversarial training phase achieved its purpose, while CellDART failed to do the same.

Table 4.12: Performance of CellDART and ADDA comparing miLISI scores on real spots before and after domain adaptation stage. Test samples were used for both sets, except for PDAC.

Model	Stage	dlPFC AUC (↑better)	PDAC training AUC (↑better)	Mouse cortex cosine distance (↓better)

CellDART (ours)	Pretrain	0.60007 (0.01197)	0.49037 (0.00364)	0.22675 (0.06909)
	After UDA	0.67013 (0.02787)	0.51736 (0.00707)	0.14755 (0.0764)
ADDA	Pretrain	0.59167 (0.01418)	0.49461 (0.00163)	0.30749 (0.06652)
	After UDA	0.65349 (0.01498)	0.49236 (0.00141)	0.13697 (0.01517)

In terms of final performance, Table 4.12 shows an improvement in performance over just pretraining for ADDA in both the dlPFC and Spotless mouse cortex slides, but no improvement for the PDAC set. As Fig. 4.3 shows, however, this improvement only occurred in the test slide, with a decrease in performance in both the training and validation slides. In addition, the scores reached their final resting positions within a single epoch. These points put ADDA's supposed improvement in target domain performance due to adversarial training in question, which we discuss in Section 5.1.3.

4.6 Comparison with original CellDART

To ensure that comparison we made of our methods with CellDART in Section 4.1 was valid, we also compared the performance of our implementation against that produced by CellDART, shown for all metrics and datasets in Table 4.13. While the performance of either were not equal, neither performed better overall across all datasets or all metrics, and were generally within a standard deviation of one another, affirming that our reimplementation was true to the original.

Table 4.13: Comparison of test performance between our reimplementation of CellDART with the original code, using the selected hyperparameters shown in Table 4.5. In each dataset and metric, the better performing score of the two is bolded and underlined. Standard deviation is shown in braces.

Metric	**Dataset**	**Our CellDART**	**Original CellDART**
Source cosine distance (↓better)	dlPFC	0.29581 (0.04825)	**0.24922 (0.018)**
	PDAC	**0.03794 (0.00311)**	0.06093 (0.01248)
	Mouse cortex	**0.07141 (0.01079)**	0.07385 (0.01074)

Source×target RF50 (↓better)	dlPFC	0.9991 (0.00055)	**0.99605 (0.00373)**
	PDAC	**0.99977 (0.00014)**	0.99988 (0.00009)
	Mouse cortex	0.99998 (0.00004)	**0.99996 (0.00009)**
Source×target miLISI (↑better)	dlPFC (perp.=30)	1.0 (0.0)	**1.04271 (0.05164)**
	PDAC (perp.=30)	**1.33019 (0.1097)**	1.09925 (0.06027)
	Mouse cortex (perp.=5)	1.02483 (0.02368)	**1.05944 (0.03526)**
Real spots	dlPFC AUC (↑better)	**0.67013 (0.02787)**	0.63081 (0.00978)
	PDAC AUC (↑better)	0.51736 (0.00707)	**0.51978 (0.00563)**
	Mouse cortex cosine distance (↓better)	**0.14755 (0.0764)**	0.20848 (0.1336)

5. Discussion and Future Work

5.1 Analysis of results

As we show in Section 4.2.3, while we were able to produce models (ADDA and CORAL) that outperformed CellDART in 2 out of the 3 datasets tested (dlPFC and Spotless mouse cortex), and CORAL outperformed RCTD on the dlPFC dataset, we were unable to show an improvement conclusively or consistently. We can eliminate the PDAC results, as it did not use a test sample and all models showed more-or-less the same 50% AUROC performance, indicating an evaluation protocol for real spots on PDAC (see Section 3.3.3) that was possibly flawed. Even then, CellDART outperformed both ADDA and DANN for the dlPFC set, and DANN and CORAL on the Spotless mouse cortex set, and no one method outperformed CellDART in both.

Although ADDA was the best performer among the UDA models on the Spotless "gold standard 1" mouse cortex set, which was the only dataset we used that included true cell type proportions as ground truth to evaluate against, it did not outperform RCTD. Furthermore, for both, the Spotless mouse cortex dataset contained fewer genes (10,000 versus ~20,000) for the remainder as well as fewer samples (merely 45 in the training set and 9 in validation in test), which makes it a poor fit in how it might reflect real world usage. This is especially true given that ADDA's score is not a drastic improvement over CellDART (cosine distance of 0.13697 versus 0.14755, with lower being better). For a definitive result, both ADDA and RCTD would have to perform well on at least another dataset. The reverse could be said about CORAL with the dlPFC dataset, which although, being a 10x Genomics Visium dataset, reflects the main use case for cell type deconvolution as we explain in Section 3.3.3, the usage of only excitatory neurons to evaluate means that the metric used only serves as a heuristic. Similarly, the improvement demonstrated was modest (AUROC of 0.6958 versus 0.67013, with higher being better).

