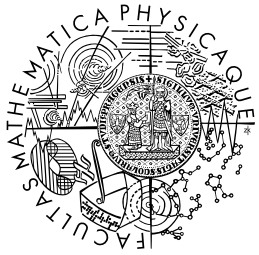


FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Jiří Mayer

Semi-supervised Learning in Optical Music Recognition

Institute of Formal and Applied Linguistics

Supervisor of the master thesis: doc. RNDr. Pavel Pecina, Ph.D.

Study programme: Computer Science - Software and Data Engineering

Prague 2022

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

I would like to thank my supervisor, doc. RNDr. Pavel Pecina, Ph.D., who has been supporting my research aspirations for the past three years. It is thanks to his guidance and valuable insights, that I am looking forward to my doctoral studies.

Huge thanks go to my fiancée. She has been very kind and selfless when I worked on this thesis and did not have time for daily chores. The same should also be said about my family – they are the reason I can study computer science.

Lastly, I want to thank my friend Michal. Our weekly discussions about this work have helped me shape it into its current form.

Title: Semi-supervised Learning in Optical Music Recognition

Author: Jiří Mayer

Department: Institute of Formal and Applied Linguistics

Supervisor: doc. RNDr. Pavel Pecina, Ph.D., Institute of Formal and Applied Linguistics

Abstract: Optical music recognition (OMR) is a niche subfield of computer vision, where some labeled datasets exist, but there is an order of magnitude more unlabeled data available. Recent advances in the field happened largely thanks to the adoption of deep learning. However, such neural networks are trained using labeled data only. Semi-supervised learning is a set of techniques that aim to incorporate unlabeled data during training to produce more capable models. We have modified a state-of-the-art object detection architecture and designed a semi-supervised training scheme to utilize unlabeled data. These modifications have successfully allowed us to train the architecture in an unsupervised setting, and our semi-supervised experiments indicate improvements to training stability and reduced overfitting.

Keywords: optical music recognition, semi-supervised learning, deep neural network

Contents

1. Introduction

Optical music recognition (OMR) is the task of extracting semantic information from scans and photos of sheet music. It is a subfield of computer vision, however, traditional computer vision methods are struggling with it unsuccessfully. Especially in the subfield of handwritten music recognition (HMR), where the large variability of handwriting styles poses a great challenge for any handcrafted recognition system. For this reason, recent advances in the field have been made mainly due to the adoption of deep learning. OMR is also a relatively niche field, studied by a minority of researchers, and as such does not have the resources to produce large annotated datasets (compared to speech recognition, natural language translation, or text recognition). Existing datasets are either small or designed for a simpler problem (like staff removal). This poses a challenge to training large neural networks and further application of deep learning. There is, however, a relative abundance of unlabeled data.

The largest dataset of handwritten music sheets is CVC-MUSCIMA, containing 1000 pages (Fornés et al. [2011]). The dataset was designed for writer identification and staff removal, therefore it can not be used to learn, for example, object detection. A subset of 140 pages has been labouriously annotated by Hajič jr. and Pecina [2017] to form the MUSCIMA++ dataset. This dataset can be used to train models for many tasks since the annotation scheme (Music Notation Graph) is very versatile, however, there are still 860 unannotated pages of the original CVC-MUSCIMA dataset, that are not used during such training. This situation is exactly what semi-supervised learning is designed to deal with.

Semi-supervised learning (SSL) is a collection of methods, that aim to incorporate the unlabeled data during training to produce a more performant model. Many of these methods are designed for classification problems. Unfortunately, we explore SSL in the context of semantic segmentation and this lets us utilize only generative methods of SSL. We do not focus on traditional classification, because it is not a scheme applicable for processing entire pages of music. It can only classify already detected symbols. Moreover, semantic segmentation is able to classify symbols of rather extreme dimensions and varying shapes (e.g. slurs and beams[1]) that cannot be realistically cropped and classified in a traditional setting. Semantic segmentation is also used as a basis for object detection, which is the most important step of every music recognition pipeline.

