Optical Music Recognition of Square Notation using Generative Adversarial Networks

Evan Savage

August 2020

Music Technology Area
Department of Music Research
Schulich School of Music
McGill University, Montréal

Abstract

Manually converting Medieval chant manuscripts, written in square notation, to a computer-readable format is costly. Optical Music Recognition (OMR), which automatically performs the conversion from scanned manuscript images, can potentially reduce this cost. Machine learning models can be used to perform OMR, though they often require large amounts of labelled training data. In this work, a generative adversarial network (GAN) is trained to generate images of the individual notes and symbols that make up square notation. A GAN entangles two neural networks in a game, where one is progressively trained to generate increasingly realistic images based on a training set to fool a second network, which iteratively evaluates whether the generated examples appear as real as the training set. The music symbols generated by the GAN are placed on staff lines on synthetic manuscript pages, mimicking the appearance and structure of a real page of square notation. A novel OMR workflow is introduced that includes two sequential machine learning OMR models used in sequence: the first for object detection of musical symbols, the second for determining the vertical staff position of the notes. The baseline OMR workflow experiments are trained with real manuscript images only, and their results are compared against the OMR workflow trained with both real and synthetic manuscript images. Two medieval manuscripts, the Salzinnes Antiphonal and the Einsiedeln Stiftsbibliothek Codex 611(89) are used for the experiments. Comparing against the baseline real data experiments, an increase in the OMR workflow's evaluation metrics demonstrates that the OMR of square notation is improved by training the workflow with both real and synthetic data, assisted by the GAN architecture. Experimental results indicate that the OMR of square notation can be improved by using GAN-synthesized manuscript data.

Résumé

La conversion manuelle de chants médiévaux depuis leur notation neumatique carrée vers un format lisible par les ordinateurs est un processus coûteux. La Reconnaissance Optique de la Musique (ROM ; OMR en anglais) pourrait contribuer à réduire ce coût en convertissant automatiquement les images numérisées de manuscrits. Des modèles d'apprentissage automatique peuvent être utilisés pour améliorer les performances de la ROM, bien qu'ils requièrent normalement un important volume de données annotées. Dans le cadre de ce projet, un Réseau Antagoniste Génératif (RAG ; GAN en anglais) a été entraîné pour générer les images individuelles des notes et des symboles que l'on retrouve dans la notation carrée. Un RAG entremêle deux réseaux de neurones en un jeu d'apprentissage mutuel : tandis qu'un des réseaux est entraîné à générer des images de plus en plus réalistes, le second évalue par itération si celles-ci sont aussi réalistes que celles utilisés pour l'entraînement. Les symboles musicaux générés par le RAG sont placés sur des lignes de portée de pages manuscrites synthétiques qui imitent l'apparence et la structure d'une vraie page de notation carrée. Une nouvelle méthode de ROM est présentée dans ce projet. Elle inclut deux modèles d'apprentissage automatique en séquence, le premier pour la détection de symboles musicaux et le second pour déterminer la position verticale des notes dans la portée. Les expériences basées sur le processus de ROM sont entraînées uniquement avec de réelles images manuscrites, et leurs résultats sont comparés avec celles entraînées avec à la fois avec des données réelles et des données synthétiques. Deux manuscrits médiévaux ont été utilisés pour ce projet, l'antiphonaire de Salzinnes et le codex 611(89) de la Einsiedeln Stiftsbibliothek. En comparaison avec les expériences menées sur données réelles, une augmentation dans les métriques d'évaluation du processus de la ROM de notation carrée démontre que celle-ci peut être améliorée avec l'aide de l'architecture du RAG, en entraînant le processus à la fois avec des données réelles et des données synthétiques. Les résultats expérimentaux révèlent que la ROM de notation carrée peut être améliorée par l'usage de données manuscrites synthétisées par un RAG.

Acknowledgements

Thank you to all of my colleagues in the Music Technology program at McGill, who inspired me to pursue this work. Specifically, within the Distributed Digital Music Archives and Libraries (DDMAL) Laboratory, I would like to thank Gabriel Vigliensoni for his continued mentoring and introduction to the world of Music Information Retrieval (MIR). Thank you for taking interest in my pursuits, repeatedly editing my thesis proposal, and showing me so much of what the music scene in Montréal has to offer. Thank you to Andrew Kam for jogging some ideas about how to envision this work and for always taking such curiosity in my developments. Thank you to Alex Daigle for continued support and introduction to the DDMAL software infrastructure and for translating my abstract into French. Thank you to Juliette Regimbal for helping me get acclimated with the Neon annotation software and for being open to hearing about my user experience with it.

An enormous thank you is owed to my supervisor, Ichiro Fujinaga, whose guidance and support culminated in the production of this thesis. Thank you for your continued enrichment of my understanding of MIR, deep learning, and computer vision topics and for keeping in constant communication during these final, more difficult months during the pandemic. I have learned so much during the past year through your advising, and I look forward to keeping in touch in the future.

Thank you to Mathias Bredholt, Mathias Kirkegaard, and Liam Welch for being wonderful friends and roommates over the past two years. Your compositions, projects, and conversations inspired me to keep on track with this work, and are motivating me to continue to further explore my own software and composition pursuits in the future.

Another immense thank you is owed to my partner, Nora Mulvey, for the multidimensional support, validation, and comfort you have provided throughout the course of this thesis. Thank you for your unwavering curiosity and check-ins that helped me better communicate my ideas and put them into writing, and for always being there when the more grueling timelines were getting the best of me. Another celebratory long bike ride, picnic, and pandemic-permitting rave is on the other side of this horizon.

Thank you to Nora's mother, Tina Grant, for being a wonderful aid during the editing process of this thesis. Thank you for generously offering your valuable time on such a short

schedule; your expertise helped to polish off my explanations and reasoning, leading to clearer observations throughout.

Finally, thank you to my parents for supporting my decision to move up north and pursue this degree. It has been a tough two years, and you have always made your time available to talk things out and console me whenever the going got tough. Here I am on my way to a music degree, though I have yet to perform a killer drum fill that I have been wanting to nail for more than twenty years. Thank you for always supporting my dynamic endeavors.

Contents

List of Figures

List of Tables

Acronyms

Capitan A corpus collected by an electronic pen while tracing isolated music symbols from Early manuscripts. 28

Cedar A general-purpose node on Compute Canada. 40

CIFAR Canadian Institute For Advanced Research. 18, 19

CNN Convolutional Neural Network. 10, 13, 24, 26–28

CRNN Convolutional Recurrent Neural Network. 28

CVC-MUSCIMA A database of handwritten music score images for writer identification and staff removal. 25

CWMN Common Western Music Notation. 25, 27, 28, 66

D Discriminator Model. 1, 12

DCGAN Deep Convolutional Generative Adversarial Network. 14, 39

DeepScores Dataset of high-quality musical scores. 28

dSAR diplomatic Symbol Accuracy Rate. 22

F_1-Score A metric combining the precision and recall values of an object detection task. 51, 56–59, 61, 63, 64

FCN Fully Convolutional Network. 22

FFHQ Flickr-Faces-HQ dataset. 39

FID Fréchet Inception Distance. 51, 54, 55

G Generator Model. 1, 12, 14

GAN Generative Adversarial Network. 1, 2, 4, 12–16, 18, 20, 23, 29–31, 34, 35, 37–41, 51–56, 59–61, 63–66

GPU Graphics Processing Unit. 39, 40

Inception Score A measure for evaluating the accuracy of images generated by a Generative Adversarial Network. 51

IoU Intersection over Union. 46, 47, 50

LoGAN Logo-centric Generative Adversarial Network. 13, 17

LSGAN Least-Square Generative Adversarial Network. 14

LSUN Large Scene UNderstanding database. 13, 14

mAP mean Average Precision. 10, 26–28, 49, 50, 54, 56–60, 62, 63

MEI Music Encoding Initiative. 21, 24, 25, 30, 35, 37, 38, 49, 53

MIR Music Information Retrieval. 1, 20

MNIST Modified National Institute of Standards and Technology database. 40

MUSCIMA++ Dataset of handwritten music notation for musical symbol detection. 27, 28

Neon Neume Editor ONline. 24, 35–38, 50, 52, 53, 63, 67

NVIDIA's V100 Volta A powerful Graphics Processing Unit available for use on Compute Canada. 40

OMMR4all Semiautomatic online editor for medieval music notations. 22, 23

OMR Optical Music Recognition. 1, 2, 4, 8, 9, 20–31, 36, 37, 40, 41, 44, 45, 47, 50–54, 59, 63–66

PASCAL VOC 2007 Pattern Analysis, Statistical Modelling, and Computational Learning Visual Object Classes Challenge 2007. 10

PhotoScore Music score scanning program. 25

PyTorch Open source machine learning library for Python. 45, 49

R-CNN Region-Based Convolutional Neural Network. 10, 26–28, 45

RenderGAN Generative Adversarial Network with additional rendering step. 20

ResNet Residual Neural Network. 14, 26

ResNet 18 model 18-layer Residual Neural Network. 48

RetinaNet A combined Focal Loss and Feature Pyramid Network. 28

RoI Region of Interest. 10

RPN Region Proposal Network. 10

SIMSSA Single Interface for Music Score Searching and Analysis. 8, 23, 25, 66

SS Selective Search. 10

StyleGAN Style-based Generative Adversarial Network. 13, 17, 39, 40, 51

SVM Support Vector Machine. 22

TorchVision Computer vision package for PyTorch. 48

U-Net Convolutional Neural Network developed at the University of Freiburg, Germany for Biomedical Image Segmentation. 28

VGG16 Visual Geometry Group model from Oxford (OxfordNet). 10

w-mAP weighted mean Average Precision. 54, 56, 58–60, 62–64, 67

WGAN Wasserstein Generative Adversarial Network. 13

WGAN-GP Wasserstein Generative Adversarial Network with Gradient Penalty. 13

YOLO You Only Look Once Object Detection System. 27

1 Introduction

Square notation, dating back to the 13th century, is used in a number of manuscripts maintained by many libraries and archives. In order to preserve the cultural and musical information found in these manuscripts, many of them have been scanned and uploaded for widespread digital access, unlocking the possibilities of applying computer vision operations in the realm of Music Information Retrieval Music Information Retrieval (MIR) to automatically extract musical information from the scanned images. Within MIR, Optical Music Recognition Optical Music Recognition (OMR) is a research field focused on the detection and encoding of musical information into a machine readable output, taking a scanned image of musical content (e.g., a page of a score) as input. When applying heuristic, conditional programming to the detection and pitch classification of square notation, it is difficult to fully anticipate the inherent variety in the manuscripts' handwriting styles, ink bleed-through, and background textures. In machine-learning-based OMR workflows, often a large amount of annotated manuscript training data is needed to encapsulate these variables. Unfortunately, manually transcribed manuscripts are rare, and without increasing their amount, which can be costly, the OMR models may not learn the necessary parameters for accurate results.

Traditional data augmentation seeks to resolve this problem by transforming existing input data to synthetically create more training data for a machine learning task. While these processes introduce new data by, for example, rotating or adding noise to an image, the results are only slight adjustments to what is already existent. Recent research has introduced a novel strategy for data augmentation from a more fundamental standpoint, generating completely new examples instead of transforming pre-existing ones. Generative Adversarial Networks Generative Adversarial Network (GAN) entangle two neural networks in a game, where one network, generator (G), is trained to generate examples with the same statistics as a given training set. The second network, discriminator (D), takes data from the training set and data generated by G, and evaluates whether each input is real or not. The errors from D are propagated backwards towards G, which learns to generate more realistic data, representative of the training set. This loop optimally continues until D is rating inputs 50% real and 50% fake on average, meaning that D is fooled by G and cannot distinguish which examples are real or synthetic. The goal is to use this G to create realistic manuscript training data for use in a novel OMR workflow for square notation, comparing whether training datasets comprised of real and synthetic manuscript data outperform exclusively real training datasets.

The novel OMR workflow for square notation envisioned for this research is used to automatically locate the important musical elements on each manuscript page and classify their vertical position on their respective staff, the four-line system used to establish the notes' musical pitches. To locate the musical elements on the page, an object detection model is proposed. By providing the model with the coordinates and name of each relevant musical symbol on the page, it learns to make predictions for where these objects are located in the overall manuscript image. The detected musical symbols then need their vertical position to be classified in relation to the nearest staff upon which they appear. By establishing the vertical position of clefs and notes on the page, their final pitches can also be encoded. This task is handled by a separate classification model, trained on the enlarged bounding boxes of musical symbols that reveal the surrounding staff lines and the musical symbol's relation to them. These two models will be trained with the real manuscript data to establish the baseline evaluation metrics. The models will then be retrained with the identical manuscript data in addition to the GAN-synthesized data, and the evaluation metrics will be compared. An increase in the metrics pertaining to the real and generated dataset will demonstrate that the OMR of square notation is improved with the use of synthetic manuscript training data.

1.1 Thesis Organization

This thesis spans five chapters, including this introductory chapter. Chapter 2 expands into background literature for the machine learning methods used in this thesis. The first section is focused on modern deep learning, object detection, and data augmentation practices. The second section includes an overview to GANs and how they have been used for creating training data for other machine learning tasks. The third section provides a more in-depth understanding of square notation, and a review of existing research on the topics of machine-learned OMR for square and other music notations. Chapter 3 begins with an overview of the the manuscript image generation and processing workflows and the Medieval manuscripts used in this research. The extraction of the ground truth manuscript information is then explained, followed by the processes for using the GAN to generate synthetic neume components and manuscript pages. The following sections break down the OMR workflow into the separate object detection and staff position classification steps, and finish with the evaluation metrics used to measure the performance of the workflow. Chapter 4 encompasses the experimental results of the baseline and synthetic manuscript page training datasets for the OMR workflow. Finally, Chapter 5

includes additional discussion of the findings in Chapter 4 and suggests possible avenues into future work.

2 Background

This chapter presents more information about recent strides in machine learning that have made way for the development of Generative Adversarial Networks (GANs) and the existing body of work that has already been applied to the automatic transcription of square notation. Section 2.1 provides a general overview of square notation. In Section 2.2, the general concept of machine learning is introduced, including deep learning, object detection, and data augmentation, as a gateway to Section 2.3, where an overview of GANs and their utility for creating training data is provided. Finally, Section 2.4 includes a survey of recent OMR research performed for the automatic transcription of square and other music notations.

2.1 Square Notation

In the 13^{th} century, square notation evolved out of the earlier forms of recording music in writing. It was introduced as a notation for Gregorian chants, the unaccompanied, monophonic sacred songs in the Roman Catholic Church. Earlier chant manuscripts featured neumes—musical symbols—directly above corresponding lyrical text on a page, though no staff lines were included. Square notation introduced a standard staff with four lines and *clef* symbols to establish relative pitch relationships for neumes on the staff (Figure 2.1).

Figure 2.1: Page of the square notation Salzinnes Antiphonal manuscript. [1]

Square notation is an early form of musical notation used for monophonic vocal music, or chants, the singing of words and phrases in religious settings. Lyrical text below each staff organizes the musical information into phrases or syllables, which are further broken down into neumes, and once more into neume components, the class of individual symbols that make up the most succinct musical information in square notation. Similar to modern musical notation, *clefs* at the beginning of each staff indicate the relative pitch position for the following neume components. The rectangular and diamond-shaped *punctum* and *inclinatum* components, respectively, represent individual note pitch information. *Obliques*, the wide, descending parallelograms represent two pitches, one at the beginning and ending staff position. At the end of most staves, the *custos* indicates what the first pitch will be on the following staff. A section of a square notation manuscript can be seen in Figure 2.2.

Neumes in square notation generally refer to musical phrases of lyrical text, which broadly

1. `https://smu.ca/academics/archives/the-salzinnes-antiphonal.html`

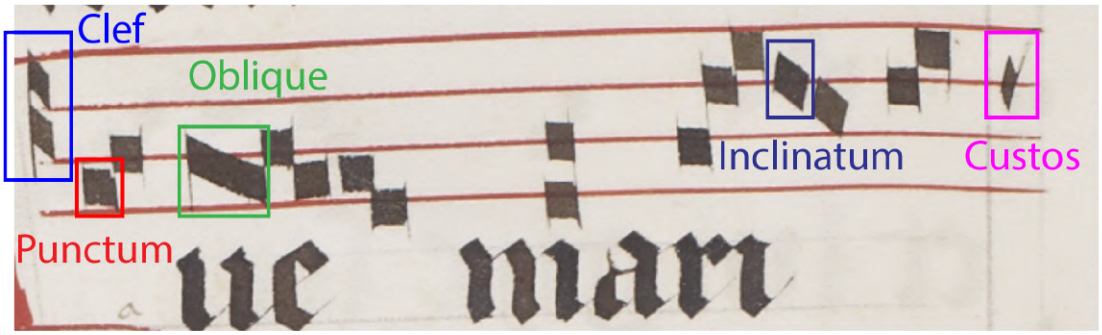

Figure 2.2: Neume components in square notation. From left to right, the first unique symbol occurrences are a *clef*, followed by rectangular *punctums*, a diagonal *oblique*, and descending diamond-shaped *inclinatums*.

refers to a sequence of one or more successive note pitches. The neume's constituent neume components are the smallest building blocks of musical information in square notation. Figure 2.3 shows the most commonly occurring neume components. *Punctums* and *inclinatums* represent individual pitches, while *obliques* represent two pitches: one at each of the starting and end positions of its diagonal form. Compound neumes are made up of two or more consecutive neume components either explicitly connected by a penstroke, or they are grouped together based on the relationship to a corresponding syllable or phrase in the lyrics (Figure 2.4). Musicologists have been able to identify a subset of common compound neumes, though many are too complex and uncommon to receive a specific categorization other than generally being a compound neume. Thus, individual neume components will be focused on exclusively throughout this thesis since they represent all of the note-by-note information on a page, which can be encoded discretely into a machine-readable format.

Figure 2.3: Neume component examples.

Figure 2.4: Compound neume examples.

Similar to modern musical notation, square notation features *clefs* that signal the reader of the position to pitch relationships from staff to staff (Figure 2.5). The vertical center position of *c*- and *f-clefs* establish this relationship for the following neume components, indicating on which staff line the reference pitch is found. Figure 2.6 shows the nine possible neume component positions on the staff and illustrates how the relative pitch is subsequently established from the most recent occurring clef change.

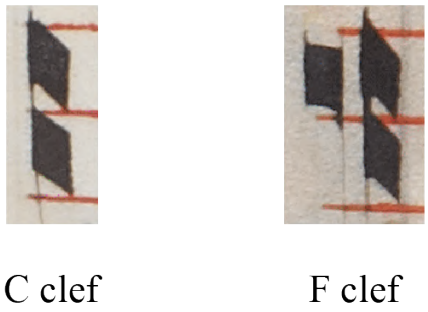

Figure 2.5: Examples of the *c*- and *f-clefs*.

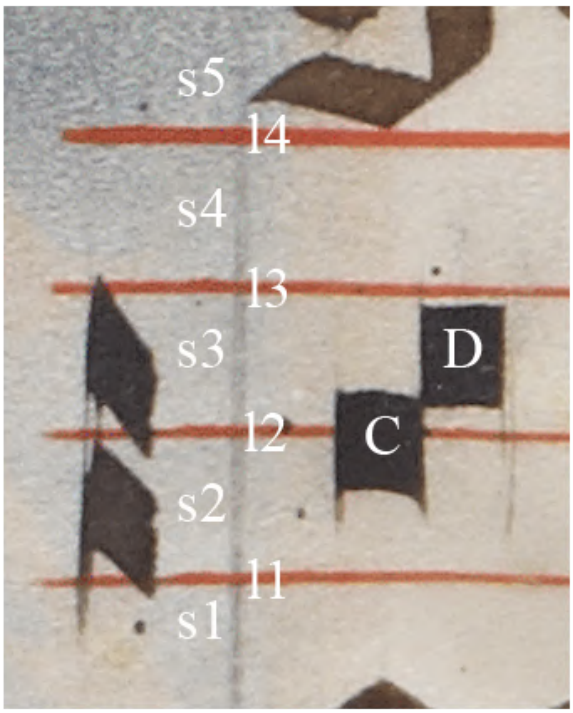

Figure 2.6: Example representing a *c-clef* that centers the C pitch on line 2 (l2) and subsequent neume component pitch encodings.