5.1.1 Gradient reversal layer causes poor adversarial training stability

The only consistent conclusion in terms of real spot performance we can make is that we demonstrated that DANN was consistently the worst performer among the methods tested. As we reported in Section 4.4, it was difficult to train DANN to be stable, with nearly half (301 out of 600) of all tested hyperparameter configurations in our random search never achieving the stability criteria we outlined for DANN for early stopping and hyperparameter selection in Section 3.4.3. Additionally, compared with ADDA, when DANN did achieve stability, it did so in a manner with large oscillations. These contrasting results can be seen by comparing Fig. 4.3 for ADDA and Fig. 4.4 for DANN, each showing trends in the domain discrimination task over the course of training. In our initial testing, we were also unable to discern any clear pattern as to what ranges of hyperparameters resulted in a stable equilibrium being reached, which is what necessitated our early stopping and learning rate annealing protocol, shown in Algorithm 3.1. Even then, stability was difficult to achieve, as our remarks thus far have all been about training DANN with this protocol in place.

There are several reasons why DANN may be so difficult to train. The first is that DANN trains on both the prediction task and domain discrimination task at the same time. This confounds each task, and there are obvious trade-offs, especially given that adversarial training by nature is unstable and thus can be knocked out of a metastable equilibrium easily. This may also explain CellDART's poor performance in the level 2 domain variance metrics RF50 and miLISI (see Table 4.2 and Table 4.3 in Section 4.2.2), where it failed to produce latent representations that corrected for differences in distribution between source and target. As CellDART trains both tasks in alternating iterations, it would also suffer from this problem somewhat as well, although it does not train both objectives at any given time. We also note that DANN generally performed better

than CellDART in RF50 and miLISI; this may also be due to our rule that only hyperparameter configurations where our stability criterion was met could be selected as part of the random search.

A second reason why DANN has difficulty achieving an equilibrium during adversarial training is its use of a GRL to put the objectives of the discriminator and encoder in opposition to one another. As the authors of ADDA note, as the discriminator converges, the gradient as passed through the gradient reversal layer vanishes [29]. This means that the more the discriminator learns and as it performs better on the domain discrimination task, the smaller the weight updates of the encoder for the opposing task become. This may explain the large oscillations in Fig. 4.4: as the discriminator begins to converge in a certain direction in loss space following the gradient, it is difficult for the encoder to stop that convergence. This would also explain the modal collapse we observed in Fig. 4.5. In contrast, by training the encoder and discriminator in separate steps and using inverted labels, stronger gradients are passed to the encoder as accuracy in the discrimination task increases, which is why this technique is used in GANs [73] and ADDA. In light of this, we suggest that the label inverting approached used in CellDART [27] and ADDA are better suited for adversarial training tasks.

5.1.2 Performance of ADDA wholly a result of initialization of target encoder

ADDA was unique amongst our methods tested in that the subnetworks trained during pretraining to label source domain data, the source encoder and predictor, were independent of the subnetworks trained during the adversarial phase, the target encoder, and the discriminator. Given that the source and target encoder do not share weights, this, as stated by the authors of ADDA [29] and supported by our own observations, means that ADDA's performance is highly dependent on how the target encoder is initialized, which is using the final pretraining weights of the source encoder. This is why we limited the number of adversarial epochs to 20. We also observe, as shown

in Fig. 4.4, that the performance of ADDA on the real spot cell type deconvolution task settles to an equilibrium within a single epoch, and that the performance on the training and validation sets actually decrease over the pretrained model (shown as epoch "-1").

A crucial observation when examining the implementation of ADDA is that the target encoder never has as an objective to specifically present a "correct" embedding for any given sample. Its only objective, across all stages of training, is to present to the discriminator embeddings from each of the source and target datasets that the discriminator is unable to discern from which dataset it came from. Thus, it does not matter to either the discriminator or the encoder whether the embedding presented to the discriminator represents the same cell type as was given as input to the encoder. The only way to enforce true mapping of embeddings is if samples from each of the source and target domain were matched. For these reasons, and our own observations as well, we theorize that the only reason that ADDA can perform any cell type deconvolution with any degree of accuracy at all is purely due to the initialization of the weights of the encoder.

5.1.3 Remark on domain discrimination performance

A proposition that we provided as justification for including the three levels of metrics, as defined in Section 3.3, was that a combination of different methods of evaluation could be used as a substitute for using datasets with hard ground truth labels in the target domain. We surmised that if a method performed well in pseudo-spots and demonstrated well matched latent mappings from each of the source and target domains, that a conclusion could be made that it would also perform well on target domain data. Our results in Section 4.1 do not bear this hypothesis out. While ADDA performed the best across all three datasets in the source domain, as shown in Table 4.1, and demonstrated relatively good performance across all three datasets in RF50 and miLISI, as shown in Table 4.2 and Table 4.3, the only dataset where its real target spot performance was good was

on the Spotless mouse cortex set. Furthermore, while CellDART showed middling performance in the level 1 pseudo-spots metrics and failure in the level 2 domain discrepancy metrics, CellDART was one of the better performers in level 3 on the dlPFC dataset and the best on the PDAC dataset. We must note, however, that we used ADDA's source encoder for source data; since we used the target encoder for the inference task on the real spatial transcriptomics spots, the results in level 1 metrics may not truly reflect the learning that would be transferred to the spatial transcriptomics domain. Additionally, we must also note that, as previously discussed in this section, that the spatial transcriptomics metric for the PDAC dataset may have been faulty, and thus the performance of CellDART in this must be met with a degree of skepticism.