It has been shown that the U-Net architecture yields the best results for semantic segmentation of musical symbols (Ronneberger et al. [2015], Pacha et al. [2018]). The architecture is also similar to a variational autoencoder, which is at the heart of generative semi-supervised learning. This motivates us to use a U-Net for our SSL exploration. The U-Net architecture was designed specifically for semantic segmentation and the presence of skip connections and the fact of being fully convolutional are direct consequences of this intention. These same properties, however, make it difficult to use for semi-supervised learning. Skip connections prevent the use of this architecture for unsupervised learning (as a generative model) and the fully convolutional property complicates interpretability of the latent space.

[1] slurs and beams are elongated symbols of music notation

We have designed a novel scheme for training a U-Net model that sidesteps these complications and lets us train the model in a semi-supervised way. We consider this scheme to be the main contribution of this thesis.

Our scheme pivots around the idea of masking out parts of the input image and training the network to reconstruct the full image. This allows us to train the model like an autoencoder and have it learn abstract features, despite having skip connections between the encoder and the decoder. Figure 1.1 shows a reconstructed image, where generated stafflines and stems are clearly visible.

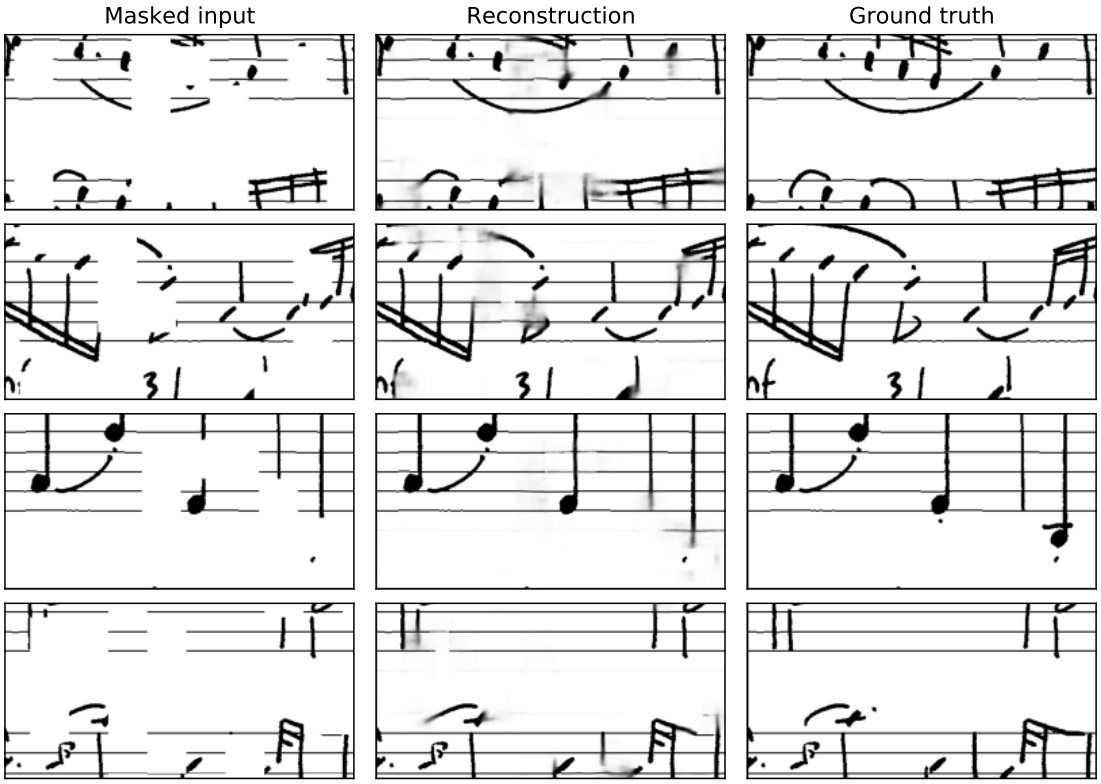


Figure 1.1: Our training scheme manages to train a U-Net architecture in an unsupervised manner to produce reconstructions of masked images. Each row is an example of a performed reconstruction (middle column), together with the input and expected output (side columns). We can see that stafflines are reconstructed very well, whereas more complicated symbols tend to get blurred out.