The focus on the OMR of square notation stems from a research initiative centered at McGill, the Single Interface for Music Score Searching and Analysis (SIMSSA) project (Hankinson et al. 2012). Motivated by community efforts to make archival music information widely available and digitally accessible, SIMSSA incorporates an expanding range of tools for encoding, editing, and searching musical information in a large, public database. Ideally, this research will be incorporated into SIMSSA, which could increase the OMR efficiency and accuracy of square notation manuscripts to be included in the database.

2.2 Machine Learning

With recent developments in the computational efficiency and processing power of personal computers, machine learning has become a quintessential toolset for researchers. In contrast to traditional heuristic algorithms, a complex function or model is built out of a large set of input examples referred to as the training dataset. In supervised machine learning, every example in the training dataset has a number of features, x, that are assumed to correspond to an output label, y. Through multiple iterations or passes over the training dataset, the model is tuned

to the examples provided, learning how x generally leads to y. Tuning a model too quickly or without enough examples can lead to consistent, but inaccurate results. The model does not accurately capture the features that contribute to y, lacking the complexity to learn about any underlying trends. This is referred to as underfitting. Training a model for too long without a broad set of examples can lead to memorization of the training data, resulting in a model that is too complex and does not leave room for making accurate predictions when provided with new data. This is referred to as overfitting. More data always comes as a benefit to machine learning operations, assuming that the examples involved represent a wide diversity that will generalize well to the test dataset, which is comprised of examples with the same set of features x, but without an output label, y. Examples from the test dataset are fed into the final trained model, and the model provides a corresponding y as an output. The classification accuracy of the test dataset is often used as the core metric for evaluating the model.

2.2.1 Deep Learning

In recent years, machine learning has become increasingly complex, achieving higher levels of abstraction via deep learning in areas of image processing and speech recognition (LeCun et al. 2015; Goodfellow et al. 2016). In a general deep learning framework, multiple processing layers are strung together to evaluate various characteristics from the original input data. Individual features described in the previous section compound to form these high-level attributes. For example, a deep learning model can be trained to detect facial attributes in a celebrity portrait dataset, handling the individual detection of eyes, noses, and mouths in different processing layers (Liu et al. 2015). This is referred to as object detection, a popular application in deep learning algorithms that has powerful implications for OMR research.

2.2.2 Object Detection

Object detection with machine learning is an age-old task that has continued to stay relevant for many types of image processing operations (Papageorgiou and Poggio 2000). Instead of classifying the entire image, smaller targets in the overall image are identified and usually assigned a label. An object detection dataset is composed of these "scenes" alongside corresponding text files that detail the coordinates of the bounding box that surround each target in the overall image, referred to as a bounding box, often including an assigned label as well. For

example, an operation to detect bounding boxes for cats and dogs in an outdoor setting will require many images of the animals in this setting, with an accompanying file for each image stating where the animals are located in the image pixel-wise (bounding box), and whether they are a dog or a cat. This simple localization task has a significant amount of application in security, medical, and engineering disciplines (Buczak and Guven 2016; Kourou et al. 2015).

A number of deep learning models have been generalized for adaptive use in custom object detection tasks. Girshick (2015) introduced the Fast Region-based Convolutional Neural Network (Fast R-CNN), improving upon the original R-CNN also proposed by Girshick et al. (2014). Fast R-CNN was able to train deep detection networks such as VGG16 (Simonyan and Zisserman 2014) nine times faster than the original. The improved architecture takes an image and regions of interest (RoIs) or bounding boxes as input into a fully connected CNN. Each RoI is divided into an array of sub-windows where each is individually pooled, reducing their dimensionality into a fixed-size feature map. Passing through fully connected layers of the CNN, the feature map is projected onto a final feature vector with two outputs: softmax probability estimates over the number of RoI input classes and the four values representing the coordinates of the bounding box (per RoI). Ren et al. (2015) developed the Faster R-CNN architecture as an extension of Fast R-CNN. They introduced a novel Region Proposal Network (RPN), which is a CNN used to take an input image and output bounding-box object candidates, each with a score representing whether it refers to the relevant set of object input classes (e.g., any of dog, cat, car, etc.) or the background of the image. The RPN is intertwined with the Fast R-CNN by sharing a subset of fixed convolutional layers between the two models during training. RPN is first trained independently to output object proposals used for training input to the Fast R-CNN detector. These detections are then used to continue training the RPN while fixing the two model's shared convolutional layers and only updating the unshared layers in the RPN. Finally, the unshared layers in the Fast R-CNN are updated while keeping the shared layers fixed, resulting in a unified object detection architecture. Compared against previous Selective Search (SS) pipelines (Uijlings et al. 2013) on the PASCAL VOC 2007 dataset (Everingham et al. 2007), a baseline mean average precision (mAP) of 68.5% was achieved for a Faster R-CNN with *unshared* convolutional layer, beating the SS baseline of 66.9% and evaluating in 198 ms as opposed to 1830 ms with SS. Using a Faster R-CNN with *shared* convolutional layers and trained on the PASCAL VOC 2007 and 2012 data, following Girshick (2015), the mAP increased to 73.2%. The Faster R-CNN, selected for use in this research, is a competitive architecture for increasingly accurate and efficient object detection.

As universal as object detection has become, it is still a very expensive, time-intensive process on most consumer hardware, motivating researchers to introduce novel approaches for reducing the overall computational load and processing time. This also coincides with an increase in camera resolution, which provides researchers with more detailed, pixel-rich source images for object detection at an increased computational cost. Images can be down-scaled, but this comes as a direct loss of information, reducing the overall number of pixels, especially considering those that might make up a very small target in a scene. For detecting small objects in a large scene, image segmentation or tiling is a simple way to transform a dataset of computationally demanding images into new sets of smaller image scenes (Plastiras et al. 2018). This is done by first selecting desired dimensions for each chunk of the larger image. Next, x- and y- pixel overlap values are declared to ensure no targets in the scene are omitted because they would have otherwise been broken up by a non-overlapping segmentation. The x-overlap and y-overlap values must exceed the minimum width and height respectively of any target in the scene or the targets are at risk of not being included after the segmentation is performed.

As mentioned previously, image segmentation or tiling can be used to localize very small objects in a large image scene, such as locating small note heads in a large music manuscript. Tiling is thoroughly explored by Unel et al. (2019). By segmenting a large image of a street scene, the researchers performed an object detection and classification task for cars and people in the smaller images. Training their model with the smaller tiles, the car and human targets were detected in each segment and later merged to encompass the overall scene. When merging the segmented chunks after detection, some objects in the scene will have been detected more than once, since they could have appeared in more than one tile due to the segmentation pixel overlap. If bounding box areas overlap by more than 25%, the detection/classification with the higher accuracy is chosen and the other intersecting boxes are removed. The researchers also found that model complexity linearly increases with the number of tiles chosen, and to improve smaller object detection, using a larger number of smaller tiles leads to a direct increase in detection accuracy. Thus, it is a sensitive trade-off of quick real-world inference and accuracy of results, though segmenting images into a small number of tiles tended to maintain both desirable features.

2.2.3 Data Augmentation

In order to reduce the possibility of overfitting a machine learning model, it is desirable to have a large amount of diverse training data. Data augmentation is the process of creating new modified data out of pre-existing data by perturbing a subset of the original features in a training dataset, since they are sometimes small, or in rarer cases, too clean to generalize well to possibly noisier unseen data. In image datasets specifically, translation, mirroring, blurring, and other transforms are utilized to augment an existing example without changing its overall label, effectively increasing the number of examples in the dataset (Simard et al. 2003). This has remained a popular dataset extension tool for many years (Baird 1990), but the new data introduced by these augmentations do not represent entirely novel examples, providing only a limited set of possible alterations to the existing data. Very recently, the use of Generative Adversarial Networks (GANs) have been proposed as a possible solution to generate a greater diversity of augmented training data.

2.3 Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs), introduced and coined by Goodfellow et al. (2014), are comprised of two distinct deep learning models to train: the generator (G) and discriminator (D). The two models are intertwined in a sort of adversarial game, where G is trained to fool D by synthesizing data that mimics some real input data. Every iteration of training, D receives an input from one of two sources, the real input data or G-synthesized data, and it evaluates whether the input material is real or not. The errors from D are routed backwards through the architecture to G, via a method referred to as backpropagation (LeCun et al. 1989; Rezende et al. 2014). This training loop optimally continues until D is rating inputs 50% real and 50% fake on average, equivalent to a random guess. In other words, G becomes so proficient at creating synthetic data that D cannot distinguish between what is real and fake. GANs can be used to synthesize any type of input data, though their development has made significant impacts on image processing and data augmentation. The general GAN architecture can be seen in Figure 2.7.

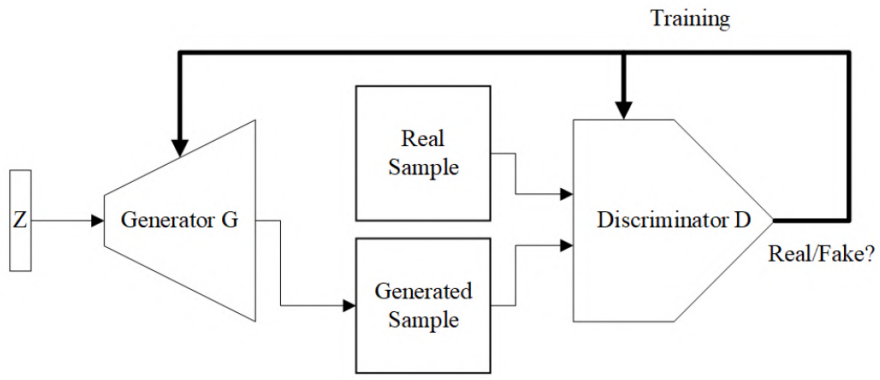

Figure 2.7: Standard GAN architecture (Öngün and Temizel 2018, 5).

2.3.1 GANs and Image Processing

Since the inception of GANs, synthetic image generation has become a very popular application of the architecture. Improved stability in Wasserstein GAN (WGAN) architectures made large strides in creating realistic synthetic images from real input data (Arjovsky et al. 2017), achieving high-quality image synthesis on the LSUN bedroom image datasets (Gulrajani et al. 2017) (Figure 2.8). Gulrajani incorporated a gradient penalty to the WGAN architecture (WGAN-GP) which improves upon the undesirable behavior of weight clipping in the discriminator model. Wang et al. (2018) introduced a conditional GAN architecture that allowed for specific object omission or replacement with a new object when synthesizing images. For example, a training dataset containing photos from the perspective of a car dashboard with objects labelled in the scene led to a conditional GAN that would take inputs from the developer to omit cars in the road, or change the road material from asphalt to cobblestone (Figure 2.9). This form of style transfer was explored by Gatys et al. (2016), who employed a Convolutional Neural Network (CNN) to render natural image scenes in different artistic styles labelled prior to learning (Figure 2.10). Karras et al. (2018) developed StyleGAN, which accomplishes an unsupervised separation of physical features in the input images. For example, the researchers trained StyleGAN with a large dataset of real human portrait images, and the architecture provided a generator that synthesized highly realistic images, with learned, adjustable parameters for generating portraits with differing freckle density, hair type, etc. (Figure 2.11). Oeldorf and Spanakis (2019) augmented the original StyleGAN to create their own multi-class architecture, LoGAN, used to synthesize company logos. With LoGAN, labels are used to train the network to generate distinct categorizations of logos, while still inferring more specific, class-dependent features with unsupervised learning during the training process (Figure 2.12). The

photo-realistic qualities of GAN image synthesis have many implications towards extending training datasets for subsequent machine learning operations.

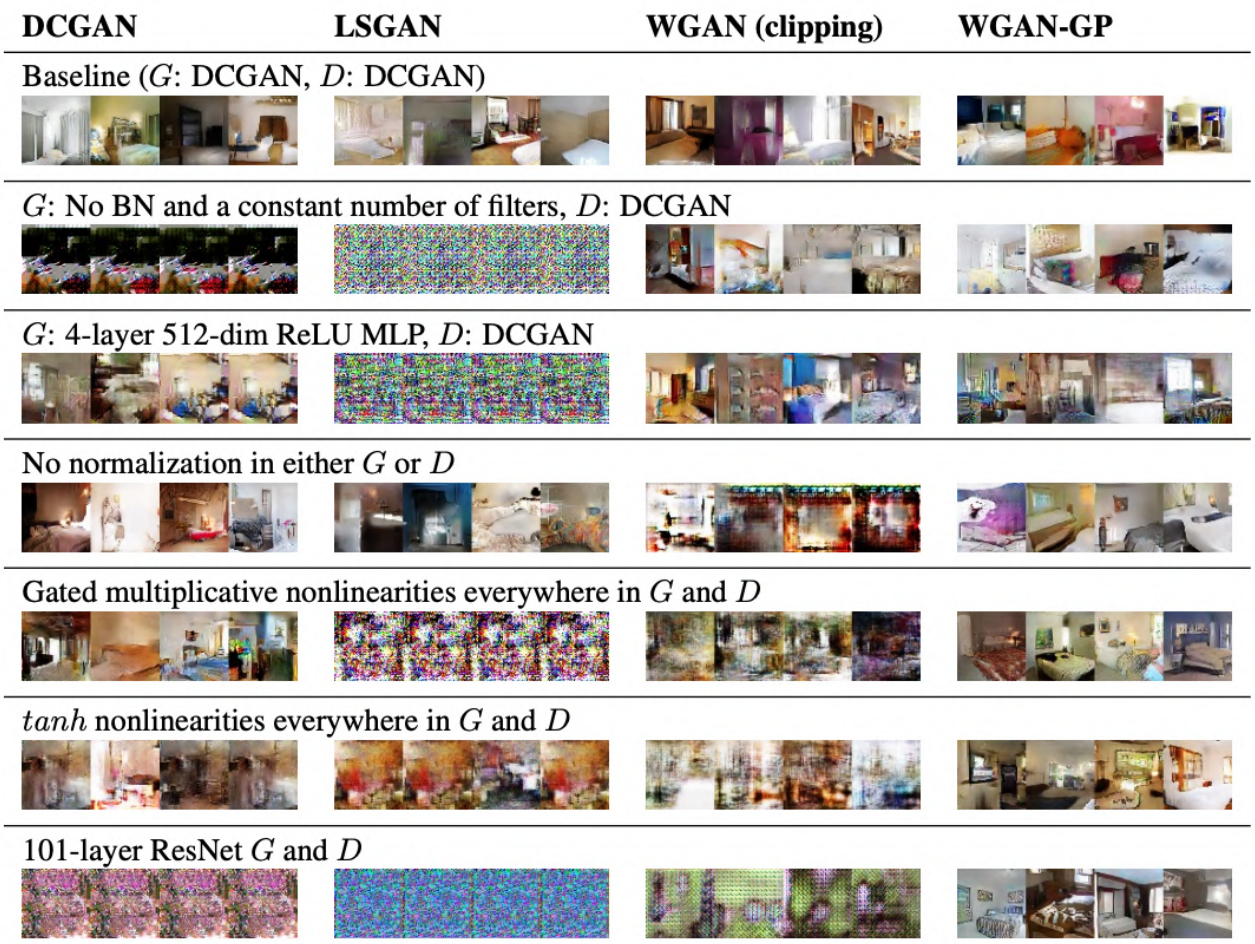

Figure 2.8: Enhanced training of the LSUN bedroom dataset with the improved Wasserstein GAN architecture (Gulrajani et al. 2017, 6). The baseline row implements the deep convolutional GAN (DCGAN), the second row without Batch Normalization (BN) in G, the third a Rectified Linear Unit (ReLU) Mutilayer Perceptron (MLP), the last a 101-layer Residual Neural Network (ResNet). The second column is a Least-Squares GAN (LSGAN) architecture. In row five, the gated multiplicative nonlinearities are Long Short Term Memory (LSTM) gates.

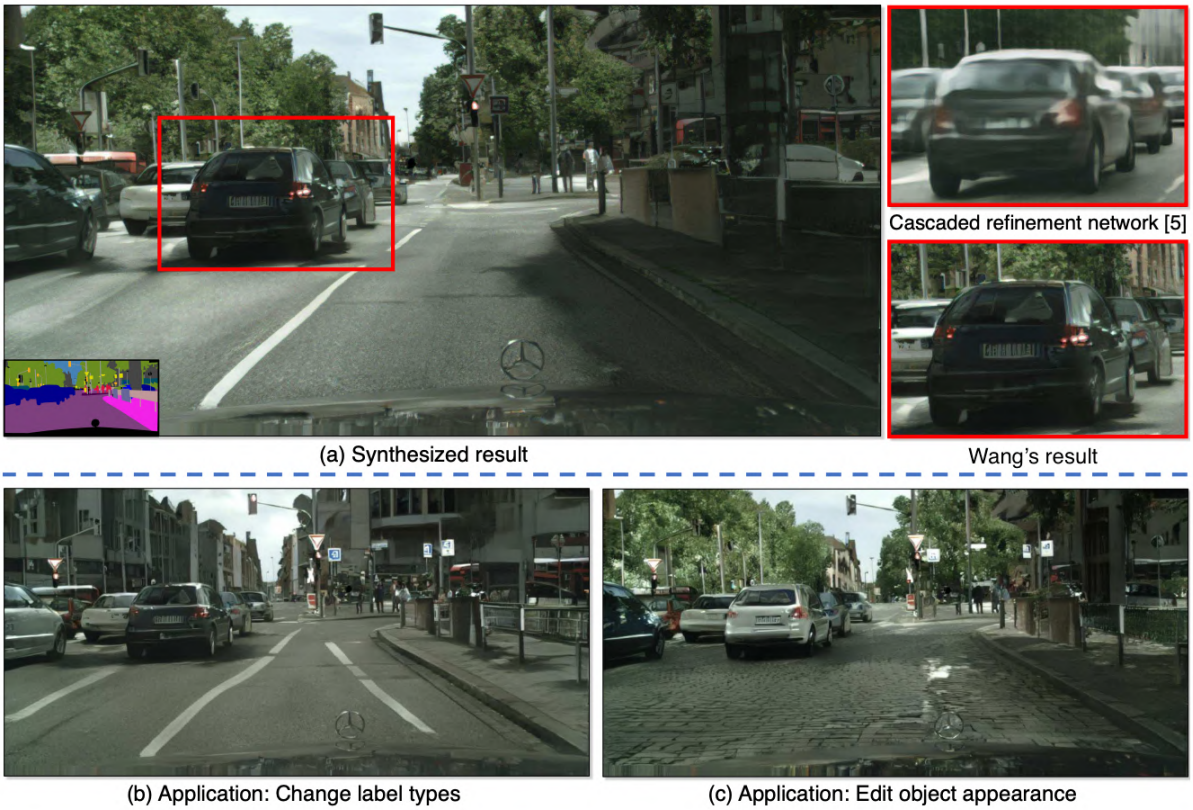

Figure 2.9: Style transfer GAN examples (Wang et al. 2018, 8798). In (b) and (c), style transfers are made to the road material, surroundings, and car color from (a).

Figure 2.10: Artistic style transfer GAN examples (Gatys et al. 2016, 2418). The original image (A) is applied learned styles from famous artworks by J.M.W. Turner (B), Vincent van Gogh (C), Edvard Munch (D), Pablo Picasso (E), and Wassily Kandinsky (F).