5.2 Future work

The results we obtained, while not definitive, raised some interesting questions. One might speculate that RCTD did not perform well on the dlPFC set because the task involved identifying the layers in which excitatory neurons belonged; it performed otherwise the best on the Spotless "gold standard 1" set, and thus one possible explanation is that RCTD can broadly model cell types but is unable to distinguish between specific sub types and/or cell states; further analysis may be needed to investigate whether or not this is the case. Additionally, a potential reason for why CORAL performed poorly on the Spotless mouse cortex dataset was the small batch sizes due to overall small size of the dataset causing effects on calculated feature covariances. While this led to CORAL performing the best in the level 2 domain invariance metric for that dataset, we also noted in Section 4.3 that the weighting parameter λ selected in our hyperparameter search was the lowest among all models. We could investigate the effect of different batch sizes when using CORAL on the dlPFC set, which was our largest.

In terms of possible future approaches, in light of CellDART remaining a good performer, and as we remarked earlier in this section that the drawback of ADDA and DANN respectively were ADDA's lack of source domain training during its adversarial phase and DANN's gradient reversal layer, CellDART's approach of using ADDA's method of inverting labels for the adversarial task while using DANN's approach of training both the prediction and domain discrimination tasks in the same phase appears to be an optimal solution. Given that CellDART's primary issue was its inability to reduce domain mismatch, we believe that the most immediate avenue of work involves using methods to better improve its adversarial methods, such as by using the stability protocol we used with DANN, shown in Algorithm 3.1.

Another interesting avenue to explore is addressing ADDA's domain mismatch issue. Addressing the sample matching issue would help the target encoder produce better true embeddings for target samples. One approach to do this would be the use of a variety of cycle consistency loss consistent loss using cycle-consistent adversarial networks [111]. The authors of ADDA later presented cycle-consistent adversarial domain adaptation (CyCADA), which uses a pair of adversarial models in tandem with cycle-consistency loss, to ensure that individual images being mapped from one domain to another could be mapped back using a complementary model [112]. A similar approach could also be used for our implementation of ADDA by applying the same method to embeddings. As this approach would still allow for separate source and target encoders and would not rely on initialization, we could also tailor each encoder so for each modality, for example by using a GCN to for the spatial data to incorporate spatial information.

We finally include the caveat that, as we used validation ground truths to perform hyperparameter tuning, our results are only valid in circumstances where there exists already at least 1 annotated slide to validate against. Otherwise, we suggest end users seeking to use our

methods to use the tuned hyperparameters we report in Section 4.3. Given current standard practice in UDA is to use some amount of "peeking" at the target labels to fine-tune models, as summarized in a review [113], and CellDART in fact did not use any holdout samples, using the same spatial transcriptomics slides to train, validate and tune hyperparameters, and evaluate their model, our conclusions are at least as robust as the current literature. That being said, the usefulness of UDA depends on its ability to perform well where no labels exist, and so a number of truly unsupervised methods of validation have been proposed, such as reverse validation, which involves using the trained UDA model to first label target domain samples, then training a second model on those samples as the new source domain and validating using source labels [114]; notably, this method was used to evaluate DANN when first presented by its authors [30]. We thus view leveraging these methods as a top priority for any future work so that we will be able to obtain truly robust results that are usable in many circumstances.

6. Conclusion and Summary

In this work, we thoroughly compared and assessed the performance and behaviour of 3 deep UDA methods, being ADDA [29], DANN [30], and Deep CORAL [31], along with a task-specific UDA baseline, CellDART [27], and a Bayesian task-specific baseline, RCTD [32], of cell type deconvolution for spatial transcriptomics spots. Although we were unable to conclusively demonstrate that any of the UDA methods we compared (CellDART, ADDA, DANN, or CORAL), or indeed even RCTD, performed better or worse when applied to deconvolving target domain spots, we were able to gain insights that open the door for further investigation. We firstly found that DANN was difficult to train in a stable manner. We secondly found that while ADDA performed the best in the source scRNA-seq prediction task across all datasets, its performance on the target domain deconvolution task was wholly dependent on pre-initialization from a pretraining phase on the source domain. Finally, we noted that performance in inter-domain variance metrics did not necessarily translate in terms of performance from scRNA-seq source domain data to real target domain data. We finally laid the groundwork for future directions of research, including improved ADDA models to address the dependency on initialization, the including of GCNs to incorporate spatial information, and using truly unsupervised methods of hyperparameter tuning such as reverse validation [114].

References

[1] R. J. Britten and E. H. Davidson, "Gene regulation for higher cells: A theory," *Science,* vol. 165, no. 3891, pp. 349-357, 1969, doi: 10.1126/science.165.3891.349.

[2] J. Guo, "Transcription: the epicenter of gene expression," *J. Zhejiang Univ. Sci. B,* vol. 15, no. 5, pp. 409-11, May 2014, doi: 10.1631/jzus.B1400113.

[3] F. Ginhoux, A. Yalin, C. A. Dutertre, and I. Amit, "Single-cell immunology: Past, present, and future," *Immunity,* vol. 55, no. 3, pp. 393-404, 2022, doi: 10.1016/j.immuni.2022.02.006.

[4] H. M. Levitin, J. Yuan, and P. A. Sims, "Single-cell transcriptomic analysis of tumor heterogeneity," *Trends Cancer,* vol. 4, no. 4, pp. 264-268, Apr 2018, doi: 10.1016/j.trecan.2018.02.003.

[5] J. Ding *et al.*, "Integrating multiomics longitudinal data to reconstruct networks underlying lung development," *Am. J. Physiol. Lung Cell Mol. Physiol.,* vol. 317, no. 5, pp. L556-L568, 2019, doi: 10.1152/ajplung.00554.2018.