Most of our experiments focus on notehead segmentation, because noteheads are an ideal symbol for this kind of experimentation. They are abundant in the input images, they are the most important symbol semantically, and they have great variability in terms of appearance (handwritten noteheads).

We train the model in the context of three experiments and compare it against a corresponding supervised baseline. The model prediction is evaluated using pixelwise F1 score, as it aggregates both precision and recall into a single number. Performing the evaluation pixelwise lets us compare the model to the baseline with minimal added complexity (e.g. bounding box detection) and thus provide direct feedback on the only property varied – the amount of unlabeled data.

All three experiments suggest, that training a U-Net in this semi-supervised scheme acts as regularization and helps stabilize the training. Unfortunately, we did not manage to surpass the supervised baseline, we only got the same per-

formance. We belive that training the reconstruction with *pixelwise binary cross entropy* loss function causes the model to produce fuzzy reconstructions and to learn representations that are not useful for improving segmentation performance. This may be because there are many possible reconstructions of a masked area, and the model learns to reconstruct their fuzzy average, instead of picking one and making a good-looking reconstruction. We belive that adding a smarter loss function, such as a discriminator network, could lead to improvements. We plan to explore this idea in a future work.

The primary contributions of this thesis are:

- Modification of the U-Net architecture for simultaneous segmentation and reconstruction.
- Unsupervised training scheme for the architecture based on reconstructions of masked patches.
- A thorough exploration of relevant hyperparameters.
- Detailed experiments, designed to assess the viability of this model in the semi-supervised setting.

Code for the model and all experiments is available in a GitHub repository at `https://github.com/Jirka-Mayer/MasterThesis`.

1.1 Thesis Outline

Chapter 1 This chapter is a condensed overview of the entire thesis, starting with the context and our motivation, describing the architecture and experiments, and closing by talking about possible future work.

Chapter 2 This chapter is a short overview of the most influential articles for this thesis. It introduces the U-Net architecture and describes it in the context of music recognition. It concludes by highlighting the abilities of semi-supervised generative models.

Chapter 3 This chapter is a brief overview of the field of optical music recognition. It describes the transition from traditional approaches to deep learning, lays out the currently accepted recognition pipeline, and ends by introducing all of the datasets used in this thesis.

Chapter 4 This chapter is an overview of semi-supervised methods, grouped into their respective categories. More attention is given to generative semi-supervised methods, which are explored in the second half of the chapter.

Chapter 5 This chapter contains a detailed description of the proposed architecture and the preparation of its training data. It also talks about the chosen evaluation metrics.

Chapter 6 This is the core chapter of this thesis. It starts with an overview of relevant hyperparameters and experiments we did to set their default values. The last three sections describe three major experiments. The first two assess the viability of our model in the semi-supervised setting (Section 6.3 and Section 6.4). The last experiment is a mixture of transfer learning and semi-supervised learning (Section 6.5), where we attempt to learn handwritten music segmentation from printed ground-truth music.

2. Related Work

The following text is a short overview of the most influential articles for this thesis. It perimarily revolves around the U-Net archtiecture and its use in musical object detection. The last article serves as a showcase of the abilities of semi-supervised generative models.

2.1 The U-Net Architecture

The article *U-Net: Convolutional Networks for Biomedical Image Segmentation* (Ronneberger et al. [2015]) introduces a fully-convolutional neural network, that is surprisingly good at performing image segmentation. The article uses this architecture for semantic segmentation and instance segmentation of various biomedical images.