Figure 2.11: StyleGAN-generated portrait examples. [2]

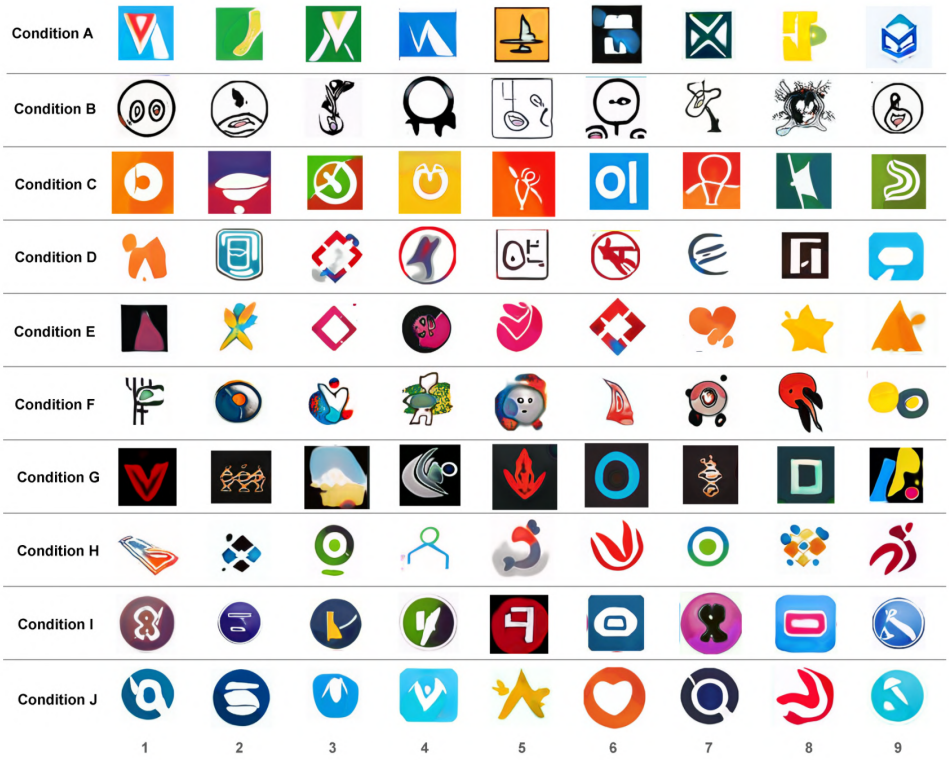

Figure 2.12: LoGAN-generated logo examples. [3]

2. https://github.com/NVlabs/stylegan
3. https://github.com/cedricoeldorf/ConditionalStyleGAN

2.3.2 Using GANs for Creating Training Data

By utilizing GANs, researchers have been able to extend training datasets from a more foundational standpoint. Parameterizing the general aspects of the source material and synthesizing convincing new examples based on the original, labelled dataset, researchers have evaluated the utility of GAN data augmentation through both qualitative and quantitative means (Bowles et al. 2018).

GANs are continually creating realistic renderings of labelled images. Using the CIFAR dataset, Denton et al. (2015) created synthetic images that were highly realistic to human evaluators (Figure 2.13). Utilizing a Laplacian pyramid framework [4] to generate the images, 40% of the time, subjects were not able to discern synthetic CIFAR examples from the real ones. This is a large increase over a 10% discernible difference subjects perceived from a general GAN baseline. Shmelkov et al. (2018) argued that subjectively rating GAN images is not enough of an evaluation since it is lacking any quantitative criteria. They introduced two separate measures of recall ("GAN-train") and precision ("GAN-test") by incorporating classification neural networks to handle the separate evaluations. For the GAN-train metric, the network was trained on synthetic images and evaluated on a set of real images. The GAN-test metric is comprised of the inverse: training a model on real images and evaluating with synthetic ones. Synthetic images are considered good quality when GAN-train accuracy is close to validation accuracy. The same case is made for GAN-test, with the caveat that a higher accuracy in GAN-test is indicative of the original GAN simply memorizing the dataset and overfitting. During experimentation, they found that a training dataset of 2,500 real images padded with 50,000 GAN images resulted in a higher GAN-train accuracy than a network trained with 5,000 real images. In practice, the GAN images were not as realistic as their counterparts, though they still revealed an underlying diversity that 5,000 real images alone could not capture. Both of these subjective and quantitative evaluations suggest that GAN architectures can extend real datasets and lead to improvements in the training process.

No matter how large a dataset, it is generally preferred to have a surplus of available training data to reduce overfitting (Goodfellow 2017). As discussed previously, GANs can be used to increase the size of the dataset of any task by shifting the training focus inward, artificially

4. A Laplacian pyramid consists of resampling and smoothing an image while continually reducing its overall resolution. In this case, a GAN was used in between each step of the pyramid on the way to the final output result.

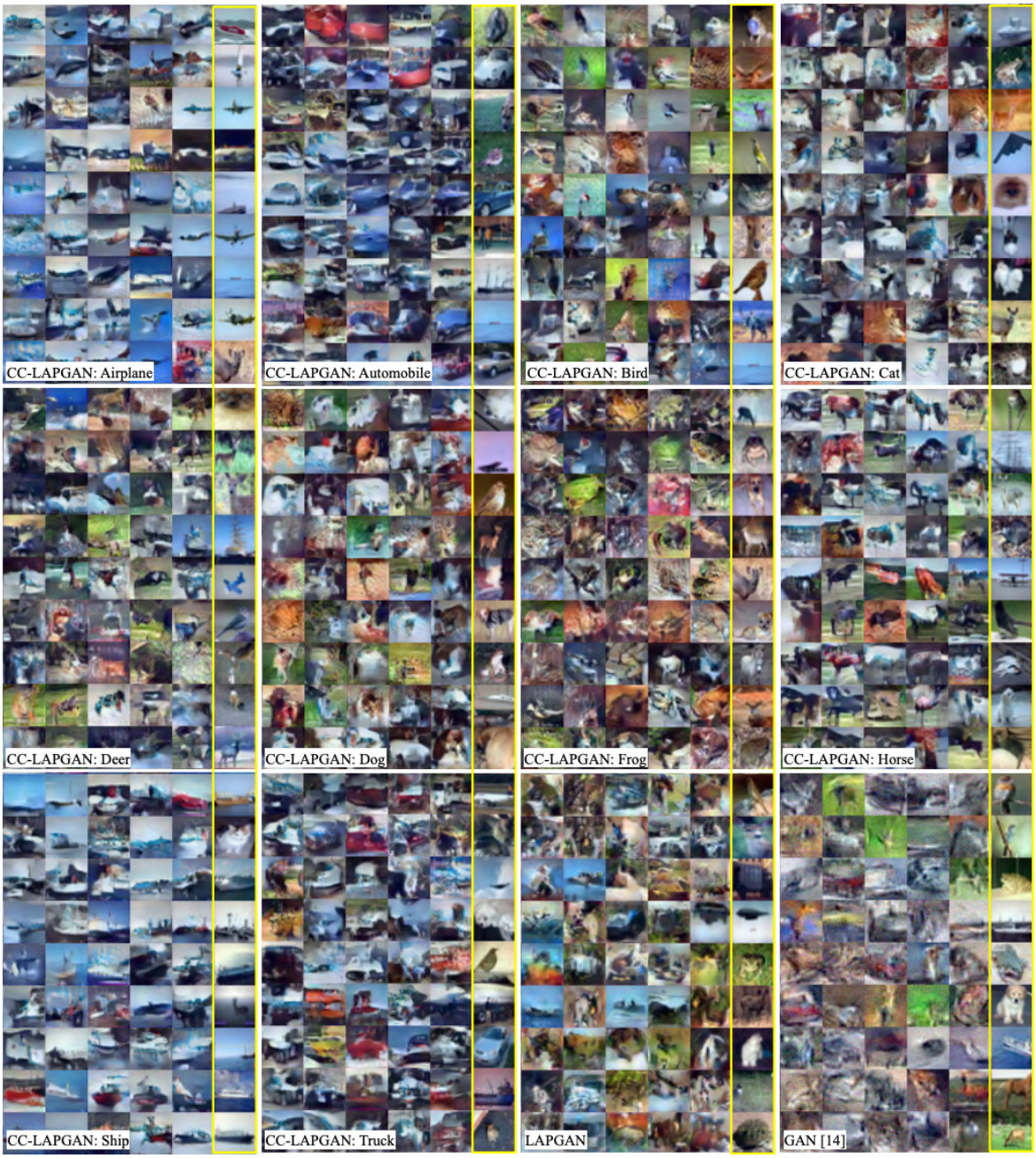

Figure 2.13: Synthetic CIFAR images for subjective evaluation (Denton et al. 2015, 7).

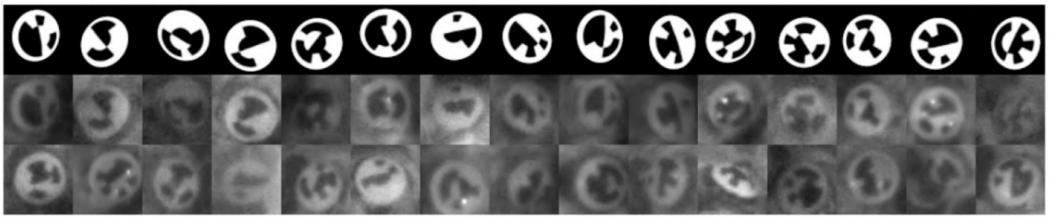

Figure 2.14: **First row:** Model images without augmentation functions. **Middle row:** RenderGAN output images. **Last row:** Real images of bee orientations (Sixt et al. 2018, 3).

extending the amount of available data by generating new examples with corresponding features and labels. For example, RenderGAN has been developed for these purposes, greatly reducing the amount of time needed to manually label many individual examples of bee orientations in a larger image scene (Sixt et al. 2018). RenderGAN first learns to generate an image representing the orientation of a bee's positioning, then a cascade of augmentation functions are applied to blur and add lighting and background detail to the generated image (Figure 2.14). Once the detailed image is generated, the discriminator ultimately decides if the synthetic image is distinguishable from the real training data. Compared against a previous image recognition process, there was a rise from 55% to 96% correct labelling of bee orientations when using a training dataset that contained the real data and RenderGAN's synthetic images. GANs are successfully being utilized to both augment and extend otherwise finitely available real data, resulting in accuracy increases for object classification and detection tasks (Antoniou et al. 2018). In this thesis, GANs are used to synthesize artificial training data to increase the performance of a novel OMR workflow for square notation.

2.4 Optical Music Recognition

Automatic transcription is a popular application in the realm of Music Information Retrieval Music Information Retrieval (MIR), specifically, with optical music recognition Optical Music Recognition (OMR) software. OMR is focused on the detection and encoding of musical information into a machine-readable output, taking a scanned image of musical content (e.g., a page of a manuscript) as input. The general OMR workflow is outlined by Rebelo et al. (2012):

1. Image pre-processing
2. Staff line detection and recognition of musical symbols
3. Reconstruction of the musical information

4. Construction of a musical notation model

Handwritten square notation was one of the first notations to introduce the use of staff lines to establish position and pitch relationships for neumes, the musical symbols of the notation. It is used in a number of manuscripts collected by many libraries and archives, and thus, it is a sustained research area on the development of archival OMR software. The following sections provide an introduction to the existing research on employing both heuristic- and machine-learning-based automatic transcription workflows for OMR of square and other music notations.

2.4.1 OMR for Square Notation

With and without machine learning, many researchers have employed a variety of strategies for the automatic transcription of square notation. Vigliensoni et al. (2011) assembled existing heuristic transcription algorithms to create an automatic transcription workflow for square notation in the *Liber Usualis*. They combined six distinct processes that used staff-finding algorithms from Miyao and Okamoto (2004) and the MusicStaves Gamera Toolkit [5] to detect the positioning of neumes in the staff and establish their pitch relationships to the most recent occurring *clef*. Special conditions were also established for compound neumes such as the *podatus* and *torculus* (Figure 2.4), which involved separating the symbol into individual neume components. Their error analysis was performed on a dataset of "20 random pages with a total of 2219 neumes and 3114 pitches correctly labelled" (Vigliensoni et al. 2011, 427). The detections were encoded into the Music Encoding Initiative (MEI) structure, a research community effort to standardize the machine readable format of various musical notations. [6] Using the Miyao staff-finding algorithm, compound neume considerations, and staff line spacing correction led to a 97% correct detection rate of the first pitch of a neume. Compared against a workflow using the MusicStaves Gamera Toolkit, with no compound neume considerations and no staff line spacing correction, this was a statistically significant increase from an 85% detection rate of the first neume pitches.

Ramirez and Ohya (2014) envisioned a new workflow for OMR of square notation that combined staff area detection and machine-learned classification of musical information. Contrary to arguments made by Rebelo et al. (2012), Ramirez argued that pixel-based staff line removal

5. `http://music-staves.sf.net/`
6. `https://music-encoding.org/`

could possibly lead to the introduction of unwanted noise. Starting from a grayscale binarization of the target manuscript image, they created a general staff template that was iteratively compared against the actual staves on a page, in an optimization task that rotated the template until it had a maximum alignment. This was repeated for all of the staves on a page, continually updating the staff rotation angle and keeping track of their respective pixel coordinates. They also created neume templates that were scaled and rotated with respect to the staff's height and rotation angle, which were similarly used in a maximum alignment optimization process for the neumes within each detected staff area. In the maximum alignment detection task, they detected 804 of 847 possible staves (95%), though only 5,000 of around 8,000 neumes were detected and correctly labelled (62%). Another 2,150 neumes were detected but incorrectly labelled (27%). Surveying the machine-learned classifiers explored by Rebelo et al. (2012), they decided to use individual Support Vector Machines (SVMs) to redo the symbolic classifications of the 7,150 detected neumes, both correctly and incorrectly classified. Training separate SVM classifiers with individual sets of 8, 9, and 16 neume and neume component classes, the SVM trained on 8 classes achieved an improved classification accuracy of 92% across the detected neumes.

Wick et al. (2019) developed a novel staff line and symbol detection workflow for square notation, employing a Fully Convolutional Network (FCN) to handle pixel-based predictions for both (Long et al. 2015). Their staff detection algorithm acts on a grayscale, deskewed manuscript page as a whole, training an FCN to detect individual staff lines that are heuristically grouped into "polylines," making up every staff on the page. The coordinates from each staff grouping are then used to extract individual staff images from the whole manuscript page, and use them as input to the following symbol detection step of the workflow. For locating the clefs, individual neume components, and accidentals, they trained another FCN to handle the classification of these individual glyphs. For their staff detection approach, over 99% of all staff lines were both detected and correctly classified, culminating in an F_1-score of 99.7%. An F_1-score is a metric that combines the precision, the ratio of correctly predicted positives to the total number of predicted positives, and recall, the ratio of correctly predicted positives to the total number of actual positives. Their best FCN model for neume component detection and classification achieved an F_1-score above 96%. They also introduced their own metric, diplomatic symbol accuracy rate (dSAR), factoring in the correct labelling of symbol type and location, and achieved about 87% accuracy with the same model. Wick and Puppe (2019) incorporated the staff line and symbol detection models into *OMMR4all*, a web-based OMR

and correction framework for square notation (Figure 2.15).

OMR has seen many advancements in recent years, and it coincides with a large-scale focus in the community to make archival music information available in a digitally accessible, public database. Hankinson et al. (2012) introduced the Single Interface for Music Score Searching and Analysis [7] (SIMSSA) project, which is still in development today. Incorporating digital document viewing, musical symbol searching, and OMR software, it is a growing online database that encompasses the wider community focus to digitize archival manuscripts and encode their coinciding musical information in a searchable manner. Successful transcription of square notation with GANs developed in this thesis could speed up the data acquisition process for SIMSSA, processing and encoding more archival manuscripts into the database.

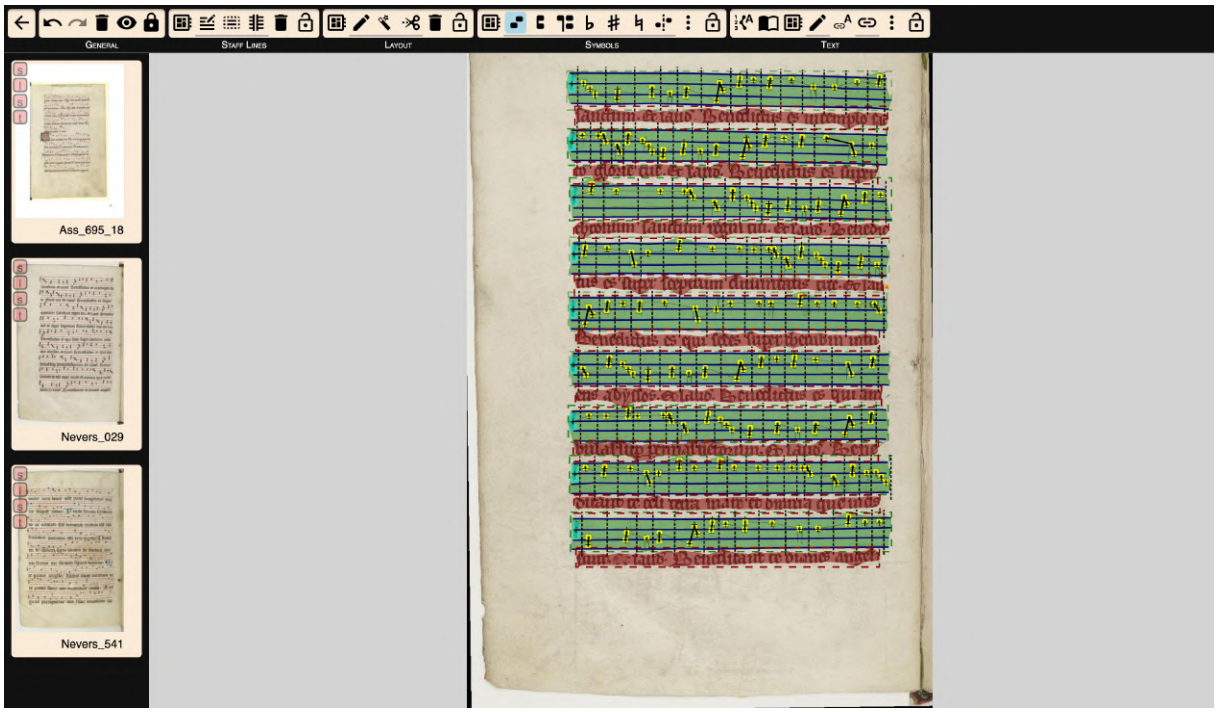

Figure 2.15: OMMR4all user interface. [8]

2.4.1.1 Partial OMR for Square Notation

Researches have also focused on individual steps of the OMR workflow. Calvo-Zaragoza et al. (2018) introduced a pixel-level classification of a manuscript page of square notation to separate the relevant musical data from the other features. Each pixel in the ground truth manuscript data was first labelled as belonging to the background, text, musical symbol, or

7. https://simssa.ca/
8. https://ommr4all.informatik.uni-wuerzburg.de/en/

staff layer of the page. Focusing on a pixel of interest, they enlarged the area around it to include surrounding pixels, operating on the hypothesis that neighboring information would provide the information necessary to correctly classify the original pixel (Figure 2.16). These blocks were passed as training input to a Convolutional Neural Network (CNN), which learned to classify the center pixel of interest. Training the CNN with two distinct manuscripts, the highest overall layer labelling accuracy of 88% resulted in an F_1-score of 88% in manuscript *M1* and 91.3% in manuscript *M2* at block input sizes of 51 x 51 pixels. They also performed a cross-manuscript adaptation, where the CNN was trained on one manuscript's data and evaluated on the second. The CNN trained with *M1* and evaluated with *M2* resulted in an F_1-score of 87.2% while the converse received a score of 73.3%, providing evidence that this adaptation is actually feasible, considering *M2*'s F_1-score only changed by 4% (91.3% to 87.2%).

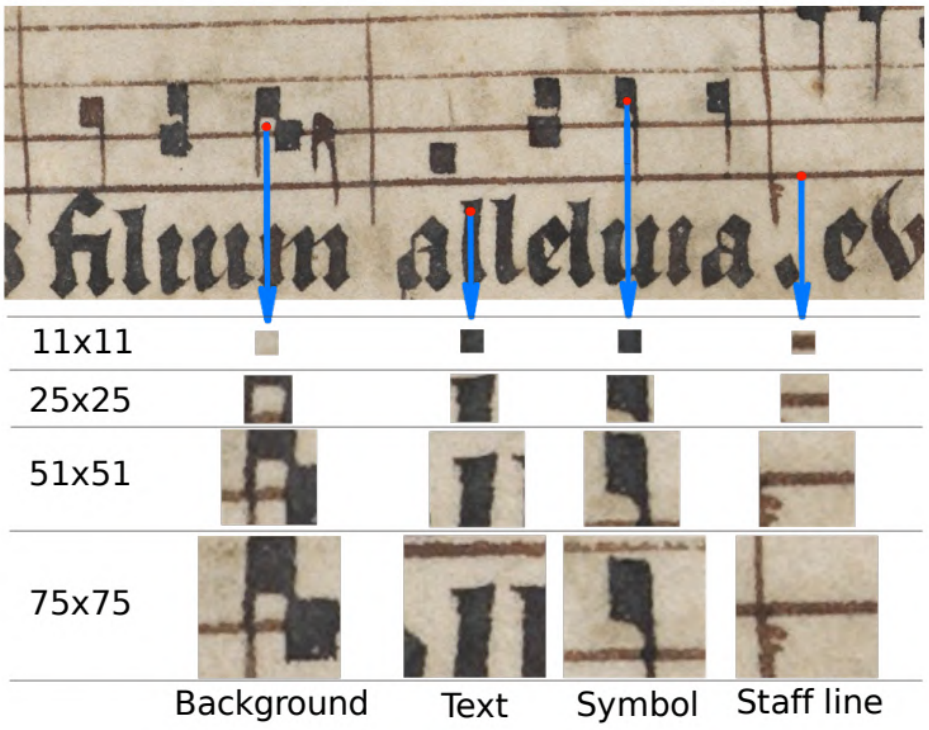

Figure 2.16: Block extraction examples for different CNN input sizes (Calvo-Zaragoza et al. 2018, 6).