[6] H.-E. Park *et al.*, "Spatial transcriptomics: Technical aspects of recent developments and their applications in neuroscience and cancer research," *Adv. Sci.,* p. 2206939, 2023, doi: 10.1002/advs.202206939.

[7] F. Cremisi, A. Philpott, and S.-i. Ohnuma, "Cell cycle and cell fate interactions in neural development," *Curr. Opin. Neurobiol.,* vol. 13, no. 1, pp. 26-33, 2003, doi: 10.1016/S0959-4388(03)00005-9.

[8] R. Dries, J. Chen, N. Del Rossi, M. M. Khan, A. Sistig, and G.-C. Yuan, "Advances in spatial transcriptomic data analysis," *Genome Res.,* vol. 31, no. 10, pp. 1706-1718, 2021, doi: 10.1101/gr.275224.12.

[9] A. Emad and S. Sinha, "Inference of phenotype-relevant transcriptional regulatory networks elucidates cancer type-specific regulatory mechanisms in a pan-cancer study," *NPJ Syst. Biol. Appl.,* vol. 7, no. 1, pp. 1-14, 2021, doi: 10.1038/s41540-021-00169-7.

[10] K. Hurley *et al.*, "Reconstructed single-cell fate trajectories define lineage plasticity windows during differentiation of human PSC-derived distal lung progenitors," *Cell Stem Cell,* vol. 26, no. 4, pp. 593-608.e8, 2020, doi: j.stem.2019.12.009.

[11] C. Su, S. Rousseau, and A. Emad, "Identification of transcriptional regulatory network associated with response of host epithelial cells to SARS-CoV-2," *Sci. Rep.,* vol. 11, no. 1, pp. 1-12, 2021, doi: 10.1038/s41598-021-03309-5.

[12] A. D. Lander, "How cells know where they are," *Science,* vol. 339, no. 6122, pp. 923-927, 2013, doi: 10.1126/science.1224186.

[13] P. Rompolas, K. R. Mesa, and V. Greco, "Spatial organization within a niche as a determinant of stem-cell fate," *Nature,* vol. 502, no. 7472, pp. 513-518, Oct. 2013, doi: 10.1038/nature12602.

[14] S. K. Longo, M. G. Guo, A. L. Ji, and P. A. Khavari, "Integrating single-cell and spatial transcriptomics to elucidate intercellular tissue dynamics," *Nat. Rev. Genet.,* vol. 22, no. 10, pp. 627-644, Oct. 2021, doi: 10.1038/s41576-021-00370-8.

[15] A. E. Moor and S. Itzkovitz, "Spatial transcriptomics: Paving the way for tissue-level systems biology," *Curr. Opin. Biotechnol.,* vol. 46, pp. 126-133, 2017, doi: 10.1016/j.copbio.2017.02.004.

[16] Z. A. Rasheed, W. Matsui, and A. Maitra, "Pathology of pancreatic stroma in PDAC," in *Pancreatic Cancer and Tumor Microenvironment*, P. J. Grippo and H. G. Munshi Eds. Trivandrum (India): Transworld Research Network, 2012.

[17] M. Korc, "Pancreatic cancer-associated stroma production," *Am. J. Surg.,* vol. 194, no. 4 Suppl, pp. S84-6, Oct 2007, doi: 10.1016/j.amjsurg.2007.05.004.

[18] D. von Ahrens, T. D. Bhagat, D. Nagrath, A. Maitra, and A. Verma, "The role of stromal cancer-associated fibroblasts in pancreatic cancer," *J. Hematol. Oncol.,* vol. 10, no. 1, p. 76, 2017, doi: 10.1186/s13045-017-0448-5.

[19] C. Murdoch, M. Muthana, S. B. Coffelt, and C. E. Lewis, "The role of myeloid cells in the promotion of tumour angiogenesis," *Nat. Rev. Cancer,* vol. 8, no. 8, pp. 618-631, 2008, doi: 10.1038/nrc2444.

[20] S. P. Kerkar and N. P. Restifo, "Cellular constituents of immune escape within the tumor microenvironment," *Cancer Res.,* vol. 72, no. 13, pp. 3125-3130, 2012, doi: 10.1158/0008-5472.Can-11-4094.

[21] K. H. Chen, A. N. Boettiger, J. R. Moffitt, S. Wang, and X. Zhuang, "Spatially resolved, highly multiplexed RNA profiling in single cells," *Science,* vol. 348, no. 6233, p. aaa6090, 2015, doi: 10.1126/science.aaa6090.

[22] E. Lubeck, A. F. Coskun, T. Zhiyentayev, M. Ahmad, and L. Cai, "Single-cell in situ RNA profiling by sequential hybridization," *Nat. Methods,* vol. 11, no. 4, pp. 360-361, 2014, doi: 10.1038/nmeth.2892.

[23] L. Larsson, J. Frisén, and J. Lundeberg, "Spatially resolved transcriptomics adds a new dimension to genomics," *Nat. Methods,* vol. 18, no. 1, pp. 15-18, 2021, doi: 10.1038/s41592-020-01038-7.

[24] A. Chen *et al.*, "Spatiotemporal transcriptomic atlas of mouse organogenesis using DNA nanoball-patterned arrays," *Cell,* vol. 185, no. 10, pp. 1777-1792. e21, 2022, doi: 10.1016/j.cell.2022.04.003.