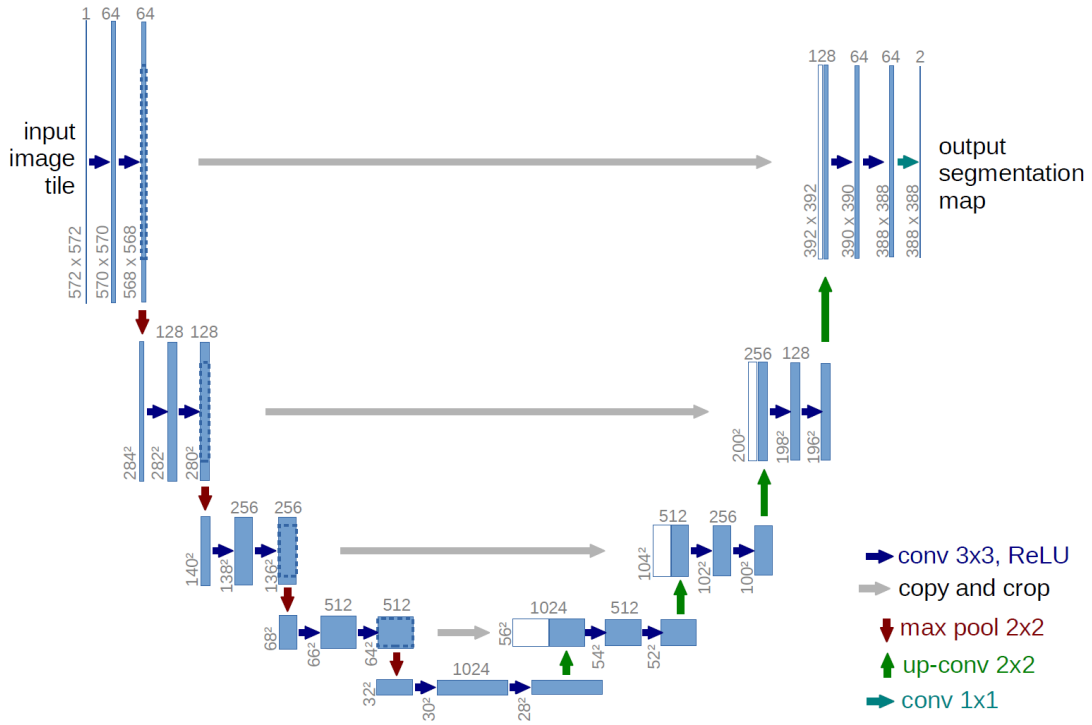


Figure 2.1: The U-Net architecture with a contracting path and an expansive path. The image is taken from Ronneberger et al. [2015].

The authors describe the left part of the network as the contracting path. It follows the typical structure of a convolutional network. The right part is called the expansive path. It is built as an inverse to the contractive part. The combination of contraction and expansion causes the network to first gather context information around each point of the image and then spread it back out to inform the segmentation process. The two halves are connected using skip-connections so that the expansive path also has accurate local information available. Since the architecture resembles an autoencoder, we refer to the two halves of the network as an encoder and a decoder. One additional advantage of this architecture is its speed of both inference and training.

2.2 Music Object Detection

A variation of the U-Net architecture has been first utilized in the context of music recognition in the article *On the Potential of Fully Convolutional Neural Networks for Musical Symbol Detection* (Dorfer et al. [2017]). Authors used it for notehead detection, as a continuation of their previous research into convolutional neural networks for music symbol detection. They managed to outperform other approaches with a much simpler system and faster inference time.