While these various strategies and steps of the OMR workflow can correctly classify a majority of neumes and neume components, there is always a need for extraneous errors to be corrected. Burlet et al. (2012) introduced a browser-based, music notation editor called Neon.js, conceived for the editing of square notation transcription data through the web. Alongside Hankinson et al. (2011), Neon.js increased awareness of the MEI format. In accordance with

version 4.0 of the MEI, Regimbal et al. (2019) recently introduced Neon2, which uses Verovio [9] to render and edit the corresponding MEI files for square notation. The SIMSSA OMR workflow includes the current version of Neon2 and the pixel-level classification work by Calvo-Zaragoza et al. (2018).

2.4.2 Recent Applications of Object Detection in OMR

Researchers have recently pursued heuristic and machine-learned OMR for other handwritten musical notations such as common Western and mensural music notations. For example, Baró et al. (2016) created a learning-free method for recognizing compound music notes in polyphonic handwritten common Western music notation (CWMN) scores. Detecting primitive elements such as note heads, stems, beams, and flags, with heuristic line and blob detection, they were clustered into a hierarchical representation for identifying the compound notes (Figure 2.17). They compared their methodology against the commercial OMR software, PhotoScore [10] on two CWMN subsets of the CVC-MUSCIMA dataset (Fornés et al. 2012). They achieved higher precision and recall metrics than PhotoScore for one of the partitions containing mostly compound notes, though much lower scores in the other, where PhotoScore succeeded in detecting "easier" individual notes.

9. `http://www.verovio.org/`
10. `https://www.neuratron.com/photoscore.htm`

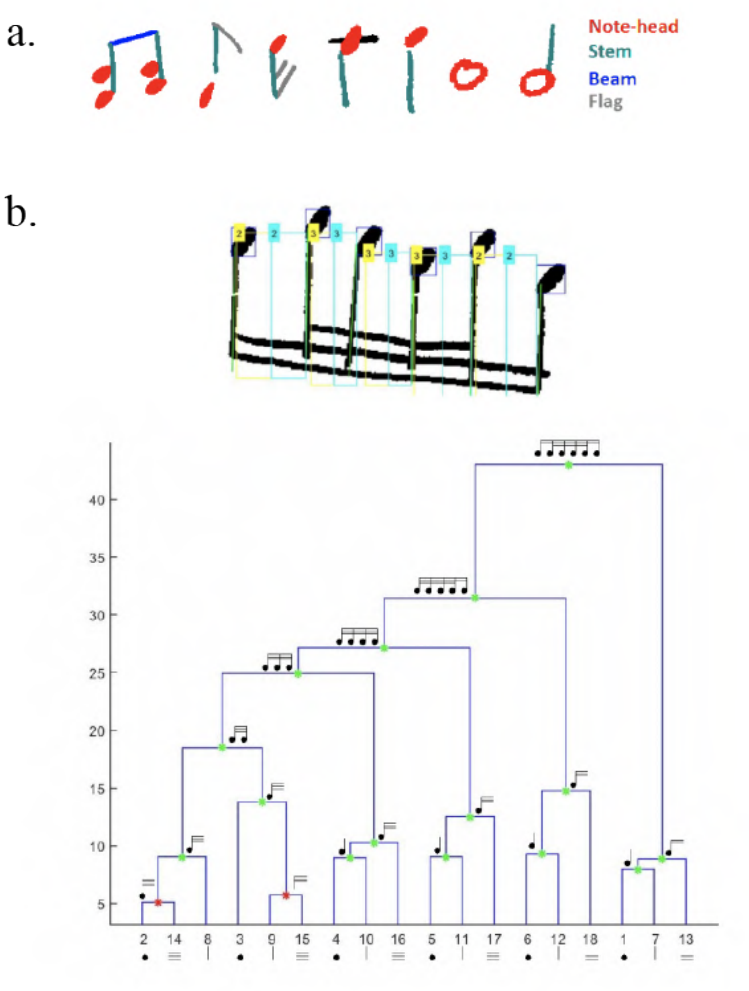

Figure 2.17: **(a.)** Primitive note elements detection candidates **(b.)** A detected compound note and hierarchical representation (Baró et al. 2016, 3).

In the realm of machine learning-based OMR, Pacha and Calvo-Zaragoza (2018) applied region-based CNNs for the automatic detection and staff position classification of notes and clefs in mensural notation. Utilizing the Faster R-CNN model (Ren et al. 2015) with a Inception-ResNet-v2 feature extractor (Szegedy et al. 2017), it was trained to detect the bounding boxes of all the musical symbols on the page. The vertical position of each musical symbol had to be established in the staff, so each detection was enlarged, then extracted with the target musical symbol still in the center (Figure 2.18). A label was assigned to the extracted image to indicate the position of the music symbol on the staff. A second CNN was separately trained on the extracted images in Figure 2.18 to determine their positions. Their test experiments yielded 66% mean average precision (mAP) and 76% weighted mAP for the detection of musical symbols with R-CNN, and 98% accuracy on the correct position classification of the detected

musical symbols on the staff. This work successfully represented an end-to-end OMR workflow as an extendable object detection problem.

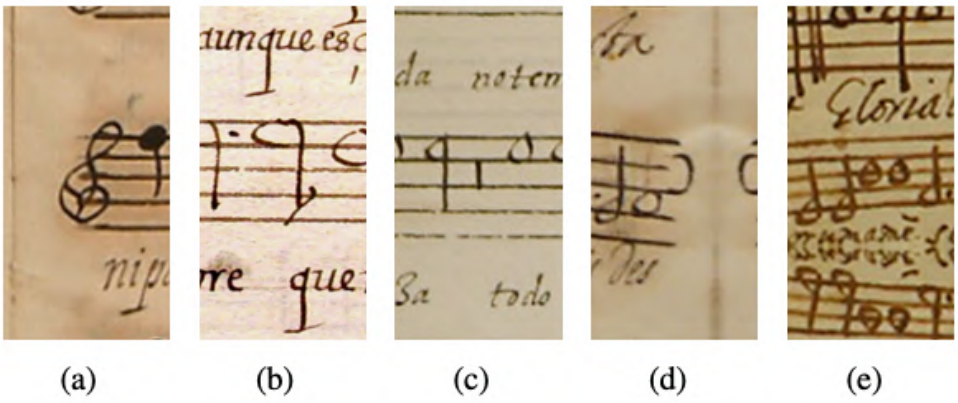

Figure 2.18: Example inputs to the position classification CNN (Pacha and Calvo-Zaragoza 2018, 243).

Metaj and Magnolfi (2019) also used the Faster R-CNN model to detect notes in the MUSCIMA++ dataset (Hajič jr. and Pecina 2017), containing 140 images of handwritten CWMN with and without staff lines and fully annotated across 105 classes of music symbols. They compared the object detection performance of a Faster R-CNN pre-trained with ImageNet [11] versus one trained from scratch, finding that the pre-trained model achieved higher mAP scores more quickly, even though it was trained on image contexts different from music notation. The model trained from scratch produced increasing mAP scores as it continued to train, though the pretrained network consistently maintained higher mAP scores, indicating that transfer learning can work well in diverse training contexts.

Huang et al. (2019) handled the object detection and staff position classification of printed CWNM in a single step. Using the You Only Look Once (YOLO) object detection system (Redmon and Farhadi 2018), they annotated the coordinates of each note alongside a number of classes: type, pitch, and duration. In the feature map generated by their training model, they expanded the coordinates of the candidate bounding boxes with seven defined pixel-value pairs and assessed the highest overall confidence across the type, pitch, and duration labels. In the 91% of notes correctly detected, they achieved a duration accuracy of 92% and a pitch accuracy of 96%. They achieved an average increase in accuracy from van der Wel and Ullrich (2017), who achieved an 80% note detection with 94% duration and 81% pitch correctness.

11. `http://image-net.org/index`

In a similar fashion, Calvo-Zaragoza et al. (2019a) trained a Convolutional Recurrent Neural Network (CRNN) to handle the end-to-end OMR of mensural notation. They reduced the *diplomatic symbol error rate*, a calculation of the necessary manual operations to correct an OMR process, to 7%, a decrease from 25% in another approach using Hidden Markov Models (Calvo-Zaragoza et al. 2019b).

Individual sections of the OMR workflow have also been scrutinized in recent years. Pacha et al. (2018a) focused on the object detection step, training a deep convolutional neural network on the MUSCIMA++ dataset. Optimizing different hyperparameters such as overall architectures, feature extractors, and the inclusion or omission of staff lines, the highest achieved mAP of their model was 87.8% on the test set. Lower detection accuracies were found in music symbol classes with less occurrences than others, a common issue in automatic OMR research.

Pacha et al. (2018b) extrapolated upon their previous OMR work and tested three popular object detection models against three distinct OMR datasets for symbol detection: MUSCIMA++ (handwritten CWMN), DeepScores (printed CWMN) (Tuggener et al. 2018), and Capitan (handwritten mensural notation, extended from the dataset used in Pacha and Calvo-Zaragoza (2018)). The three models compared on each dataset were Faster R-CNN, RetinaNet (Lin et al. 2017), and U-Net (Ronneberger et al. 2015), all variations of a general CNN architecture for object detection. RetinaNet had the shortest training and evaluation time for all datasets, though it suffered greatly in detecting any small objects, even common symbols like note heads. U-Net, classifying the score images on a pixel-based level, achieved the highest mAP scores across all three datasets, though it took an enormous amount of time to train the 107, 39, and 56 classes respectively, about 2–3 hours per symbol, making it impractical for situations requiring consistent retraining. Faster R-CNN performed well on the DeepScores and Capitan datasets, though it struggled with small, bunched symbols in MUSCIMA++.

The survey performed by Pacha et al. (2018) provided insights regarding how to best design and implement the object detection portion of the OMR workflow in this thesis. Pacha and Calvo-Zaragoza (2018) directly motivated the design of the position classification task. In the next chapter, the specific methodologies for the synthetic page generation preprocessing step and the OMR workflow will be described, along with the metrics established for evaluating the entire process.

3 Methods

In this chapter, the implementation of the neume component object detection and position classification workflows is discussed in detail. In Section 3.1, an overview of the workflow is provided, outlining the GAN pre-processing chain for the synthetic manuscript data and the main block of processes trained with the real and synthetic data. In the subsequent sections, each process is explained in further detail, providing evidence of initial approaches that led to the final overall workflow.

3.1 Overview of Workflow

The workflow envisioned for this project includes a network of procedures where both the real and synthetic manuscript data is processed. For the synthetic pages, there is a data creation phase that happens before the aforementioned main stream of processes. The synthetic data is combined with the real data in a number of different training scenarios, highlighted in Chapter 4. Figure 3.1 highlights the preprocessing steps for preparing the real and synthetic data that is used as input for two distinct OMR evaluation pipelines: one using solely real input data and the other using both real and synthetic input data. Sections 3.2, 3.3, 3.4, and 3.5 cover the preprocessing steps, explaining how real ground truth data is extracted and used as input to a GAN architecture to synthesize glyphs and neumes which make up the synthetic manuscript images and ground truth data. Sections 3.6 and 3.7 discuss the object detection and position classification processes in the OMR pipelines that are evaluated and compared using metrics described in 3.8.

Figure 3.1: Overview of the preprocessing and OMR workflow implementations. The real ground truth data is input to a GAN to create the synthetic ground truth data. Then, two OMR pipelines, one with solely real data, and the other with real and synthetic data, are used to detect and classify the glyphs and neumes on the manuscript pages, and their evaluation metrics are compared.

Synthetic manuscript images are entirely generated based on the real manuscript images and transcription data. First, a GAN is used to create the individual neume components. It is trained on the set of the smallest building blocks of musical information in square notation: *punctum*, *inclinatum*, *custos*, *clef*, and *oblique* (Figure 3.2). This data for training is obtained from the ground truth manuscript MEI files from two Medieval manuscripts. After the synthetic

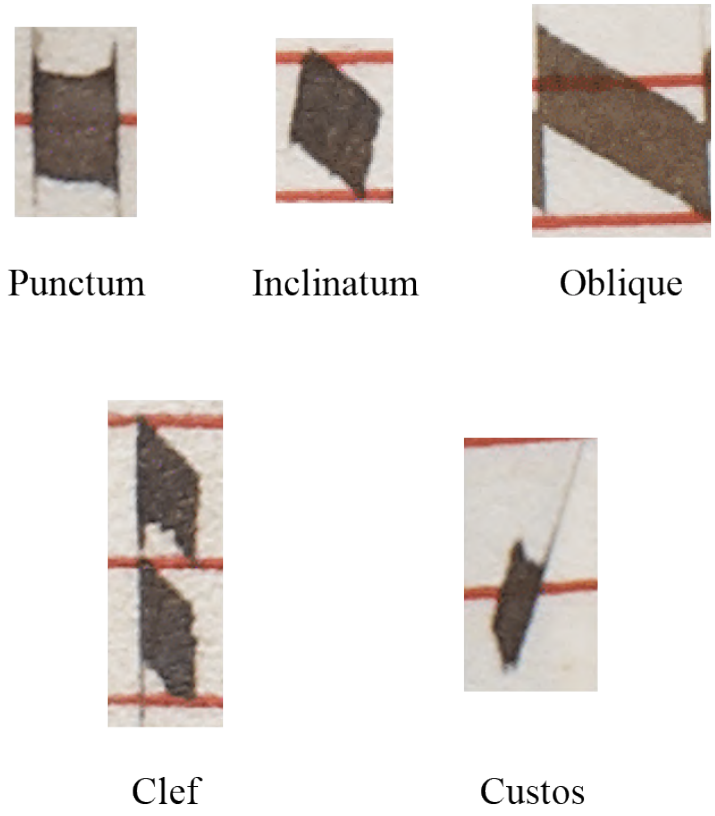

Figure 3.2: Neume component classes.

neume components are generated by the GAN, they are placed pseudo-randomly into the nine possible pitch positions on a staff in square notation. This is determined by assigned relative weights to each neume component type, controlling which appear more often than others. A full manuscript page is synthesized with neume components on staff lines, and non-musical features such as decorative text and lyrics are not included. The coordinates, type, staff position, and pitch of each neume component placed on the page are recorded in a corresponding text file, providing all the synthetic "ground truth" parameters necessary for training and evaluating the recognition workflow.

The main block of processes for the parallel OMR workflows consists of two distinct portions:

— Object Detection and Classification of Glyphs and Neumes

— Position and Pitch Classification of Glyphs and Neumes

First, the manuscript page is divided into smaller tiles to make the object detection operation more efficient and accurate. Motivated by Unel et al. (2019), a novel page partitioning algorithm separates each manuscript page into equal-sized tiles. Object detection is then performed on each tile, and the relative coordinates and type of each neume component detection

are recorded. The tiles are then stitched back together into the original manuscript page, and the relative coordinates for each neume component detection are re-established in relation to the overall original image. Every detected glyph and neume is then classified on position and later pitch, which is determined by the most recently occurring *clef*. In the following subsections, each of these core processes are further broken down and discussed.

3.2 Real vs Synthetic Manuscript Pages

The major hypothesis being tested in this thesis is whether a training dataset extended with synthetic manuscript data can outperform one comprised of only finite real data for the automatic recognition of square notation. In order to prepare a viable dataset, the synthetic manuscript pages need to encompass all of the pertinent information for performing automatic music symbol recognition found in the real manuscript data. The synthetic manuscript images must contain 4-line staves that fill the page from top to bottom with surrounding margins vertically and horizontally. Each staff will begin with a *clef*, end with a *custos*, and neume components will fill the space in between. Some of the neume components will be placed immediately next to one another, mimicking the presence of common compound neumes in real manuscript data (See Figure 2.4). These features encompass the necessary attributes that must be included in the synthetic manuscript pages for use in the automatic recognition workflow. In the next two sections, the two Medieval manuscripts used in this research are introduced.

3.2.1 Salzinnes Antiphonal

Produced in 1554 and 1555, the Salzinnes Antiphonal (See Figure 2.1) is a choir manuscript written in square notation, containing the music associated with Divine Office, a set of chants for blessing each day with prayer (Dietz 2006). It was commissioned by Dame Julienne de Glymes, prioress of the Cisterian Abbey of Salzinnes, Namur, in present day Belgium. The manuscript spans two volumes across 240 folios, or 480 pages in total, with 2932 chants. Each page measures 39.4 x 61.5 cm, and the digital scans are 4414 x 6993 pixels. Alongside the chants, there are a number of full-page illustrations depicting biblical and other historical scenes, in addition to full-length portraits of 34 nuns with their names and associated patrons' coat of arms. The manuscript, salvaged from the destruction of the Abbey in 1795 by the French Revolutionary Army, was likely in the possession of Bishop William Walsh, who brought it to

Canada as the first Archbishop for the Archdiocese of Halifax in the mid-nineteenth century. In 1975, it was donated to the Patrick Power Library at Saint Mary's University in Halifax, Nova Scotia.

3.2.2 Einsiedeln Codex 611(89)

The Einsiedeln Codex 611(89) manuscript (Figure 3.3), part of the Stiftsbibliothek collection at the Abbey Library of Saint Gall, Switzerland, was created in the fourteenth century, most likely prior to 1314. [12] It originated in Einsiedeln, Switzerland, and remained in use at the Benedictine Abbey of Einsiedeln until the 17th century. This manuscript, also written in square notation, contains chants framed in the anticipation of the arrival of certain important saints, including John the Baptist and Peter from biblical times. Four folios were added in the 16th century, bringing the total to 281 folios, across 562 pages. Each page measures 22 x 32 cm, and the digital scans are 4872 x 6496 pixels. The manuscript is digitally archived through the Virtual Manuscript Library of Switzerland. [13]

12. https://www.e-codices.unifr.ch/en/list/one/sbe/0611
13. See footnote 12

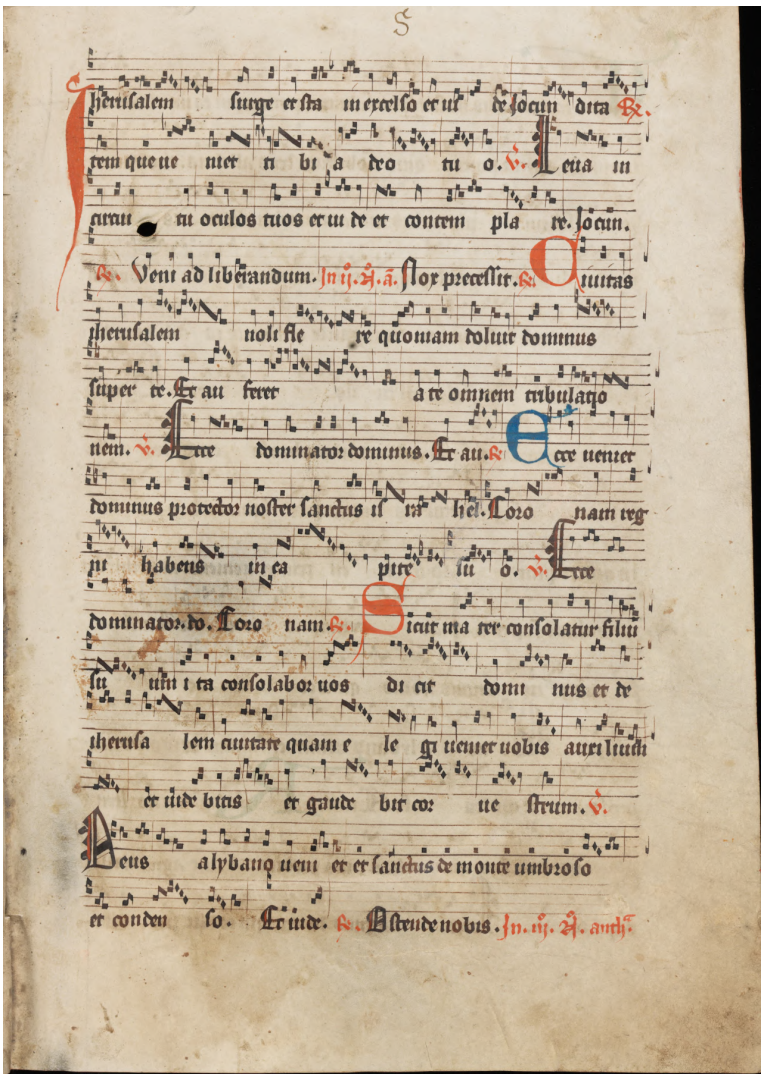

Figure 3.3: Page from the Einsiedeln Codex 611(89).