[25] R. D. Hodge *et al.*, "Conserved cell types with divergent features in human versus mouse cortex," *Nature,* vol. 573, no. 7772, pp. 61-68, 2019, doi: 10.1038/s41586-019-1506-7.

[26] R. Moncada *et al.*, "Integrating microarray-based spatial transcriptomics and single-cell RNA-seq reveals tissue architecture in pancreatic ductal adenocarcinomas," *Nat. Biotechnol.,* vol. 38, no. 3, pp. 333-342, 2020, doi: 10.1038/s41587-019-0392-8.

[27] S. Bae, K. J. Na, J. Koh, D. S. Lee, H. Choi, and Young T. Kim, "CellDART: Cell type inference by domain adaptation of single-cell and spatial transcriptomic data," *Nucleic Acids Res.,* vol. 50, no. 10, pp. e57-e57, 2022, doi: 10.1093/nar/gkac084.

[28] P. Singhal, R. Walambe, S. Ramanna, and K. Kotecha, "Domain adaptation: Challenges, methods, datasets, and applications," *IEEE Access,* vol. 11, pp. 6973-7020, 2023, doi: 10.1109/ACCESS.2023.3237025.

[29] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 7167-7176.

[30] Y. Ganin *et al.*, "Domain-adversarial training of neural networks," in *Domain Adaptation in Computer Vision Applications*, G. Csurka Ed. Cham: Springer International Publishing, 2017, pp. 189-209.

[31] B. Sun and K. Saenko, "Deep CORAL: Correlation alignment for deep domain adaptation," in *Comput. Vis. - ECCV 2016*, Amsterdam, The Netherlands, October 8-10 and 15-16 2016, vol. 14, no. 3: Springer, pp. 443-450.

[32] D. M. Cable *et al.*, "Robust decomposition of cell type mixtures in spatial transcriptomics," *Nat. Biotechnol.,* vol. 40, no. 4, pp. 517-526, Apr. 2022, doi: 10.1038/s41587-021-00830-w.

[33] C. Sang-aram, R. Browaeys, R. Seurinck, and Y. Saeys, "Spotless: a reproducible pipeline for benchmarking cell type deconvolution in spatial transcriptomics," *bioRxiv,* p. 2023.03.22.533802, 2023, doi: 10.1101/2023.03.22.533802.

[34] F. Crick, "Central dogma of molecular biology," *Nature,* vol. 227, no. 5258, pp. 561-563, Aug. 1970, doi: 10.1038/227561a0.

[35] S. L. Salzberg, "Open questions: How many genes do we have?," *BMC Biol.,* vol. 16, no. 1, p. 94, Aug. 20 2018, doi: 10.1186/s12915-018-0564-x.

[36] S. Nurk *et al.*, "The complete sequence of a human genome," *Science,* vol. 376, no. 6588, pp. 44-53, Apr 2022, doi: 10.1126/science.abj6987.

[37] J. C. Venter *et al.*, "The sequence of the human genome," *Science,* vol. 291, no. 5507, pp. 1304-1351, 2001, doi: 10.1126/science.1058040.

[38] Richard A. Young, "Control of the embryonic stem cell state," *Cell,* vol. 144, no. 6, pp. 940-954, 2011, doi: 10.1016/j.cell.2011.01.032.

[39] X. Han *et al.*, "Mapping the mouse cell atlas by Microwell-seq," *Cell,* vol. 172, no. 5, pp. 1091-1107.e17, 2018, doi: 10.1016/j.cell.2018.02.001.

[40] Tong I. Lee and Richard A. Young, "Transcriptional regulation and its misregulation in disease," *Cell,* vol. 152, no. 6, pp. 1237-1251, 2013, doi: 10.1016/j.cell.2013.02.014.

[41] E. Fuchs, T. Tumbar, and G. Guasch, "Socializing with the neighbors: Stem cells and their niche," *Cell,* vol. 116, no. 6, pp. 769-778, 2004, doi: 10.1016/S0092-8674(04)00255-7.

[42] K. A. Miles, "Tumour angiogenesis and its relation to contrast enhancement on computed tomography: a review," *Eur. J. Radiol.,* vol. 30, no. 3, pp. 198-205, 1999, doi: 10.1016/S0720-048X(99)00012-1.

[43] Z. Dong and Y. Chen, "Transcriptomics: Advances and approaches," *Sci. China Life Sci.,* vol. 56, no. 10, pp. 960-967, Oct. 2013, doi: 10.1007/s11427-013-4557-2.

[44] P. A. McGettigan, "Transcriptomics in the RNA-seq era," *Curr. Opin. Chem. Biol.,* vol. 17, no. 1, pp. 4-11, 2013, doi: 10.1016/j.cbpa.2012.12.008.

[45] V. Y. Kiselev, T. S. Andrews, and M. Hemberg, "Challenges in unsupervised clustering of single-cell RNA-seq data," *Nat. Rev. Genet.,* vol. 20, no. 5, pp. 273-282, May 2019, doi: 10.1038/s41576-018-0088-9.

[46] H. Li *et al.*, "Dysfunctional CD8 T cells form a proliferative, dynamically regulated compartment within human melanoma," *Cell,* vol. 176, no. 4, pp. 775-789.e18, 2019, doi: 10.1016/j.cell.2018.11.043.

[47] J. A. Pai and A. T. Satpathy, "High-throughput and single-cell T cell receptor sequencing technologies," *Nat. Methods,* vol. 18, no. 8, pp. 881-892, 2021, doi: 10.1038/s41592-021-01201-8.