3.3 Extracting Existing Manuscript Information

Both of the Medieval manuscripts are filled with chants, and lyrics found under each staff accompany the musical information, sometimes expanding into large decorative text (Figure 3.4). The synthetic manuscript pages will not include the decorative and lyrical features from real manuscript images since the focus here is on the location of neume components on the page, which the GAN is solely trained on.

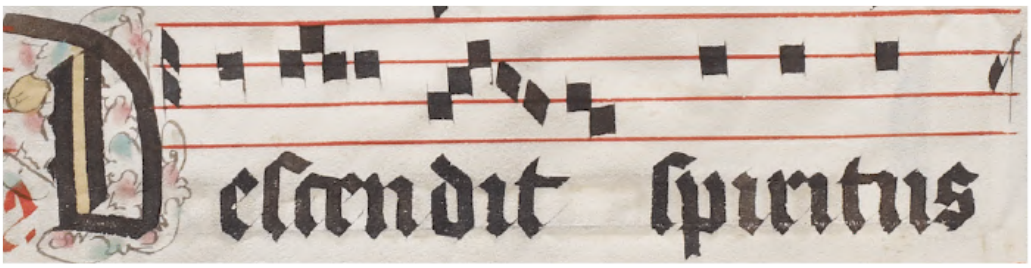

Figure 3.4: Decorative and lyrical text example in the Salzinnes Antiphonal.

In order to generate synthetic manuscript data and train object detection and position classification algorithms, ground truth MEI data needed to be created for both manuscripts. The data needed to have labels for every musical glyph in the manuscript image, including its coordinates, type of symbol, position in the staff, and musical pitch. All of the manuscript pages are digitally available as high-resolution scanned images, though none were encoded into verifiable ground truth data prior to this work. Neon, the online neume editor (Burlet et al. 2012, Regimbal et al. 2019), was used to create the ground truth MEI data for both manuscripts. Employing a web-based graphical user interface, Neon provides a useful set of tools for quickly annotating the square notation. Neon receives a manuscript image of square notation as input, in addition to an MEI-formatted file. The user selects general staff and neume component shapes from a sidebar editing panel and places the selection on the manuscript image by clicking on the respective target where a virtual overlay of the glyph is placed (Figure 3.5). Staff shapes can be skewed to align with the unaligned staves on the page, and the possible neume component positions on the staff are automatically bound to the placed staff shape. The skewing is necessary to align the virtual boundaries of the staff with the distorted staff shapes in the manuscript image, due to every page's bounding to the spine of the thick physical manuscript and the imperfectness of handwriting (Figure 3.6). When the annotation is complete, the data can be saved as an MEI file, which is parsed to select the necessary features for training synthesizing GAN images and training the automatic music recognition pipeline. With the use of Neon, annotated manuscript page data was created, edited, and verified for use in the training processes.

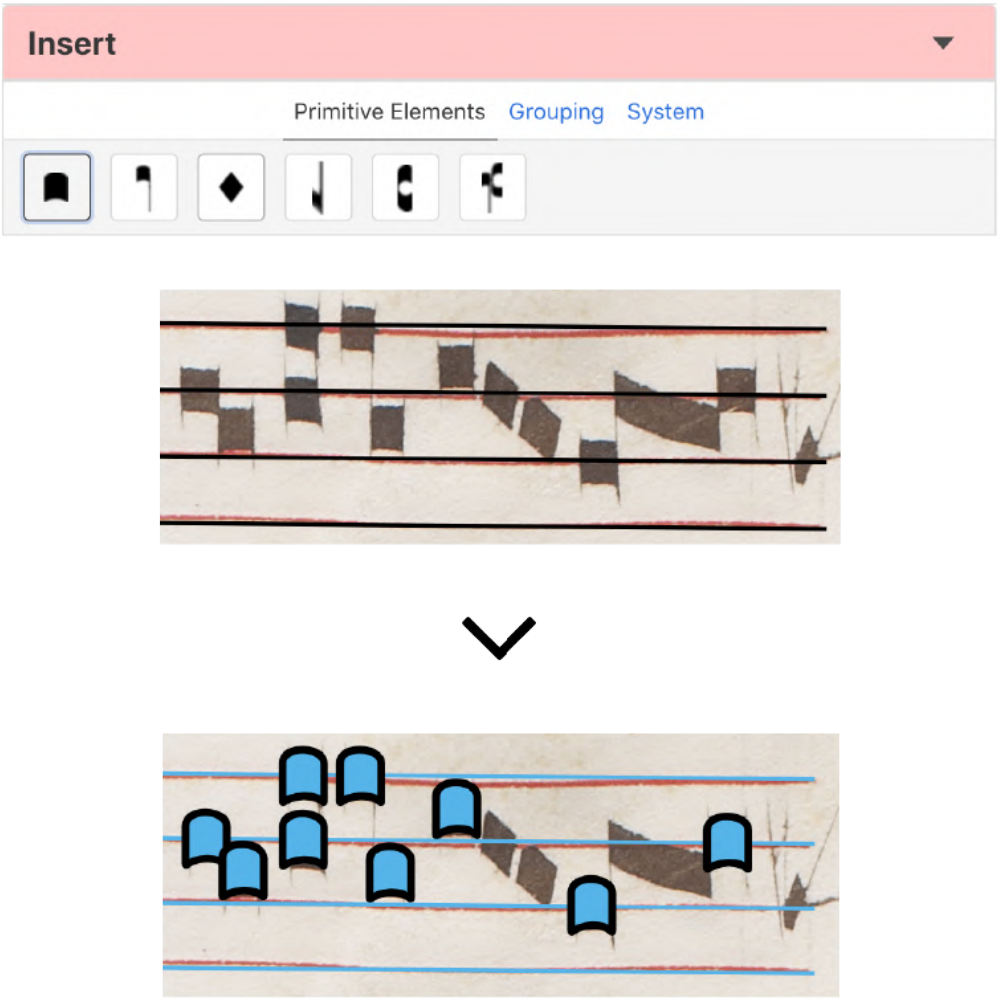

Figure 3.5: Annotation example of punctums in a small section of the Salzinnes Antiphonal using Neon's insert portion of the editing interface.

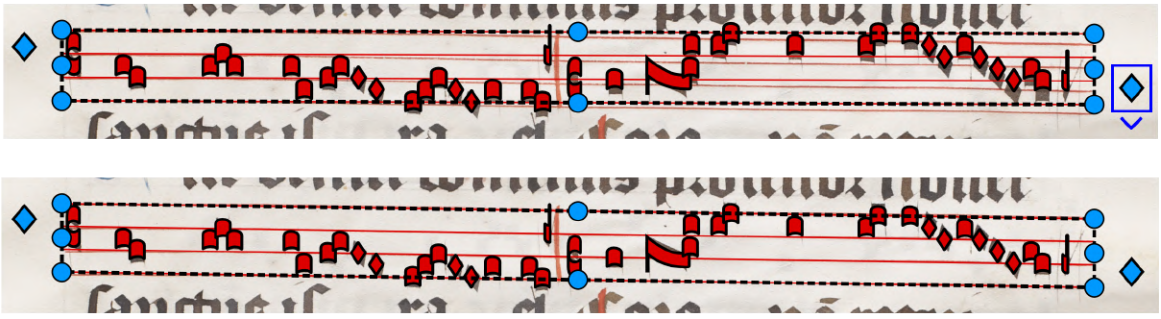

Figure 3.6: Staff skewing example in Neon. Moving the diamond-shaped icon on the right side aligns the annotated glyphs to the correct coordinates.

Prior to this research, the entirety of the Salzinnes manuscript had been processed with a pre-existing OMR workflow, though the output data had not been fully corrected or verified.

A brief visual evaluation made it clear that most pages contained a number of errors, so they needed to be edited for use in the ground truth dataset. Fortunately, the pre-existing data was already encoded in the MEI format, which required only editing in Neon instead of annotating from scratch. Twenty pages of the Salzinnes manuscript were collected and edited in Neon for use in the overall workflow.

The Einsiedeln manuscript did not have any prior annotation data encoded in MEI. Thus, pages for this manuscript needed to be annotated from scratch in Neon. This was a time-consuming process, since the average number of neume components and glyphs per page for Einsiedeln is around 500. This resulted in 10 pages of Einsiedeln for use alongside 20 pages of Salzinnes in the overall workflow.

3.4 Creating Synthetic Glyphs and Neumes

Square notation is comprised of many compound sequences of musical notes that are collectively referred to as neumes. Depending on the combination of neume components that make them up, they determine the overall classification for the neumes (see 2.3.1). The groupings of neume components are dependent on proximity and syllabic relationships to lyrical text. Without a strong theoretical understanding of square notation, it can be difficult to discern where one compound neume ends and the next begins. There are definitive neume sequences and compound neumes that can be identified in the manuscripts, though they do not encompass all of the possible combinations of neume components. Sequences of neumes often become too long and do not often feature the same ordering or composition of neume components. Due to their high variability, it is difficult to provide classifications that generalize well for all neume sequences. The classifiable compound neumes also appear infrequently, which does not provide many examples for training a GAN architecture. It would be more feasible to train a GAN with individual neume components, since they encompass a specific set of classifications. Hence, it was decided to only synthesize the neume components and glyphs that have definitive shapes and musical information: the *c-clef*, *f-clef*, *custos*, *inclinatum*, *oblique*, *punctum*, and *virga*. Most of these glyphs can be seen in Figures 2.2 and 3.7.

In order for the GAN to synthesize neume components, a set of cropped neume component images and corresponding type labels needed to be prepared. For training the object detection and position classification steps of the OMR workflow, the coordinates, position, and type

of each neume component also needed to be prepared. The 30 MEI files annotated in Neon encompassed the ground truth data used for training, verifying, and testing the entire workflow. The files contained the coordinates, type, position, and pitch of every neume component on each page. A Python script was composed to parse the MEI files, extracting the coordinates, type, staff position, and pitch for each neume component. The script receives a manuscript image and relevant MEI file as input. From the MEI file, the script records the aforementioned set of attributes for every glyph on the page in a comma-separated text file format. Using the coordinates, the script then locates the placement of every glyph on the page and saves a separate image cropped to the glyph within the coordinates. Each cropped image file name was listed alongside its type in another corresponding text file for use in GAN training.

When training a GAN image synthesis model, each input needs to be of the same dimensions. The cropped neume component images varied in their sizes, and a possible solution was resizing them to the same dimensions. Unfortunately, any image rescaling to a standard size reduces the number of overall pixels in the larger neume component cases such as *obliques* and *clefs*, which results in information loss. In order to avoid any resizing, a neume component padding approach was introduced for the creation of input images for the GAN (Figure 3.7). Starting from a white background image, every extracted neume component was vertically and horizontally centered on an individual background image to avoid the need for any resizing, as long as the height and width of the background image exceeded those dimensions in the largest of neume component cases. The standard background image size of 256 x 256 was chosen because the largest width and height of all the neume components found in the two manuscripts were 178 and 155 pixels, respectively. As intended, every neume component was then centered on the white background image. The padded images and corresponding neume component type labels were then used as the training dataset for the GAN architecture.

Figure 3.7: Padded neume component GAN training dataset examples. From left to right: *oblique3*, *virga*, and *clef.c*.

Although several open-source GAN architectures were tried, due mainly for efficiency, StyleGAN was chosen (Karras et al. 2018). Others tried included DCGAN (Radford et al. 2016) in Pytorch [14] and Torchfusion [15]. Both local and web-based programming environments such as Google Colab [16] and Kaggle [17] had enough hardware resources to compute a dataset of 128 x 128 images for these architectures, but not enough when scaling to 256 x 256, the chosen input dimensions for the GAN. An open-source version of StyleGAN was pursued in Tensorflow, [18] another popular machine learning library for Python. The public repository for StyleGAN [19] indicates how long the training process will take when using 256 x 256 input images and the GPU hardware memory requirements on the 70,000 image FFHQ dataset, [20] ensuring that it was feasible to train with available hardware, though at a cost for time, taking at most five days to complete. The neume component dataset contained 12,000 images, so it was likely that training time would be considerably reduced. This information made it possible to move forward with StyleGAN.

The base StyleGAN architecture is able to synthesize images with the same overall class labelling, such as a human portrait, but it does not come equipped with an option for training a multi-class supervised GAN approach, where one model could generate different classes based on a coinciding label. A multi-class training scenario was preferable to training individual GAN

14. `https://pytorch.org/`
15. `https://github.com/johnolafenwa/TorchFusion`
16. `https://colab.research.google.com/`
17. `https://www.kaggle.com/`
18. `https://www.tensorflow.org/`
19. `https://github.com/NVlabs/stylegan`
20. `https://github.com/NVlabs/ffhq-dataset`

architectures for each class to maintain overall training efficiency and reduce the amount of parallel processes needed to train the dataset partitions. StyleGAN includes dataset preparation configurations for class-based architectures such as the MNIST handwritten text dataset, and the code was augmented to prepare the custom neume component multi-class dataset in the Tensorflow format.

Compute Canada [21] is a high-performance computing system, remotely accessed for use in this work. The network is partitioned into a number of different computing clusters or nodes that encompass a number of use cases. Cedar, a general-purpose node, features a large set of powerful GPUs available for user access. GPUs are highly beneficial to machine learning operations involving images, and Compute Canada offers a number of possible configurations. The most powerful GPU available on Cedar, NVIDIA's V100 Volta, [22] provided sufficient memory (32 Gb) for the GAN generation portion of this project.

3.5 Creating Synthetic Manuscript Pages

The final pre-processing step before the OMR object detection and position classification operations was creating synthetic manuscript pages. The generated neume components from the GAN were placed onto a manuscript page, mimicking the structure of a real page of square notation. Omitting the decorative text and lyrical features, the neume components were placed on 4-line staves in one of the nine possible pitch positions for square notation (See Figure 2.6). The synthetic *clefs* and *custos* were also placed in their designated positions at the beginning and end of each staff respectively, culminating in the creation of synthetic full page manuscripts image for use in training the novel automatic music recognition workflow.

3.5.1 Staff Height Normalization

In order to mimic the real manuscript pages, the height of the staves on the synthetic manuscript pages needed to be normalized. Normalizing the height of the staves influenced the final sizing of the synthetic neume components on a page. In handwritten manuscripts, the height of each staff on a page tends to vary as well as the number of staves. In addition, the Salzinnes and Einsiedeln manuscript image scans used in this research are not of the same

21. `https://www.computecanada.ca/`
22. `https://docs.computecanada.ca/wiki/Cedar`

dimension. The size of neume components on a manuscript page is relative to the height of their respective staff, so the height of the space determines their relative size. For *punctums* and *inclinatums*, their height is approximately equal to the height of one staff space (See Figure 2.6), or one third of the entire staff height. Thus, by setting the synthetic manuscript staff space height to the average staff heights of the two manuscripts, the neume component sizing was also generally standardized. Between the two distinct handwriting styles in the manuscripts, there is still some deviation in neume component sizing since the Einsiedeln glyphs are slightly smaller, leaving more space in between staff lines. Measuring a sample of 5 Salzinnes and 5 Einsiedeln pages, the average staff space height was discovered to be 60.0 pixels. The average staff space heights for the individual manuscripts were 63.33 pixels for Salzinnes and 56.67 pixels for Einsiedeln. These average dimensions were used for normalizing neume component sizes when generating manuscript pages.

3.5.2 Page Generation

In order to generate the synthetic pages for training the OMR workflow, staff lines were first placed on a manuscript page then filled in with synthetic neume components. Staff lines for the two manuscripts were created in Adobe Illustrator by drawing four horizontal and parallel lines spaced vertically with the average staff height dimensions from Section 3.5.1. For the Salzinnes manuscript, the lines were colored red, and for Einsiedeln, they were colored gray. They were set to an arbitrary length, able to be later resized when placing on the manuscript page, since any horizontal or vertical stretching introduced little to no visible distortion. As mentioned previously, the GAN was trained to generate neume components centered in a 256 x 256 white background image. The white space surrounding the neume components needed to be removed before placement on the manuscript page, since it would cover the staff lines and nearby neume components, and it does not contain any information from the original real data. Using Python, each generated image was automatically cropped to remove the white space surrounding the neume component. The resulting images were then placed on the synthetic manuscript page.

As the final step in preparing synthetic manuscript pages for training, each page was populated with staves and synthetic neume components (Figure 3.8). The pages were populated with 12 to 15 individual staves for Salzinnes and 15 to 18 for Einsiedeln, the average number of staves in the pages from each manuscript. Every synthetic page had a 200 pixel top margin, 500 pixel bottom margin, and 200 pixel margins on the left and right. The staves

filled the remaining coordinates, deviating between 97% to 103% of the remaining page width. The margins and randomized staff widths were selected to mimic the outer structure of the real manuscript pages and the slight variability in the widths of each staff. For each staff, a blank staff image was placed on the page background. The staff image was three times the height of the normalized staff space height from 3.5.1, since there are three internal spaces per staff. Starting at the lower y-coordinates of the respective staff, the nine possible pitch positions on the staff were calculated by dividing the staff space height by two and incrementing by this value nine times until reaching the top of the staff. These nine y-coordinate values represented the vertical position placement options for neume components on each staff, which were randomly selected when placing *punctums*, *inclinatums*, and *custos*. *Clefs* and *obliques* individually span multiple vertical positions and were placed on a smaller subset of the possible locations. *Clefs* are vertically centered around staff lines, and they only appear at positions *l2*, *l3*, and *l4*. Obliques cover three distinct types: *oblique2*, *oblique3*, and *oblique4* which specify the amount of pitch positions spanned by the neume component. They were placed relative to their leftmost starting position, and the rightmost position must remain within the nine possible pitch positions. Thus, the *oblique2* was only placed in position *l1* and above, the *oblique3* only *s2* and above, and the *oblique4* only *l2* and above. This guaranteed the descendant right position would remain within the range of possible positions.

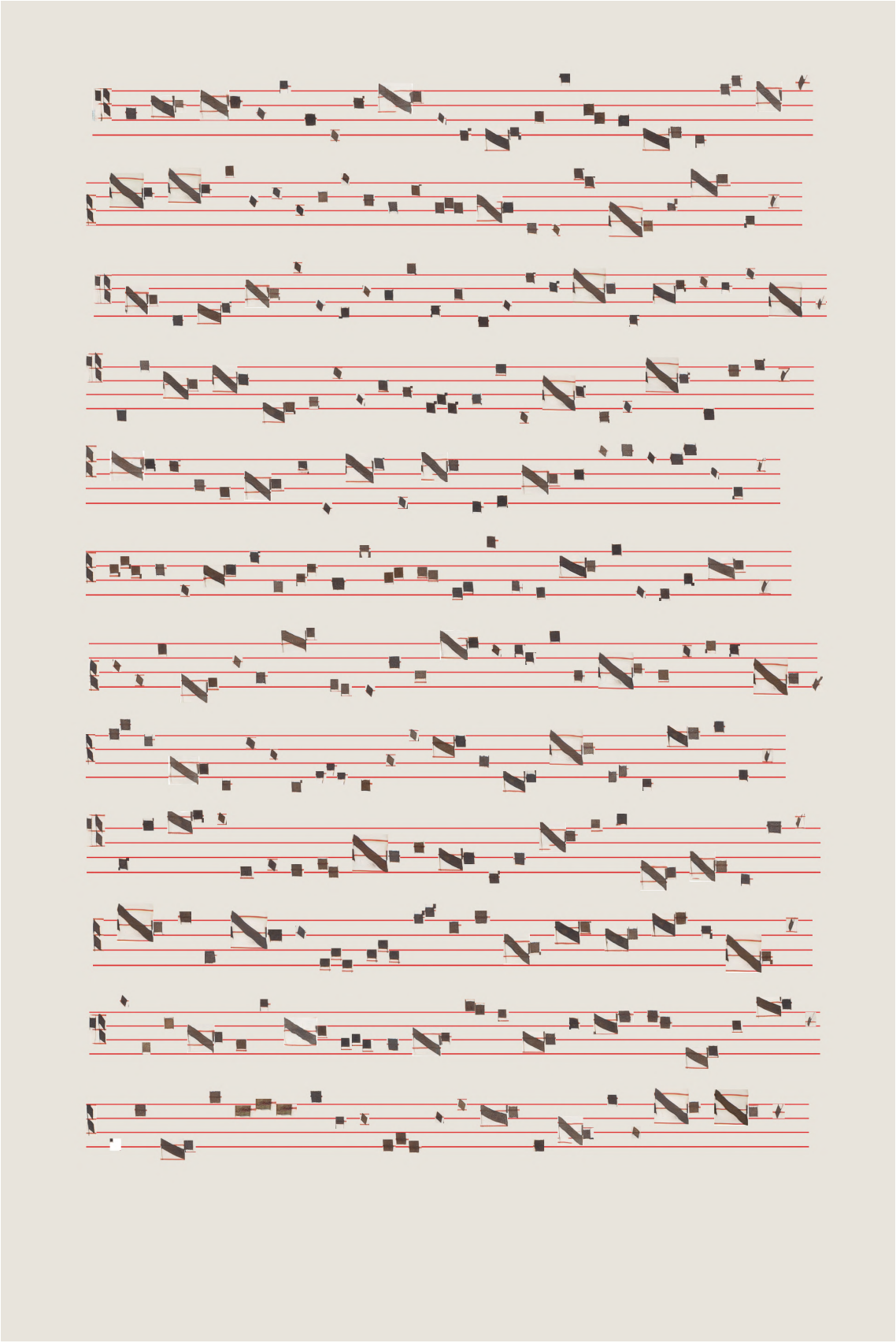

Figure 3.8: Synthetic Salzinnes page example.

On each staff, the first and last glyphs to appear are a *clef* and *custos*. In between them,

there are on average 25 synthetic neume components placed per staff on the page, based on the average occurrences of neume components per staff in the two manuscripts. This was randomly chosen in a range from 20 to 30 neume components per staff. They were spaced horizontally 40 to 60 pixels from one another, except when a compound neume was created. Neume components were instead placed immediately adjacent to one another to create compound neumes. An algorithm randomly selected a predetermined compound neume sequence and starting vertical position. The sub-process randomly occurred from 0 to 5 times per staff. For example, to create a *clivis*, two *punctums* were joined horizontally to one another with the second placed one position down from the first. A *torculus* was created by placing three *punctums* in a horizontal row, with the middle *punctum* one pitch position above the other two (Figure 3.9). Mentioned previously in 3.4, the object detection step in the OMR workflow was not trained to detect these compound neumes, though it was tasked with detecting the individual neume components that make up the sequenced neumes, which added another layer of realistic features to the synthetic data that exist in the real data. The object detection model was challenged to separately detect these closely connected neume components in both the synthetic and real manuscript images.

Figure 3.9: Synthetic clivis and torculus examples.

3.6 Object Detection of Glyphs and Neumes

Following the creation of the synthetic manuscript pages, both the real and generated images were then used as input to the automatic OMR workflow. The first step of this OMR process involved the detection of all neume components on a page of square notation. By breaking down the manuscript images into tiles, training a model to detect the neume components in each respective tile, and restitching the tiles into the original image, each glyph can be efficiently detected and classified in the overall scene.

3.6.1 Object Detection Algorithm

Initially, a machine learning model needed to be trained to detect neume components candidates on the manuscript pages. Based on the previous review of machine-assisted recognition of musical symbols in Section 2.4.2, the Faster R-CNN model (Ren et al. 2015) emerged as the optimal candidate for training this model. The model features shorter training times than other open-source detection frameworks with transfer learning (Metaj and Magnolfi 2019), performs well on datasets that do not have bunched objects (Pacha et al. 2018), and is conveniently embedded in PyTorch, the machine learning framework of choice for the OMR workflow. To train the Faster R-CNN model, every input image is provided alongside an annotation file. The annotation file specifies the coordinates for all of the neume components in the image and the class label.

3.6.2 Page Tiling

With available hardware, it would have been demanding to train an object detection model on full-size manuscript pages. Salzinnes manuscript pages are 4414 x 6993 pixels and contain hundreds of neume component candidates. Resizing a manuscript page would remove the details of small neume components, making the object detection task harder to train. Instead, the original resolution manuscript page can be partitioned into overlapping tiles of smaller-size images for training the object detection model. This considerably reduces the computational load of the training process and decreases the number of detection candidates in each input example.

In Python, a script was created to handle the tiling of manuscript pages for training the object detection model. The script had four input parameters: the number of tiles in the x and y direction and the amount of overlap in pixels respective to each dimension. If a bounding box for a neume component was cut off by a partition, then the object was not included for detection in the tile. As long as the x and y pixel overlaps exceeded the maximum width and height of any glyphs in the dataset, every glyph had the opportunity to be detected. By specifying 10 tile splits in the x and y directions, and 200 and 160 pixel overlap values respectively, every neume component was guaranteed to appear in at least one tile. These input parameters produced average tile sizes of 600 x 750 pixels (width x height).

The coordinates of each neume component needed to be re-established in the context of the

tile in which they resided. The coordinates of each tile were recorded, keeping track of their offset from the top-left corner of the original page. The tile coordinates were subtracted from the neume component coordinates, which justified the neume component coordinates relative to their respective tile. Since adjacent tiles from the original image are overlapping, the same neume component in the original page could appear in multiple tiles. If the object detection model detected the same glyph in multiple tiles, then a decision needed to be made for the final classification when connecting the tiles back together into the full original page. This was handled in the final restitching phase of the object detection step.

3.6.3 Restitching

After detecting the objects in each individual tile, the original manuscript page needed to be stitched back together before performing position and pitch classification on the neume components. It was preferred to reconnect all of the tiles, since the space above and below the detected neume components was necessary for position classification, and it was possibly cut off by the segmentation process. Using each tile's pre-recorded coordinates of its original position in the full page, the coordinates of each detected element on the tile were re-established in context to the top-left corner of the original manuscript image.

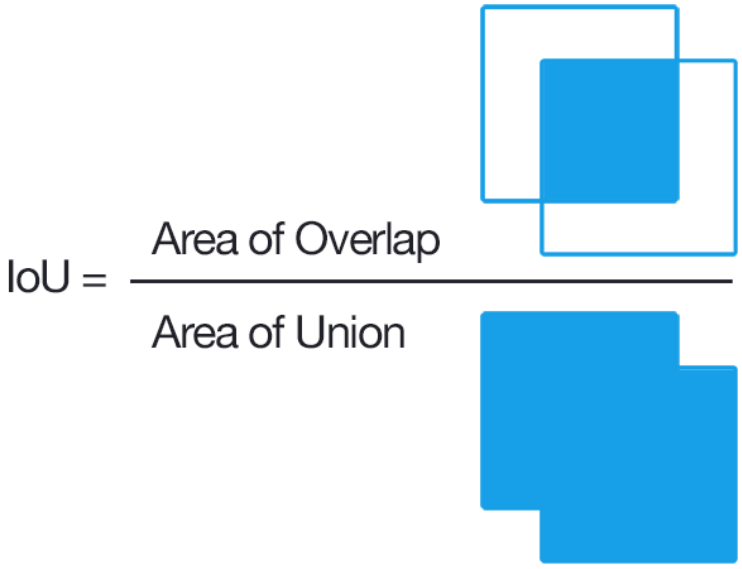

Figure 3.10: IoU visualization. [23]

In the event of overlapping detection candidates, a single prediction bounding box had to

23. https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

be selected. The detection boxes from each tile were combined into one large array per page, and an Intersection over Union (IoU) calculation was made to prune the final detection boxes based on the ground truth data. IoU (Figure 3.10) measures the overlap between the prediction and ground truth data boundaries by calculating a ratio of the area that both boundaries share (intersection) to the total area that both boundaries cover (union). The predicted detection with the highest IoU score and correct class label was considered the final detection in the original page. For example, if the object detection stage reported two instances of a *punctum* with approximately the same global page coordinates with IoU scores of 0.8 and 0.9 respectively, this resulted in a final classification of the second punctum with the higher score. Figure 3.11 shows an example representing multiple candidate pairs detected in adjacent overlapping tiles. Once all of the multiple candidates were pruned, the predicted coordinates of each neume component were passed to the position and final pitch classification tasks.

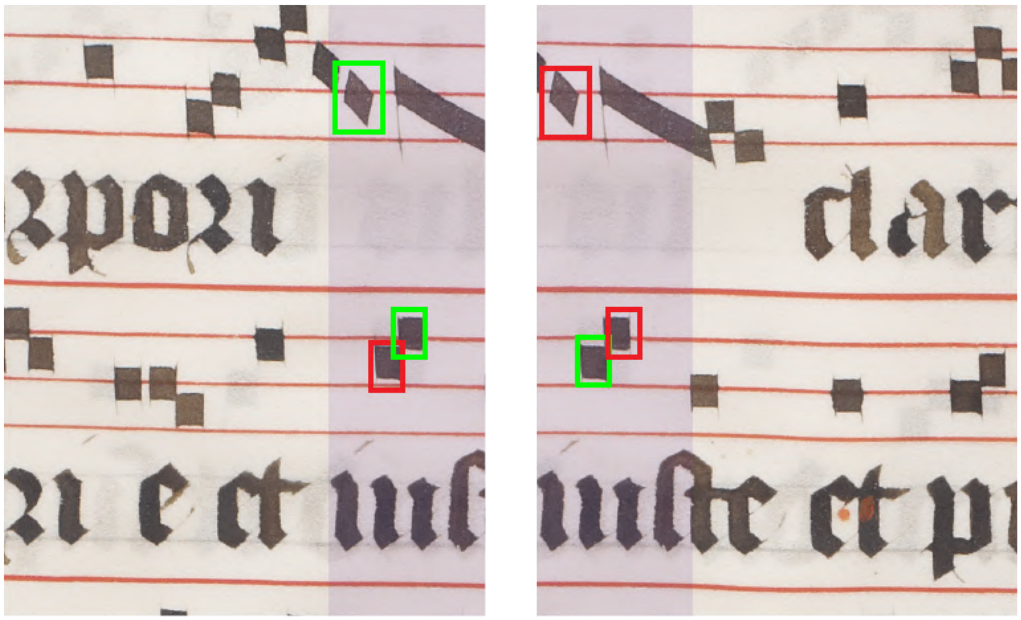

Figure 3.11: Two adjacent and overlapping tiles from the same region of the manuscript image. The neume components with bounding boxes represent the same neume component in the original manuscript image, and the darkened regions are the total overlap area between the two tiles. The green bounding boxes indicate which detection received the higher IoU score when restitching the page and comparing to the ground truth data and will be passed on to the next step of the OMR workflow.

3.7 Position and Pitch Classification

With all the neume components and glyphs detected on the page, they were then musically classified, first on their position on the staff, followed by their pitch relationship to the most recent *clef*. The position of a neume component is only known by determining its vertical placement relative to the staff lines. This involves identifying which of the nine staff positions a neume component is located. From the ground truth data, a position label was included for each neume component in the nine possible positions: *s1, l1, s2, l2, s3, l3, s4, l4, s5* (See Figure 2.3). Using the bounding box coordinates in the ground truth data, the height of the bounding boxes were extended with the detected neume components still in the center (Figure 3.12), cropped as individual images for the training set, and labelled on the centered neume components' staff positions. All bounding box heights were extended by two times the average staff space height above and below the vertical center of the detected neume component to ensure that enough of the staff lines were visible for classifying the position. For the *oblique* glyphs, the labels were assigned to the starting positions of their diagonal shapes, and for *clefs*, the labels were assigned to the staff line that they were centered on. Regardless of the neume component's position on the staff, this height extension included more staff lines in the bounding box, which needed to be visible for the following position classification task.

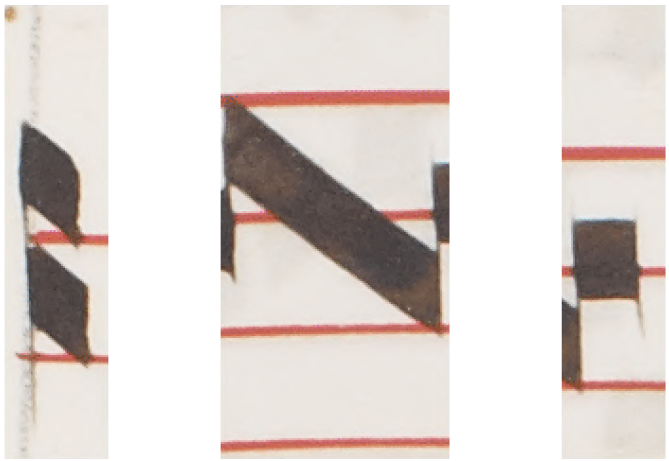

Figure 3.12: Extended bounding boxes for position classification model examples. From left to right: *clef.c* at l4 (classified on the pitch encoding at the vertical center), *oblique3* at s4, and *punctum* at l3.

A separate neural network from the object detector was trained to classify neume component positions on the staff. This was a ResNet 18 model (He et al. 2016) imported from TorchVision

within PyTorch, trained from scratch on the neume component dataset. The model expects inputs of 224 x 224 pixels, so the vertically extended bounding boxes were resized to this standard dimensionality. Once this classification was made, the final pitch classification could be heuristically established.

Finally, with the neume component's position classified, a pitch classification was performed. Neume component pitches are a function of both position on the staff and their relationship to the most recent *clef* occurrence, similar to modern musical notation. Before defining this relationship, all the neume components and glyphs on the page had to be organized in reading order, checking their coordinates first from top to bottom, then left to right. Neume components in the same staff appear at different heights, so their relationship to the nearest staff also has to be established to infer the correct "reading order." Iterating through every neume component and glyph on the page, the most recent occurring *clef* change needed to be consistently updated. The *clef* informed the algorithm of the staff line that the reference pitch was on, so every staff line and space then received a pitch encoding. Based on the position of every neume component, their pitch was finally labelled. With the coordinates, type, position, and pitch of every neume component classified on a manuscript page, this data can then be encoded into the MEI format.

3.8 Evaluation Metrics

The main evaluation being made in this research is comparing whether a training dataset comprised of both real and synthetic manuscript data leads to more accurate results than training with only real manuscript data. In order to make this comparison, metrics need to be used to measure the accuracy of the object detection and position classification models. For the object detection task, a mean average precision metric (mAP) will be used. The mAP is one of the most commonly used metrics in the field of object detection (Liu et al. 2020). mAP considers the impact of incorrect detections and defends against biases of simple accuracy metrics in unbalanced detection scenarios (e.g., when one object has many more occurrences than the others). To calculate mAP, precision and recall metrics are required. Precision is the ratio of correctly predicted positives to the total number of **predicted** positives, and recall is the ratio of correctly predicted positives to the total number of **actual** positives:

$$precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

$$recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

In any object detection scenario, it is likely that predicted bounding boxes do not consist of the exact same coordinate values as the ground truth data, and a spatial threshold value needs to be introduced to evaluate their correctness. As explained in Section 3.6.3, Intersection over Union (IoU) (Figure 3.10) measures the overlap between the prediction and ground truth data boundaries. Setting a threshold value for IoU between 0 and 1, a prediction is considered a true positive above this threshold or a false positive below it, which is used to make the precision and recall calculations for each image in the dataset. For each image, the metrics are sorted incrementally by increasing recall values and plotted against the corresponding precision values. Finally, the mAP metric is found by calculating the area under the resulting curve on this plot. The mAP will be calculated on a per class basis of the detected neume components.

For establishing an overall score across classes, an augmented weighted mAP (w-mAP) score will be used. In a typical w-mAP scenario, all of the class-specific mAP scores are multiplied by their total occurrences in the ground truth data, summed, and divided by the total number of object candidates. The augmented w-mAP metric will add more weight to the *oblique* neume components. A common evaluation factor of OMR tasks is the time spent manually correcting a workflow's annotations. This factor is included in the w-mAP metric to penalize the symbols that take more time to annotate manually, namely the *oblique*. In Neon, to annotate neume components besides the *oblique*, one clicks on the corresponding symbol on the editing panel, and then clicks where to place it in the manuscript (See Figure 3.5). Annotating an oblique involves eleven clicks, nine more than any other neume component, thus the oblique counts in the augmented w-mAP will be multiplied by 5.5 (11/2), and the total weight of all neume components will be updated accordingly. [24]

The following position classification task will also be evaluated. It will be evaluated on the set of correct, true positive detections passed on from the object classification task. For each image in this set, the precision and recall will be calculated for the predicted positions. Since this operation is assigning only one label to the overall image, the IoU threshold is not used,

24. In Neon, annotating an oblique involves selecting the *punctum* symbol, clicking twice for the start and ending positions, changing to the edit panel by neume, dragging over the two punctums, selecting "group neumes," changing to edit by neume component, dragging over the two punctums again, selecting "group neume components," dragging over the two once more, and finally selecting "toggle ligature."

and an F_1-Score will be calculated instead. F_1-scores factor in the weighting of the precision and recall values as one combined metric.

$$F_1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

The main evaluations listed above will be compared between training datasets comprised of different amounts of real and synthetic training data. The baseline dataset for comparison contains real data only. Other datasets contain the same amount of real data and are appended with synthetically generated manuscript data. The test data used for evaluation is not used in the training process. Furthermore, synthetic images are only generated using real pages that are not being tested. If the experiments using both real and synthetic image training datasets result in higher evaluation metrics than the baseline real dataset, then it will demonstrate that synthetic data generation is useful for improving the OMR of square notation.

The GAN performance, separate from the main OMR workflow, will be evaluated on its own. The Fréchet Inception Distance (FID) measures the similarity of GAN-synthesized images to the real images it is trained with (Heusel et al. 2017). Contrary to the Inception Score proposed by Salimans et al. (2016), the FID compares the statistics of the generated and real images, instead of solely considering the generated images on their own. Lower FID scores indicate more realistic reproductions. While training, the modified StyleGAN repository outputs separate model checkpoints and keeps track of the corresponding FID metric at each step. Thus, the model checkpoint with the lowest FID score will be selected for generating neume components to place on synthetic manuscript pages.

4 Experiments

In this chapter, the methodology described in Chapter 3 is evaluated through a number of different training scenarios. The main evaluation being made is whether the OMR of square notation can be improved by training with real and synthetic data as opposed to real data on its own. There are two main experimental configurations for making this comparison: both Medieval manuscripts will be evaluated separately with their own training models. In each configuration, the OMR workflow will be trained with the corresponding real data to establish a baseline accuracy of OMR in the object detection and position classification of neume components. These metrics will be compared against the same workflow trained with both real and GAN-synthesized manuscript data. An increase in the evaluation metrics when training with real and synthetic data demonstrates that the OMR of square notation is improved by training with GAN-synthesized manuscripts. The creation of the datasets is described in Section 4.1, followed by the experimental overview in Section 4.2. The results of the GAN generation step appear in Section 4.3, and the individual manuscript experimental results appear in Sections 4.4 and 4.5.

4.1 Dataset Creation

As stated previously, there are two distinct types of ground truth data prepared for use in the experiments: real annotated manuscript data and GAN-synthesized data, created from the real data. The real manuscript data consists of the manuscript images and corresponding annotations created in Neon (Regimbal et al. 2019). The Salzinnes manuscript had unverified annotation data available from a previous OMR workflow, which was reviewed and edited in Neon for use in this research. The Einsiedeln manuscript did not have any prior annotation data available, and thus its ground truth data was created from scratch. Once the real manuscript data was entirely annotated, it was used to create the GAN-synthetized manuscript data.

The first step to create the synthetic manuscript data is training the GAN with the real manuscript data. It is trained with the cropped and white-padded individual neume components from the real manuscript images with type labels assigned to each. Once training is complete, the desired number of pages to generate can be specified with a Python script, using the GAN to generate the individual neume components for placement on the page. The script also generates

a corresponding comma-separated text file, indicating the coordinates, type, staff position, and pitch of every synthetic neume component. These synthetic pages and corresponding text files are used as ground truth data along with the real data for training the object detection and position classification models.

The dataset preparation for the experiments requires a patchwork of encoding and processing steps, including the GAN architecture for generating synthetic data. The 30 manuscript MEI files annotated in Neon (20 Salzinnes, 10 Einsiedeln) are parsed into a comma-separated text file that includes information about each neume component on the real pages: their coordinates, type, staff position, and pitch. The coordinates and types of the neume components are used to extract the training dataset for the GAN, which includes the neume components as white-padded individual images that are each assigned a type label. The real manuscript images and text files are uploaded to Google Colab, where the object detection and staff position classification processes of the OMR workflow are run. A 5-fold cross validation is introduced, specifying which data will be used for training, testing, and validation in the GAN, object detection, and staff position classification models. Even though the GAN is separated from the main OMR workflow and is generating synthetic data, it would undermine the integrity of the experiments to train it with any neume components in the testing data, since generated examples would possibly appear too similar to those reserved for evaluation in the OMR workflow. Hence, the 5-fold cross validation is maintained throughout the entire workflow. Using the real data, the baseline metrics can be calculated, and GAN-synthesized pages will be included to establish the comparison metrics when training the OMR workflow with real and synthetic data.