[48] E. Armingol, A. Officer, O. Harismendy, and N. E. Lewis, "Deciphering cell–cell interactions and communication from gene expression," *Nat. Rev. Genet.,* vol. 22, no. 2, pp. 71-88, 2021, doi: 10.1038/s41576-020-00292-x.

[49] M. Asp, J. Bergenstråhle, and J. Lundeberg, "Spatially resolved transcriptomes—Next generation tools for tissue exploration," *Bioessays,* vol. 42, no. 10, p. 1900221, 2020, doi: 10.1002/bies.201900221.

[50] P. L. Ståhl *et al.*, "Visualization and analysis of gene expression in tissue sections by spatial transcriptomics," *Science,* vol. 353, no. 6294, pp. 78-82, 2016, doi: 10.1126/science.aaf2403.

[51] H. A. Alturkistani, F. M. Tashkandi, and Z. M. Mohammedsaleh, "Histological stains: A literature review and case study," *Glob. J. Health Sci.,* vol. 8, no. 3, pp. 72-9, Jun. 25 2015, doi: 10.5539/gjhs.v8n3p72.

[52] X. Wang *et al.*, "Three-dimensional intact-tissue sequencing of single-cell transcriptional states," *Science,* vol. 361, no. 6400, p. eaat5691, 2018, doi: 10.1126/science.aat5691.

[53] C. Larsson, I. Grundberg, O. Söderberg, and M. Nilsson, "In situ detection and genotyping of individual mRNA molecules," *Nat. Methods,* vol. 7, no. 5, pp. 395-397, May 2010, doi: 10.1038/nmeth.1448.

[54] M. Pavličev *et al.*, "Single-cell transcriptomics of the human placenta: inferring the cell communication network of the maternal-fetal interface," *Genome Res.,* vol. 27, no. 3, pp. 349-361, 2017, doi: 10.1101/gr.207597.116.

[55] D. E. Cohen and D. Melton, "Turning straw into gold: Directing cell fate for regenerative medicine," *Nat. Rev. Genet.,* vol. 12, no. 4, pp. 243-252, Apr. 2011, doi: 10.1038/nrg2938.

[56] R. M. Bremnes *et al.*, "The role of tumor stroma in cancer progression and prognosis: Emphasis on carcinoma-associated fibroblasts and non-small cell lung cancer," *J. Thorac. Oncol.,* vol. 6, no. 1, pp. 209-217, 2011, doi: 10.1097/JTO.0b013e3181f8a1bd.

[57] L. H. Truong and S. Pauklin, "Pancreatic cancer microenvironment and cellular composition: Current understandings and therapeutic approaches," *Cancers (Basel),* vol. 13, no. 19, Oct 8 2021, doi: 10.3390/cancers13195028.

[58] S. C. Hicks, F. W. Townes, M. Teng, and R. A. Irizarry, "Missing data and technical variability in single-cell RNA-sequencing experiments," *Biostatistics,* vol. 19, no. 4, pp. 562-578, 2017, doi: 10.1093/biostatistics/kxx053.

[59] J. T. Leek *et al.*, "Tackling the widespread and critical impact of batch effects in high-throughput data," *Nat. Rev. Genet.,* vol. 11, no. 10, pp. 733-739, Oct. 2010, doi: 10.1038/nrg2825.

[60] G. Chen, B. Ning, and T. Shi, "Single-cell RNA-seq technologies and related computational data analysis," *Front. Genet.,* vol. 10, p. 317, 2019, doi: 10.3389/fgene.2019.00317.

[61] H. T. N. Tran *et al.*, "A benchmark of batch-effect correction methods for single-cell RNA sequencing data," *Genome Biol.,* vol. 21, no. 1, p. 12, Jan. 16 2020, doi: 10.1186/s13059-019-1850-9.

[62] X. Zhou, H. Chai, Y. Zeng, H. Zhao, and Y. Yang, "scAdapt: Virtual adversarial domain adaptation network for single cell RNA-seq data classification across platforms and species," *Brief. Bioinform.,* vol. 22, no. 6, 2021, doi: 10.1093/bib/bbab281.

[63] B. Tasic *et al.*, "Shared and distinct transcriptomic cell types across neocortical areas," *Nature,* vol. 563, no. 7729, pp. 72-78, Nov. 2018, doi: 10.1038/s41586-018-0654-5.

[64] 10x Genomics. Datasets [Online] Available: https://www.10xgenomics.com/resources/datasets

[65] F. A. Wolf, P. Angerer, and F. J. Theis, "SCANPY: Large-scale single-cell gene expression data analysis," *Genome Biol.,* vol. 19, no. 1, p. 15, Feb. 6 2018, doi: 10.1186/s13059-017-1382-0.

[66] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform manifold approximation and projection for dimension reduction," *arXiv,* p. arXiv:1802.03426v3, 2018, doi: 10.48550/arXiv.1802.03426.

[67] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.,* vol. 12, no. 85, pp. 2825-2830, 2011. [Online]. Available: http://jmlr.org/papers/v12/pedregosa11a.html.

[68] J. H. Lee *et al.*, "Highly multiplexed subcellular RNA sequencing in situ," *Science,* vol. 343, no. 6177, pp. 1360-1363, 2014, doi: 10.1126/science.1250212.