4.2 Experimental Overview

The goal of the experiments is to detect the neume components on the manuscript pages and determine their position relative to the staff. This task consists of object detection and staff position classification models with distinct sets of real and synthetic data. The experiments are broken down into the comparison of real training data vs combined real and synthetic training data evaluation metrics. Experiment I is trained and evaluated with data from the Salzinnes manuscript, and Experiment II is identical with respect to the Einsiedeln manuscript. In each experiment, the baseline metric is established by training the workflow with real data only and evaluating with the testing data (Table 4.1). The workflow is retrained with the

same real data in addition to synthetic data, and the evaluation metrics are compared against the baseline values, using the same testing set. This process is performed as part of the 5-fold cross validation. For each manuscript fold, the respective GAN, object detection, and position classification models are all trained with the same manuscript data. An increase in the evaluation metrics demonstrates that the OMR of square notation is improved with the use of GAN-synthesized data. In the next sections, the results of the FID, w-mAP and F_1-scores (see Section 3.8 for the explanation of these metrics) for each manuscript configuration will be presented.

Table 4.1: General evaluation process of the individual experiments. Baseline metrics are established with the real manuscript data and compared against the metrics found in the combined real and synthetic data.

Training Dataset	**Object Detection**	**Staff Position Classification**	**Overall Metric**
Real manuscript data	Baseline mAP scores	Baseline F_1-scores	Baseline mAP x F_1
Real and GAN-synthesized manuscript data	Comparison mAP scores	Comparison F_1-scores	Comparison mAP x F_1

4.3 GAN Manuscript Data Synthesis

The preprocessing step for the main object detection and position classification workflow involves creating the synthetic GAN manuscript pages. For each manuscript, five separate GANs were trained to generate the neume component classes across the five folds, culminating in ten total GAN models. Using the modified StyleGAN codebase, each model was trained on Compute Canada using four Tesla V100 Volta GPUs. Model checkpoints were saved during training, about every 120 iterations, and a log indicated the FID scores at the respective model instances. Regardless of the fold or training data, there was significant instability in the training process of the GANs at around 8,500 training iterations. Up until this iteration, the FID score trend would decline, then jump to values equal or higher than when the training process began, around 360, indicating that features learned in previous training iterations were no longer being rendered in the synthetic images. The training process also saved image grids of

generated examples at each model checkpoint. When reaching the instability point, the images being generated were entirely black squares, reflecting the sudden change in FID scores. These were not usable model iterations for generating neume components, and the prior checkpoint with the lowest FID score was used. Early GAN training indicated this widespread instability, and to save on parallel model training time, Compute Canada resource requests were reduced to train the models to their most stable points, about 7,500 iterations, which took just over 24 hours to train per model. A plot of the FID score versus training iteration for the Salzinnes manuscript can be seen in Figure 4.1.

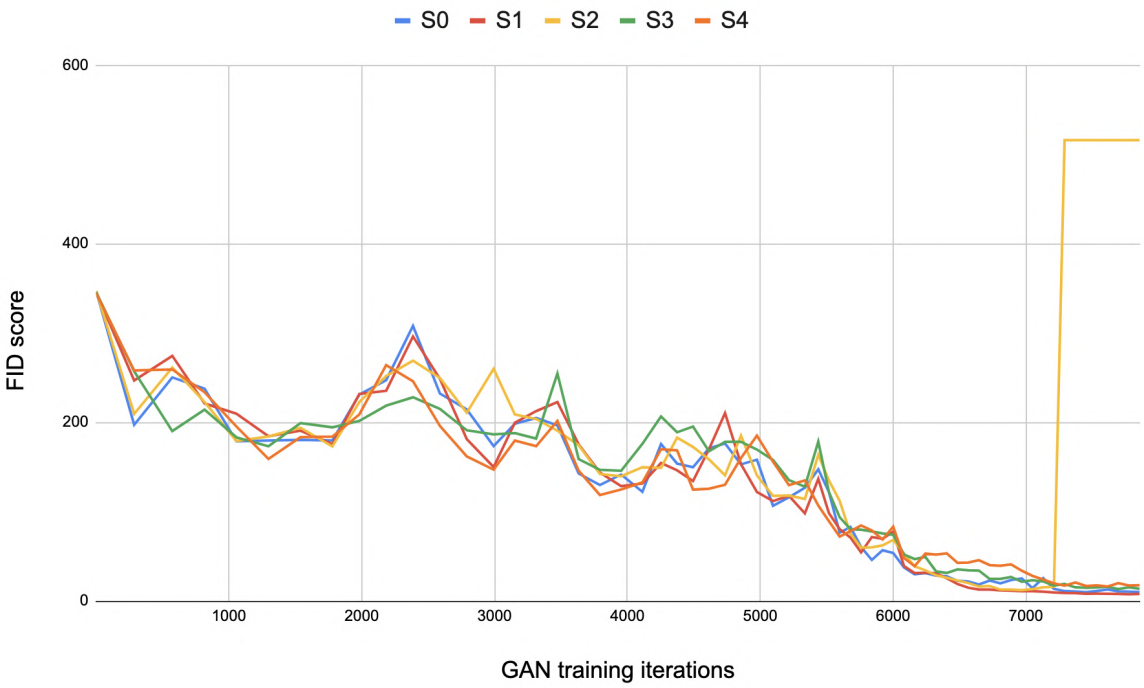

Figure 4.1: Plot of Salzinnes FID metrics across all five cross-validation splits. The jump in the S2 plot indicates the GAN's instability at 7,200 training iterations.

4.4 Experiment I

The Salzinnes manuscript dataset was comprised of 20 pages and split across five folds of 16, 2, and 2 pages respectively, an 80 (training), 10 (validating), 10 (testing) split out of 100. On the neume component level, this averaged out to an 80, 9.9, 10.1 split due to their differing amounts per manuscript page. The baseline metrics were established by training the object detection and position classification workflow with the real manuscript training data on

its own. To establish the comparison metrics, each fold of the manuscript training data was then appended with 16, 24, and 48 pages of GAN-synthesized ground truth data, expanding the amount of manuscript training pages by 100%, 150%, and 300% respectively. The overall average and weighted metrics are found in Table 4.2, including a combined mAP x F_1 metric for each training scenario. The two evaluation metrics are multiplied together since the candidates for the position classification task are dependent upon being located by the previous object detection task, and incorrectly or undetected neume components cannot be mapped to any coinciding ground truth data. The highest increase in the combined metric occurred in the 48 synthetic manuscript page case, leading to a 29.8% reduction in errors from the baseline. This is highly proportional to the reduction in errors of the object detection step, which yielded a 28.0% reduction in errors.

Table 4.2: Salzinnes manuscript w-mAP and F_1-scores at 0.5 IoU threshold.

	w-mAP (Object Detection)	**Weighted F_1-Score (Position Classification)**	**mAP x F_1**
Real Salzinnes Pages	0.950	0.993	0.943
Real + 16 Synthetic	0.963	0.996	0.959
Real + 24 Synthetic	0.964	0.992	0.956
Real + 48 Synthetic	0.964	0.996	0.960

Table 4.3 shows the average mAP scores per neume component class. The average amount of testing candidates per type class across all five folds were 37.6 (*clef.c*), 16.8 (*clef.f*), 40.8 (*custos*), 98.4 (*inclinatum*), 22.2 (*oblique2*), 6.6 (*oblique3*), 0.4 (*oblique4*), and 995 (*punctum*). The *punctum* far outnumbers the other types, encompassing 81.7% of all symbol candidates.

Table 4.3: Salzinnes mAP scores per neume component class at 0.5 IoU threshold.

	clef.c	**clef.f**	**cus**	**inc**	**ob2**	**ob3**	**ob4**	**punc**
Real	0.930	0.924	0.942	0.958	0.942	0.695	0	0.960
Real + 16 synth	0.920	0.926	0.936	0.960	0.980	0.893	1	0.965
Real + 24 synth	0.936	0.934	0.960	0.968	0.986	0.950	1	0.965
Real + 48 synth	0.948	0.958	0.958	0.968	0.968	0.988	1	0.968

Table 4.4 presents the F_1-Score results per staff position placement. The average occurences per staff position were 3.6 (s1), 23 (l1), 63.2 (s2), 121.8 (l2), 199.8 (s3), 161 (l3), 71.6 (s4), 47.8 (l4), and 4.4 (s5). The most common occurrences were in the middle staff positions.

Table 4.4: Salzinnes F_1 scores per staff position.

	s1	**l1**	**s2**	**l2**	**s3**	**l3**	**s4**	**l4**	**s5**
Real	0.975	0.995	0.998	0.994	0.996	0.994	0.990	0.984	0.921
Real + 16 synth	0.969	0.997	0.998	0.995	1	0.998	0.997	0.992	0.971
Real + 24 synth	0.985	0.993	0.997	0.987	0.997	0.990	0.996	0.987	1
Real + 48 synth	0.969	0.995	0.999	0.994	0.998	0.996	0.9965	1	1

4.5 Experiment II

Experiment II was identical to Experiment I, except the Einsiedeln manuscript was used instead of Salzinnes. The Einsiedeln manuscript dataset was comprised of 10 total pages and split across five folds of 8, 1, and 1 pages respectively. In the context of neume component totals, this averaged out to an 80, 10.1, and 9.9 split, again due to their various counts per manuscript page. The real manuscript data baseline metrics were compared against three synthetic page

scenarios, with 8, 12, and 24 additional synthetic manuscript pages respectively, marking 100%, 150%, and 300% increases in the amount of manuscript training data. The overall average and weighted metrics are found in Table 4.5, which includes the combined mAP x F_1 metric. The largest increase in the combined metric was found in the 8 synthetic page case, a 9.3% reduction in errors (0.925 to 0.932). Similar to Salzinnes, this increase was highly proportional to the performance of the object detection task, which saw an 8.5% reduction in errors.

Table 4.5: Einsiedeln manuscript w-mAP and F_1-scores at 0.5 IoU threshold.

	w-mAP (Object Detection)	**Weighted F_1-Score (Position Classification)**	**mAP x F_1**
Real Einsiedeln Pages	0.941	0.983	0.925
Real + 8 Synthetic	0.946	0.985	0.932
Real + 12 Synthetic	0.938	0.984	0.922
Real + 24 Synthetic	0.942	0.985	0.928

Table 4.6 displays the average mAP scores per neume component class. Across all five folds, the average number of candidates per class were 33.6 (*clef.c*), 1.4 (*clef.f*), 19.2 (*custos*), 108.6 (*inclinatum*), 24.4 (*oblique2*), 8.2 (*oblique3*), 2.2 (*oblique4*), 697.4 (*punctum*), and 303.4 (*virga*). Similarly to Salzinnes, the *punctum* is the most commonly occurring neume component, making up 56.6% of the training set, followed by the *virga* at 24.6%.

Table 4.6: Einsiedeln mAP scores per neume component class at 0.5 IoU threshold.

	clef.c	**clef.f**	**cus**	**inc**	**ob2**	**ob3**	**ob4**	**punc**	**virga**
Real	0.910	0.690	0.892	0.932	0.982	0.805	0.670	0.928	0.948
Real + 8 synth	0.894	0.953	0.876	0.902	0.978	0.933	0.900	0.946	0.948
Real + 12 synth	0.894	0.953	0.876	0.902	0.978	0.933	0.900	0.946	0.948
Real + 24 synth	0.908	0.730	0.884	0.924	0.974	0.948	0.88	0.946	0.930

Table 4.7 presents the F_1-Score results per staff position placement. The average occurences per staff position were 11 (s1), 35.4 (l1), 51.2 (s2), 112.8 (l2), 113.2 (s3), 142.8 (l3), 56.2 (s4), 57 (l4), and 10 (s5). Similarly to Salzinnes, the most common occurrences were in the middle staff positions.

Table 4.7: Einsiedeln F_1 scores per staff position.

	s1	**l1**	**s2**	**l2**	**s3**	**l3**	**s4**	**l4**	**s5**
Real	0.926	0.995	0.993	0.985	0.987	0.982	0.987	0.957	0.762
Real + 8 synth	0.947	0.993	0.993	0.985	0.983	0.990	0.975	0.971	0.955
Real + 12 synth	0.947	0.993	0.993	0.985	0.983	0.990	0.975	0.971	0.955
Real + 24 synth	0.921	0.985	0.991	0.983	0.989	0.988	0.990	0.970	0.954

4.6 Discussions

Generally speaking, the inclusion of GAN-synthesized data increased the overall accuracy of the object detection portion of the OMR workflow. In the Salzinnes manuscript, the w-mAP metric increased from the baseline value of 0.950 to 0.964 in the best case, a 28.0% reduction in errors when using 48 additional synthetic manuscript pages. Each individual neume component

class saw an increase in the mAP scores, most significantly among the *obliques*. The GAN-synthesized versions of each *oblique* class yielded significant increases in their respective mAPs, especially the *oblique3* and *oblique4*. In the case of the *oblique4*, it was totally undetected with only the real training data and successfully detected in both synthetic training data extensions for every occurrence. It should be noted that there were only two *oblique4* instances among the original (real) 20 Salzinnes pages. The *clef.c* and *custos* classes saw a slight reduction in their mAPs when trained with the real data and 16 synthetic pages, which conversely increased above the baseline metric when being trained with real data and 24 or 48 synthetic pages. *Punctums* and *inclinatums* saw little increase in their mAP score respectively, likely due to the large number of real examples already available to begin with. Their rectangular and diamond shapes are also among the simpler geometries of neume components.

In the Einsiedeln manuscript, the object detection w-mAP increases were less significant, though still yielded an 8.5% reduction in errors in the best case with real data and 8 synthetic manuscript pages. The mAP scores increased in the three synthetic data scenarios, though using 8 pages instead of 12 and 24 pages resulted in the highest w-mAP score. Per class, many of the mAP scores tended to suffer slightly but still saw significant increases in the *oblique3* and *oblique4* cases. The *clef.f* mAP score increased drastically in the 8 and 12 synthetic page test case to 95.3%, and dropped to 73% with 24 synthetic pages, just above the 69% baseline. On some of the Einsiedeln GAN training splits, the *clef.f* class did not achieve qualitatively accurate synthetic images, generating examples that were smeared and blurred more than any of the examples the GAN was provided with (an example is shown in Figure 4.2). The 24 synthetic pages populated with inaccurate *clef.f* reproductions dwarfed the amount of real examples, possibly leading to the lower mAP score. The *oblique2*, *oblique3*, and *punctum* classes were the only stable increases among the combined real and synthetic page training sets. There was a slight reduction in the mAP score of the *oblique2* in every synthetic dataset for Einsiedeln, and it was reduced from 0.982 to 0.974 in the worst case (24 synthetic pages).

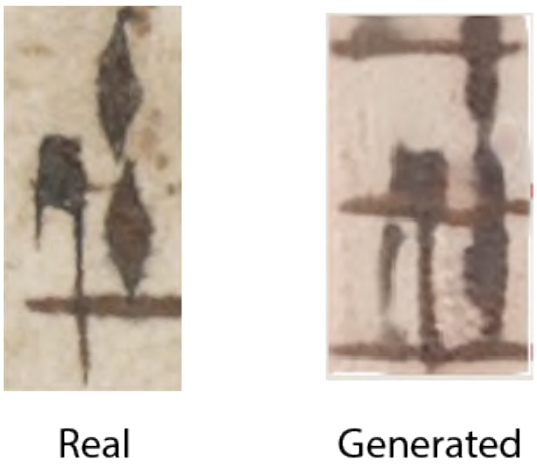

Figure 4.2: Poor Einsiedeln *clef.f* generation comparison to the real training data.

The position classification task was affected minimally by the inclusion of synthetic manuscript data. F_1-scores in the real manuscripts were already close to 100% across all position classes. The outermost stave positions, s1 and s5, above and below the stave respectively, saw the most adjustments in their metrics. The F_1-Score for position s5 increased to 100% when training Salzinnes with 24 synthetic pages and increased from 76.2% to 95.4% in the synthetic training scenarios with Einsiedeln. The F_1-scores were lower than the baseline in more classes when training with 24 synthetic pages in both manuscripts. Although these were minimal decreases, this was likely due to the real pages being outnumbered by the synthetic pages, where the placement of some neume components may have confused the model. Some of these synthetic neume component images were generated with the surrounding staff line fragments that were visible in the real training dataset (Figure 4.3). The GAN learns to generate the examples that are provided to it, and that included these line fragments. When placing the GAN images on the synthetic manuscript page, no heuristic was used to infer where a neume component should be placed based upon the intersecting staff fragments. Since the fragments do not always align with the overall staff lines on the page, the minimal decrease in F_1-scores could be a byproduct of their placement.

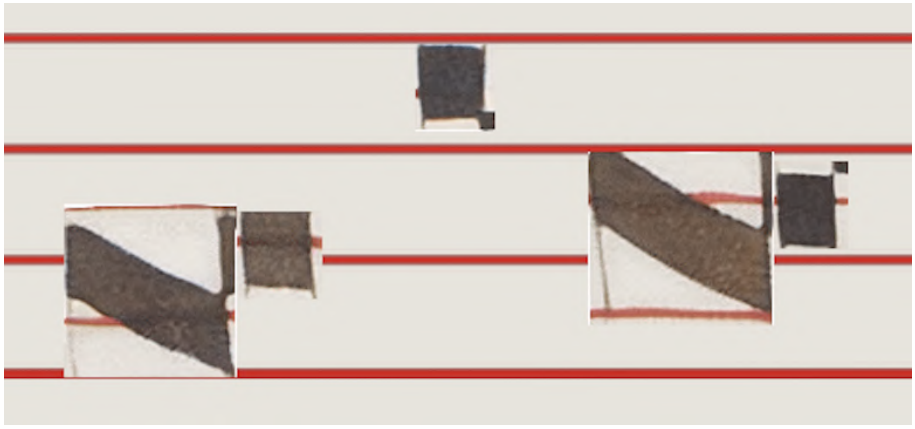

Figure 4.3: Example of truncated staff line neume component placement. The bounding box for each neume component includes generated staff line snippets that do not align with the underlying staff lines on the page.

The combined mAP x F_1 metric is highly dependent on the performance of the object detection task. The coordinates of the detected neume components are compared against the ground truth data at an IoU threshold of 0.5, which passes the highest IoU score among overlapping detections to the position classification task. Incorrectly detected or undetected neume components have no ground truth position data to link to, hence they are omitted from the position classification task. The combined metric improved in almost all synthetic manuscript cases, though it reduced slightly when training Einsiedeln with 12 synthetic manuscript pages. The object detection task suffered slightly in this scenario, the only time when w-mAP metric did not increase. Salzinnes saw the highest combined metric when training with the largest 48 synthetic page scenario, while Einsiedeln improved the most with the smallest 8 page addition.

5 Conclusions

By training a GAN to generate neume components and place them on synthetic manuscript pages, the OMR of square notation is improved for both of the Salzinnes and Einsiedeln manuscripts. In the object detection portion of the workflow, the errors were reduced by 28.0% and 8.5% respective to each manuscript. Considering that the average number of pages in these two manuscripts is over 500, using combined real and synthetic training datasets in an applied scenario has the opportunity to correctly detect a larger amount of neume components and reduce the amount of incorrect detections (false positives).

The addition of the synthetic training data significantly increased the w-mAP metrics for the *oblique* classes in both manuscripts. This is a valuable finding since the time spent correcting *obliques* far exceeds the amount of time for any other neume components in Neon. Requiring eleven clicks, nine more per occurrence, their annotation is weighted at 5.5. In Einsiedeln, the obliques outnumbered the *clef.c*, *clef.f* and *custos* classes and they were almost as common as the *clef.c* and *custos* classes in Salzinnes. Due to their rate of appearance, the time spent correcting annotations created by this OMR workflow would be considerably reduced. In the event that developments are made to Neon to shorten the time spent per *oblique* annotation, then it would retroactively reduce the impact that their more accurate mAP scores have in the overall augmented w-mAP score. With that in mind, the original w-mAP score still increased when using real and synthetic data, reducing errors by 18% in the best Salzinnes case with a 100% increase in size of the training set. The manual correction of *obliques* will still involve extra steps if improvements are made, since they vary in their width and height on the staff, requiring unique individual attention. The GAN, trained with the sparse *oblique4* class, still generated realistic results that improved the detections in the test sets on both manuscripts. Even if the GANs "memorized" these rare examples, the synthetic versions still generalized better to the testing data, resulting in significantly improved mAP scores. This finding suggests that rare symbols in the real training data should be proportionately more represented in the synthetic manuscript pages.

The position classification task was largely unaffected by the addition of synthetic manuscript data. F_1-scores were already near 1.0 when training with solely real data, leaving very little room for improvement. In Salzinnes, the weighted F_1-Score increased with the 100% larger real and synthetic dataset (0.993 to 0.996), slightly decreased at 150% (0.992), and increased again

at 300% (0.996). On Einsiedeln, the weighted F_1-Score minimally increased to 0.985 from the baseline score of 0.983 in the 100% and 300% training dataset increases.

Synthetic data generation, albeit rough, still improved the metrics for both processes in the OMR workflow. The GAN was trained to generate synthetic neume components that were not separated from the staff lines surrounding them in the original manuscript image. When placing the neume components on the page, they were not aligned into staff placements depending on where the truncated staff lines appeared in the generated image. For example, an *inclinatum* generated with a staff line through its center was not bound to placement on the four staff line positions, it could appear in any of the staff spaces, too. It would have required another processing method to either remove the staff lines from the neume components before training or after the results were generated from the GAN. Even though the synthetic pages had truncated staff lines surrounding the neume components, the position classification model did not suffer in the weighted F_1-Score. The outermost staff positions, *s1* and *s5*, saw the greatest F_1-Score improvements. In both manuscripts, the real data comprised of only a few occurrences, so increasing the number of neume components in these positions with synthetic data provided more valuable training data, even considering the generated set of truncated staff lines surrounding neume components placed in mismatching positions.

The GANs, each trained on different folds of the padded neume component images, varied in their FID metrics. There did not seem to be any correlation between lower FID scores and increased w-mAP metrics in the object detection step of the OMR workflow. Again, a lower FID score indicates that the generated examples are more realistic when comparing to the real examples than higher FID scores. The average FID metric for Einsiedeln was lower than Salzinnes, although Salzinnes had the greater reduction in errors when training with the synthetic data. The varying reproductions of the neume components generalized better to the test data in some folds, but not in others.

5.1 Future Work

The GAN and synthetic page generation preprocessing step is ripe for further inspection. One aspect of GANs that was not utilized in this thesis is latent space exploration (Bojanowski et al. 2018). The latent space is the input vector that the generator model receives to create synthetic examples. Usually comprised of completely randomized noise, the latent space vector

is the sort of genetic code for the generated image. With latent space exploration, these vectors are first qualitatively evaluated against their corresponding output examples, and examples are selected with the desired output features. New labels can be assigned to the output images with desirable features, and a simple logistic regression can be calculated to wrap around the generator model, progressively selecting new generated images with the desired qualities. Latent space exploration could be used to find generated neume component examples that do not include staff line fragments, making it easier to place them in any staff position on the synthetic manuscript page. Latent space exploration could also be used to find examples that include truncated staff lines either intersecting or surrounding the generated neume components. If generated staff lines intersect the neume component, then the example would be bound to only being placed on staff lines. Surrounding staff lines would indicate that the generated neume component should only be placed in one of the staff spaces. This experimentation could possibly improve the position classification task even further to a near perfect evaluation.

The amount of neume components per page and placement positions can be further explored. Rare neume components did not fare well in the baseline metrics, suggesting that more should be included in the synthetic manuscript data. The distribution of generated neume components mimicked the average totals for most classes in the real data, so more weight could be assigned to the under-represented classes such as the *obliques*. For staff position placements, one could experiment with only placing neume components in under-represented locations to create a more balanced dataset. The density of neume components on a single page could also be experimented with, to determine if increasing the density of neume components per staff increases difficulty on the object detection algorithm or not, requiring a fewer number of pages of synthetic training data to be generated.

The configurations for the testing and training data can be envisioned in a number of different scenarios. A transfer learning experiment could be conducted, training the OMR workflow with one model's data and evaluating with the other. In the context of the manuscripts used in this thesis, one scenario would involve training the GAN and OMR workflow with Salzinnes data and evaluating with Einsiedeln. Similarly, the baseline metrics from this transfer case would be compared against increasing training sets of real and synthetic manuscript data. If the evaluation metrics are close to those found in the singular manuscript training and testing scenarios performed in this thesis, then transfer learning could reveal some underlying features that exist between manuscripts, handwriting styles, and neume component geometries.

In order to directly evaluate the similarities that exist between different manuscripts, one could train the entire process with multiple manuscripts. For example, the GAN, object detection, and position classification models would be provided with training examples from both the Salzinnes and Einsiedeln manuscripts in a singular experiment. The GAN would then be generating neume component examples that exist in between the styles of the two manuscripts. The metrics from the singular manuscript OMR workflows could be compared against the metrics established by this multi-manuscript workflow, when both are evaluated individually with a singular manuscript's testing data. In the aforementioned example, this would involve using the same Salzinnes testing data in the singular and multi-manuscript training workflows and comparing their metrics. Similar to the transfer-learning idea, this could possibly reveal some underlying features that exist between different manuscripts.

One could also experiment with testing sets that contain more real manuscript data than the training set. The training set should still be larger than the test set, using synthetic manuscript pages to reach a viable ratio between the two. For example, one could train the workflow with 4 real and 16 synthetic pages and test with 5 real pages. Then, the amount of synthetic training data and real testing data could be successively increased, comparing the evaluation metrics of each scenario. Tests like this might infer that less time could be spent on manual annotation if the synthetic dataset increases coincided with higher evaluation metrics.

5.2 Contributions

This thesis represents a significant contribution as it provides the first study of using synthetic data generated by a GAN in any OMR processing workflow. The process envisioned is able to be replicated for any images of music notations that, similar to CWMN, require individual symbols to be located, classified, and encoded into a pitch. This workflow, created as part of the SIMSSA project, can be embedded into pre-existing document processing in the Rodan workflow engine and the SIMSSA database. As an applied tool, the creation of synthetic manuscript training data can possibly lead to greater OMR performance without any extra time spent on manual annotations. This thesis has shown that the OMR of square notation can be improved with the use of GAN-synthesized manuscript data.

Appendices

Appendix A Oblique Error Weighting

This appendix further explains the reason for applying a greater weight to the oblique classes in the w-mAP metric. In Neon, annotating an oblique involves selecting the *punctum* symbol, clicking twice for the start and ending positions, changing to the edit panel by neume, dragging over the two punctums, selecting "group neumes," changing to edit by neume component, dragging over the two punctums again, selecting "group neume components," dragging over the two once more, and finally selecting "toggle ligature." This process involves eleven clicks, nine more than any other neume component, thus the oblique counts in the augmented w-mAP are multiplied by 5.5 (11/2).

References

Antoniou, A., A. Storkey, and H. Edwards. 2018. Data Augmentation Generative Adversarial Networks. *arXiv:1711.04340 [cs, stat]*. arXiv: 1711.04340.

Arjovsky, M., S. Chintala, and L. Bottou. 2017. Wasserstein GAN. *ArXiv* abs/1701.07875. arXiv: 1704.00028.

Baird, H. S. 1990. Document Image Defect Models. In H. S. Baird, H. Bunke, and K. Yamamoto (Eds.), *Structured Document Image Analysis*, 38–46.

Baró, A., P. Riba, and A. Fornés. 2016. Towards the Recognition of Compound Music Notes in Handwritten Music Scores. In *Proceedings of the 15th International Conference on Frontiers in Handwriting Recognition*, Oct 23–26; Shenzhen, China, 465–470.

Bojanowski, P., A. Joulin, D. Lopez-Pas, and A. Szlam. 2018. Optimizing the Latent Space of Generative Networks. In J. Dy and A. Krause (Eds.), *Proceedings of Machine Learning Research*, Volume 80, July 10–15; Stockholm, Sweden, 600–609.

Bowles, C., L. Chen, R. Guerrero, P. Bentley, R. Gunn, A. Hammers, D. A. Dickie, M. V. Hernández, J. Wardlaw, and D. Rueckert. 2018. GAN Augmentation: Augmenting Training Data using Generative Adversarial Networks. *arXiv:1810.10863 [cs]*. arXiv: 1810.10863.

Buczak, A. L., and E. Guven. 2016. A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. *IEEE Communications Surveys Tutorials* 18 (2): 1153–1176.

Burlet, G., A. Porter, A. Hankinson, and I. Fujinaga. 2012. Neon.js: Neume Editor Online. In *Proceedings of the 13th International Society for Music Information Retrieval Conference*, Oct 8–12; Porto, Portugal, 121–126.

Calvo-Zaragoza, J., F. J. Castellanos, G. Vigliensoni, and I. Fujinaga. 2018. Deep Neural Networks for Document Processing of Music Score Images. *Applied Sciences* 8 (5): 654.

Calvo-Zaragoza, J., A. H. Toselli, and E. Vidal. 2019a. Handwritten Music Recognition for Mensural Notation with Convolutional Recurrent Neural Networks. *Pattern Recognition Letters* 128: 115–121.

Calvo-Zaragoza, J., A. H. Toselli, and E. Vidal. 2019b. Hybrid Hidden Markov Models and Artificial Neural Networks for Handwritten Music Recognition in Mensural Notation. *Pattern Analysis and Applications* 22: 1573–1584.

Denton, E. L., S. Chintala, A. Szlam, and R. Fergus. 2015. Deep Generative Image Models Using a Laplacian Pyramid of Adversarial Networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (Eds.), *Advances in Neural Information Processing Systems 28*, 1486–1494.

Dietz, J. 2006. Centuries of Silence: The Discovery of the Salzinnes Antiphonal. Master's thesis, Saint Mary's University.

Everingham, M., L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. 2007. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. `https://www.researchgate.net/publication/277292831_The_2005_pascal_visual_object_classes_challenge`.

Fornés, A., A. Dutta, A. Gordo, and J. Lladós. 2012. CVC-MUSCIMA: A Ground-truth of Handwritten Music Score Images for Writer Identification and Staff Removal. *International Journal on Document Analysis and Recognition* 15 (3): 243–251.

Gatys, L. A., A. S. Ecker, and M. Bethge. 2016. Image Style Transfer Using Convolutional Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 26 – July 1; Las Vegas, NV, 2414–2423.

Girshick, R. 2015. Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Dec 13–16; Santiago, Chile, 1440–1448.

Girshick, R., J. Donahue, T. Darrell, and J. Malik. 2014. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 24–27; Columbus, OH, 580–587.

Goodfellow, I. 2017. NIPS 2016 Tutorial: Generative Adversarial Networks. *arXiv:1701.00160 [cs]*. arXiv: 1701.00160.

Goodfellow, I., Y. Bengio, and A. Courville. 2016. *Deep Learning*. MIT Press.

Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. 2014. Generative Adversarial Nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 27*, 2672–2680. Curran Associates, Inc.

Gulrajani, I., F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. 2017. Improved Training of Wasserstein GANs. *ArXiv* abs/1704.00028.

Hajič jr., J., and P. Pecina. 2017. The MUSCIMA++ Dataset for Handwritten Optical Music Recognition. In *Proceedings of the 14th International Conference on Document Analysis and Recognition*, Nov 9–12; Kyoto, Japan, 39–46. ISSN: 2379-2140.

Hankinson, A., J. A. Burgoyne, G. Vigliensoni, and I. Fujinaga. 2012. Creating a Large-scale Searchable Digital Collection from Printed Music Materials. In *Proceedings of the 21st International Conference on World Wide Web*, April 16–20; Lyon, France, 903–908. ACM.

Hankinson, A., P. Roland, and I. Fujinaga. 2011. The Music Encoding Initiative as a Document-Encoding Framework. In *Proceedings of the 12th International Society for Music Information Retrieval Conference*, Oct 24–28; Miami, FL, 293–298.

He, K., X. Zhang, S. Ren, and J. Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 26 – July 1; Las Vegas, NV, 770–778.

Heusel, M., H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. 2017. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), *Advances in Neural Information Processing Systems 30*, 6626–6637.

Huang, Z., X. Jia, and Y. Guo. 2019. State-of-the-Art Model for Music Object Recognition with Deep Learning. *Applied Sciences* 9 (13): 2645–2665.

Karras, T., S. Laine, and T. Aila. 2018. A Style-Based Generator Architecture for Generative Adversarial Networks. *ArXiv* abs/1812.04948. arXiv: 1812.04948.

Kourou, K., T. P. Exarchos, K. P. Exarchos, M. V. Karamouzis, and D. I. Fotiadis. 2015. Machine Learning Applications in Cancer Prognosis and Prediction. *Computational and Structural Biotechnology Journal* 13: 8–17.

LeCun, Y., Y. Bengio, and G. Hinton. 2015. Deep Learning. *Nature* 521 (7553): 436–444.

LeCun, Y., B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. 1989. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation* 1 (4): 541–551.

Lin, T.-Y., P. Goyal, R. Girshick, K. He, and P. Dollár. 2017. Focal Loss for Dense Object Detection. In *Proceedings of the IEEE International Conference on Computer Vision*, Oct 22–29; Venice, Italy, 2980–2988.

Liu, L., W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen. 2020. Deep Learning for Generic Object Detection: A Survey. *International Journal of Computer Vision* 128 (2): 261–318.

Liu, Z., P. Luo, X. Wang, and X. Tang. 2015. Deep Learning Face Attributes in the Wild. In *Proceedings of the International Conference on Computer Vision (ICCV)*, Dec 7–13; Santiago, Chile.

Long, J., E. Shelhamer, and T. Darrell. 2015. Fully Convolutional Networks for Semantic Segmentation. *arXiv:1411.4038 [cs]*. arXiv: 1411.4038.

Metaj, S., and F. Magnolfi. 2019. MNR: MUSCIMA Notes Recognition. Using Faster R-CNN on HandwrittenMusic Dataset. Master's thesis, Politecnico di Milano.

Miyao, H., and M. Okamoto. 2004. Stave Extraction for Printed Music Scores Using DP Matching. *Journal of Advanced Computational Intelligence and Intelligent Informatics* 8: 208–215.

Oeldorf, C., and G. Spanakis. 2019. LoGANv2: Conditional Style-Based Logo Generation with Generative Adversarial Networks. In *Proceedings of the 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, Dec 16–19; Boca Raton, FL, 462–468.

Öngün, C., and A. Temizel. 2018. Paired 3D Model Generation with Conditional Generative Adversarial Networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, Sep 8–14; Munich, Germany, 473–487.

Pacha, A., and J. Calvo-Zaragoza. 2018. Optical Music Recognition in Mensural Notation with Region-Based Convolutional Neural Networks. In *Proceedings of the 19th International Society for Music Information Retrieval Conference*, Sep 23–27; Paris, France, 240–247.

Pacha, A., K.-Y. Choi, B. Coüasnon, Y. Ricquebourg, R. Zanibbi, and H. Eidenberger. 2018. Handwritten Music Object Detection: Open Issues and Baseline Results. In *Proceedings of the 13th International Workshop on Document Analysis Systems*, Apr 24–27; Vienna, Austria, 163–168.

Pacha, A., J. Hajič jr., and J. Calvo-Zaragoza. 2018. A Baseline for General Music Object Detection with Deep Learning. *Applied Sciences* 8 (9): 1488–1508.

Papageorgiou, C., and T. Poggio. 2000. A Trainable System for Object Detection. *International Journal of Computer Vision* 38 (1): 15–33.

Plastiras, G., C. Kyrkou, and T. Theocharides. 2018. Efficient ConvNet-based Object Detection for Unmanned Aerial Vehicles by Selective Tile Processing. In *Proceedings of the 12th International Conference on Distributed Smart Cameras*, Sep 3–4; Eindhoven, Netherlands.

Radford, A., L. Metz, and S. Chintala. 2016. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In *Proceedings of the 4th International Conference on Learning Representations*, May 2–4; San Juan, Puerto Rico.

Ramirez, C., and J. Ohya. 2014. Automatic Recognition of Square Notation Symbols in Western Plainchant Manuscripts. *Journal of New Music Research* 43 (4): 390–399. Publisher: Taylor and Francis Ltd.

Rebelo, A., I. Fujinaga, F. Paszkiewicz, A. Marçal, C. Guedes, and J. Cardoso. 2012. Optical Music Recognition: State-of-the-art and Open Issues. *International Journal of Multimedia Information Retrieval* 1 (3): 173–190.

Redmon, J., and A. Farhadi. 2018. Yolov3: An Incremental Improvement. *arXiv preprint arXiv:1804.02767*.

Regimbal, J., M. Zoé, G. Vigliensoni, A. Tran, and I. Fujinaga. 2019. Neon2: A Verovio-based Square-Notation Editor. Presented at the *Music Encoding Conference 2019*, May 29 – June 1; Vienna, Austria. `https://music-encoding.org/conference/2019/abstracts_mec2019/Neon2.pdf`.

Ren, S., K. He, R. Girshick, and J. Sun. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (Eds.), *Advances in Neural Information Processing Systems 28*, 91–99. Red Hook, NY: Curran Associates, Inc.

Rezende, D. J., S. Mohamed, and D. Wierstra. 2014. Stochastic Back-propagation and Variational Inference in Deep Latent Gaussian Models. *ArXiv* abs/1401.4082.

Ronneberger, O., P. Fischer, and T. Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi (Eds.), *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, Cham, Switzerland, 234–241. Springer International Publishing.

Salimans, T., I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. 2016. Improved Techniques for Training GANs. In *Advances in Neural Information Processing Systems 29*, 2234–2242. arXiv: 1606.03498.

Shmelkov, K., C. Schmid, and K. Alahari. 2018. How Good is My GAN? In *Proceedings of the European Conference on Computer Vision*, Volume 11206 of *Lecture Notes in Computer Science*, Sep 8–14; Munich, Germany, 218–234.

Simard, P., D. Steinkraus, and J. Platt. 2003. Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, Aug 3–6; Edinburgh, Scotland, 958–963.

Simonyan, K., and A. Zisserman. 2014. Very Deep Convolutional Networks for Large-scale

Image Recognition. *arXiv preprint arXiv:1409.1556*.

Sixt, L., B. Wild, and T. Landgraf. 2018. RenderGAN: Generating Realistic Labeled Data. *Frontiers in Robotics and AI* 5: 66.

Szegedy, C., S. Ioffe, and V. Vanhoucke. 2017. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, Feb 4–9; San Francisco, CA, 4278–4284.

Tuggener, L., I. Elezi, J. Schmidhuber, M. Pelillo, and T. Stadelmann. 2018. DeepScores: A Dataset for Segmentation, Detection and Classification of Tiny Objects. In *Proceedings of the 24th International Conference on Pattern Recognition*, Aug 20–24; Beijing, China.

Uijlings, J. R., K. E. Van De Sande, T. Gevers, and A. W. Smeulders. 2013. Selective Search for Object Recognition. *International Journal of Computer Vision* 104 (2): 154–171.

Unel, O. F., B. O. Ozkalayci, and C. Cigla. 2019. The Power of Tiling for Small Object Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, June 16–20; Long Beach, CA, 582–591.

van der Wel, E., and K. Ullrich. 2017. Optical Music Recognition with Convolutional Sequence-to-Sequence Models. *arXiv preprint arXiv:1707.04877*.

Vigliensoni, G., J. A. Burgoyne, A. Hankinson, and I. Fujinaga. 2011. Automatic Pitch Recognition in Printed Square-Note Notation. In *Proceedings of the 12th International Society for Music Information Retrieval Conference*, Oct 24–28; Miami, FL, 423–428.

Wang, T.-C., M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro. 2018. High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs. In *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 18–23; Salt Lake City, UT, 8798–8807.

Wick, C., A. Hartelt, and F. Puppe. 2019. Staff, Symbol and Melody Detection of Medieval Manuscripts Written in Square Notation Using Deep Fully Convolutional Networks. *Applied Sciences* 9 (13): 2646–2673.

Wick, C., and F. Puppe. 2019. OMMR4all: a Semiautomatic Online Editor for Medieval Music Notations. In J. Calvo-Zaragoza and A. Pacha (Eds.), *Proceedings of the 2nd International Workshop on Reading Music Systems*, Nov 2; Delft, The Netherlands, 31–34.