[69] I. Korsunsky *et al.*, "Fast, sensitive and accurate integration of single-cell data with Harmony," *Nat. Methods,* vol. 16, no. 12, pp. 1289-1296, Dec. 2019, doi: 10.1038/s41592-019-0619-0.

[70] Q. Song and J. Su, "DSTG: Deconvoluting spatial transcriptomics data through graph-based artificial intelligence," *Brief. Bioinform.,* vol. 22, no. 5, p. bbaa414, 2021, doi: 10.1093/bib/bbaa414.

[71] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (Adaptive Computation and Machine Learning). Cambridge, MA: MIT Press, 2016.

[72] B. Sun, J. Feng, and K. Saenko, "Return of frustratingly easy domain adaptation," in *Proc. AAAI Conf. Artif. Intell.*, Phoenix, AZ, Mar. 2 2016, vol. 30, no. 1: AAAI Press, doi: 10.1609/aaai.v30i1.10306.

[73] I. Goodfellow *et al.*, "Generative adversarial nets," in *Adv. Neural Inf. Process. Syst.*, Montreal, Canada, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, Eds., Dec. 8-13 2014, vol. 27, no. 1: Curran Associates, Inc., p. 2672.

[74] C. Nagy *et al.*, "Single-nucleus transcriptomics of the prefrontal cortex in major depressive disorder implicates oligodendrocyte precursor cells and excitatory neurons," *Nat. Neurosci.,* vol. 23, no. 6, pp. 771-781, June 2020, doi: 10.1038/s41593-020-0621-y.

[75] Y. Ma and X. Zhou, "Spatially informed cell-type deconvolution for spatial transcriptomics," *Nat. Biotechnol.,* vol. 40, no. 9, pp. 1349-1359, 2022, doi: 10.1038/s41587-022-01273-7.

[76] B. Li *et al.*, "Benchmarking spatial and single-cell transcriptomics integration methods for transcript distribution prediction and cell type deconvolution," *Nat. Methods,* vol. 19, no. 6, pp. 662-670, June 2022, doi: 10.1038/s41592-022-01480-9.

[77] M. Elosua-Bayes, P. Nieto, E. Mereu, I. Gut, and H. Heyn, "SPOTlight: Seeded NMF regression to deconvolute spatial transcriptomics spots with single-cell transcriptomes," *Nucleic Acids Res.,* vol. 49, no. 9, pp. e50-e50, 2021, doi: 10.1093/nar/gkab043.

[78] I. Virshup, S. Rybakov, F. J. Theis, P. Angerer, and F. A. Wolf, "anndata: Annotated data," *bioRxiv,* p. 2021.12.16.473007, 2021, doi: 10.1101/2021.12.16.473007.

[79] L. C. Greig, M. B. Woodworth, M. J. Galazo, H. Padmanabhan, and J. D. Macklis, "Molecular logic of neocortical projection neuron specification, development and diversity," *Nat. Rev. Neurosci.,* vol. 14, no. 11, pp. 755-769, Nov. 2013, doi: 10.1038/nrn3586.

[80] K. R. Maynard *et al.*, "Transcriptome-scale spatial gene expression in the human dorsolateral prefrontal cortex," *Nat. Neurosci.,* vol. 24, no. 3, pp. 425-436, Mar. 2021, doi: 10.1038/s41593-020-00787-0.

[81] B. Pardo *et al.*, "spatialLIBD: an R/Bioconductor package to visualize spatially-resolved transcriptomics data," *BMC Genomics,* vol. 23, no. 1, p. 434, June 10 2022, doi: 10.1186/s12864-022-08601-w.

[82] Z. Zhou, Y. Zhong, Z. Zhang, and X. Ren, "Spatial transcriptomics deconvolution at single-cell resolution by Redeconve," *bioRxiv,* p. 2022.12.22.521551, 2022, doi: 10.1101/2022.12.22.521551.

[83] J. Peng *et al.*, "Single-cell RNA-seq highlights intra-tumoral heterogeneity and malignant progression in pancreatic ductal adenocarcinoma," *Cell Res.,* vol. 29, no. 9, pp. 725-738, Sep. 2019, doi: 10.1038/s41422-019-0195-y.

[84] R. Chijimatsu *et al.*, "Establishment of a reference single-cell RNA sequencing dataset for human pancreatic adenocarcinoma," *iScience,* vol. 25, no. 8, 2022, doi: 10.1016/j.isci.2022.104659.

[85] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *Ann. Math. Stat.,* vol. 18, no. 1, pp. 50-60, 1947, doi: 10.1214/aoms/1177730491.

[86] Y. Benjamini and Y. Hochberg, "Controlling the false discovery rate: A practical and powerful approach to multiple testing," *J. R. Stat. Soc. B: Methodol.,* vol. 57, no. 1, pp. 289-300, 1995, doi: 10.1111/j.2517-6161.1995.tb02031.x.

[87] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv,* p. arXiv:1207.0580, 2012, doi: 10.48550/arXiv.1207.0580.

[88] D. I. Warton, "Why you cannot transform your way out of trouble for small counts," *Biometrics,* vol. 74, no. 1, pp. 362-368, 2018, doi: 10.1111/biom.12728.

[89] A. Gelman. "You should (usually) log transform your positive data." https://statmodeling.stat.columbia.edu/2019/08/21/you-should-usually-log-transform-your-positive-data/ (accessed Aug. 3, 2023).

[90] E. H. Simpson, "Measurement of diversity," *Nature,* vol. 163, no. 4148, pp. 688-688, Apr. 1949, doi: 10.1038/163688a0.

[91] A. Chao *et al.*, "Rarefaction and extrapolation with Hill numbers: A framework for sampling and estimation in species diversity studies," *Ecol. Monogr.,* vol. 84, no. 1, pp. 45-67, 2014, doi: 10.1890/13-0133.1.

[92] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.,* vol. 9, no. 11, pp. 2579-2605, 2008. [Online]. Available: http://jmlr.org/papers/v9/vandermaaten08a.html.

[93] M. O. Hill, "Diversity and evenness: A unifying notation and its consequences," *Ecology,* vol. 54, no. 2, pp. 427-432, 1973, doi: 10.2307/1934352.

[94] C. E. Shannon, "A mathematical theory of communication," *Bell System Tech. J.,* vol. 27, no. 3, pp. 379-423, 1948, doi: 10.1002/j.1538-7305.1948.tb01338.x.

[95] K. Slowikowski. "harmonypy." https://github.com/slowkow/harmonypy (accessed July 15, 2023).

[96] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," *J. Mach. Learn. Res.,* vol. 18, no. 17, pp. 1-5, 2017. [Online]. Available: http://jmlr.org/papers/v18/16-365.html.

[97] M. L. Minsky and S. A. Papert, *Perceptrons: An Introduction to Computational Geometry*, expanded ed. MIT Press, 1988.

[98] P. S. Leung and S. P. Ip, "Pancreatic acinar cell: Its role in acute pancreatitis," *Int. J. Biochem. Cell Biol.,* vol. 38, no. 7, pp. 1024-1030, Jan. 2006, doi: 10.1016/j.biocel.2005.12.001.

[99] M. B. Omary, A. Lugea, A. W. Lowe, and S. J. Pandol, "The pancreatic stellate cell: A star on the rise in pancreatic diseases," *J. Clin. Invest.,* vol. 117, no. 1, pp. 50-59, Jan. 2 2007, doi: 10.1172/JCI30082.

[100] R. Jiang, T. Sun, D. Song, and J. J. Li, "Statistics or biology: The zero-inflation controversy about scRNA-seq data," *Genome Biol.,* vol. 23, no. 1, p. 31, 2022/01/21 2022, doi: 10.1186/s13059-022-02601-5.

[101] T. H. Kim, X. Zhou, and M. Chen, "Demystifying “drop-outs” in single-cell UMI data," *Genome Biol.,* vol. 21, no. 1, p. 196, 2020/08/06 2020, doi: 10.1186/s13059-020-02096-y.

[102] A. Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Adv. Neural Inf. Process. Syst.*, 2019, vol. 32, pp. 8024-8035.

[103] S. Kullback and R. A. Leibler, "On information and sufficiency," *Ann. Math. Stat.,* vol. 22, no. 1, pp. 79-86, Mar. 1951, doi: 10.1214/aoms/1177729694.

[104] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv,* p. arXiv:1711.05101, 2017, doi: 10.48550/arXiv.1711.05101.

[105] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv,* p. arXiv:1412.6980, 2014, doi: 10.48550/arXiv.1412.6980.

[106] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," presented at the ICML Workshop on Deep Learning for Audio, Speech, and Language Processing, Atlanta, GA, June 16, 2013. [Online]. Available: https://ai.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf.

[107] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv,* p. arXiv:1511.06434, 2016, doi: 10.48550/arXiv.1511.06434.

[108] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Mach. Learn.*, Lille, France, F. Bach and D. Blei, Eds., Jul. 7-9 2015, vol. 37: PMLR, pp. 448-456. [Online]. Available: https://proceedings.mlr.press/v37/ioffe15.html.

[109] F. Chollet *et al.*, *Keras*. (2015). [Online]. Available: https://keras.io

[110] L. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," in *Artif. Intell. Mach. Learn. Multi-Domain Operations Appl.*, Bellingham, WA, T. Pham, Ed., Apr. 14-18 2019, vol. 11006: SPIE, in Proceedings of SPIE, p. 36, doi: 10.1117/12.2520589.

[111] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," *arXiv,* p. arXiv:1703.10593, 2020, doi: 10.48550/arXiv.1703.10593.

[112] J. Hoffman *et al.*, "CyCADA: Cycle-consistent adversarial domain adaptation," in *Proc. 32nd Int. Conf. Mach. Learn.*, Stockholm, Sweden, J. Dy and A. Krause, Eds., July 10-15 2018, vol. 80: PMLR, in Proceedings of Machine Learning Research, pp. 1989-1998. [Online]. Available: https://proceedings.mlr.press/v80/hoffman18a.html.

[113] K. Musgrave, S. Belongie, and S.-N. Lim, "Unsupervised domain adaptation: A reality check," *arXiv,* p. arXiv:2111.15672, 2021, doi: 10.48550/arXiv.2111.15672.

[114] E. Zhong, W. Fan, Q. Yang, O. Verscheure, and J. Ren, "Cross validation framework to choose amongst models and datasets for transfer learning," in *Mach. Learn. Knowl. Discov. Databases*, Barcelona, Spain, J. L. Balcázar, F. Bonchi, A. Gionis, and M. Sebag, Eds., 2010, vol. 6323, Berlin, Heidelberg: Springer Berlin Heidelberg, in Lecture Notes in Computer Science, 2010, pp. 547-562, doi: 10.1007/978-3-642-15939-8_35.