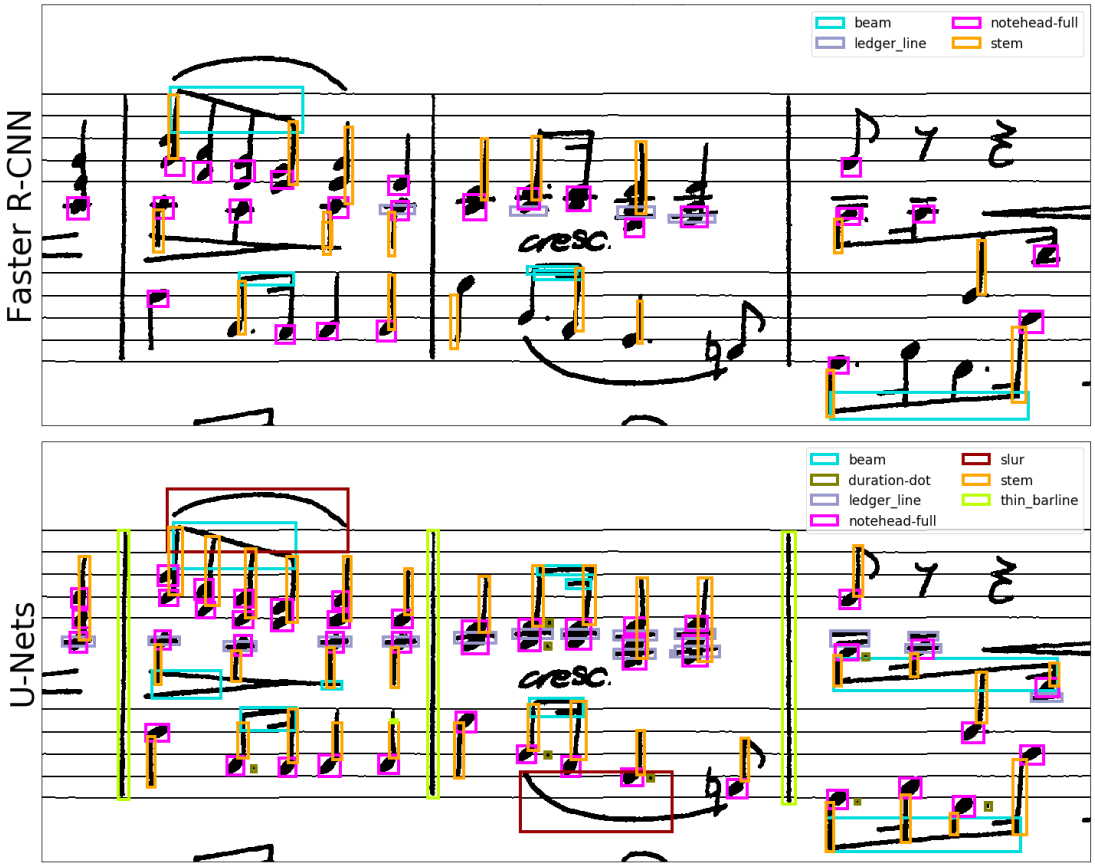


Figure 2.2: Comparison of the R-CNN and the U-Net architecture for object detection on the MUSCIMA++ dataset. The RetinaNet single-shot detection is very poor, so the image is omitted to save space. The original image with additional images showing other datasets can be found in (Pacha et al. [2018]).

Both authors then refined the approach in the article *Towards Full-Pipeline Handwritten OMR with Musical Symbol Detection by U-Nets* (Hajič jr. et al. [2018]). The architecture was extended to perform multi-class segmentation in one pass and domain-specific tricks have been added to increase performance on some symbols. A notation assembly system was designed using decision trees and a complete pipeline with MIDI output was constructed and evaluated.

In the same year, a comparison of object detection architectures for musical symbols was published: *A Baseline for General Music Object Detection with Deep Learning* (Pacha et al. [2018]). The authors trained three object detection architectures, namely Faster R-CNN, RetinaNet, and U-Net. Their evaluation was performed using three datasets with object detection ground truth and varied appearance and content. These were the Capitan dataset (mensural notation), the

DeepScores dataset (digitally printed modern notation), and the MUSCIMA++ dataset (handwritten modern notation). The article establishes baseline results in different areas of music recognition, but most importantly, identifies the U-Net architecture as the best-known architecture for object detection.

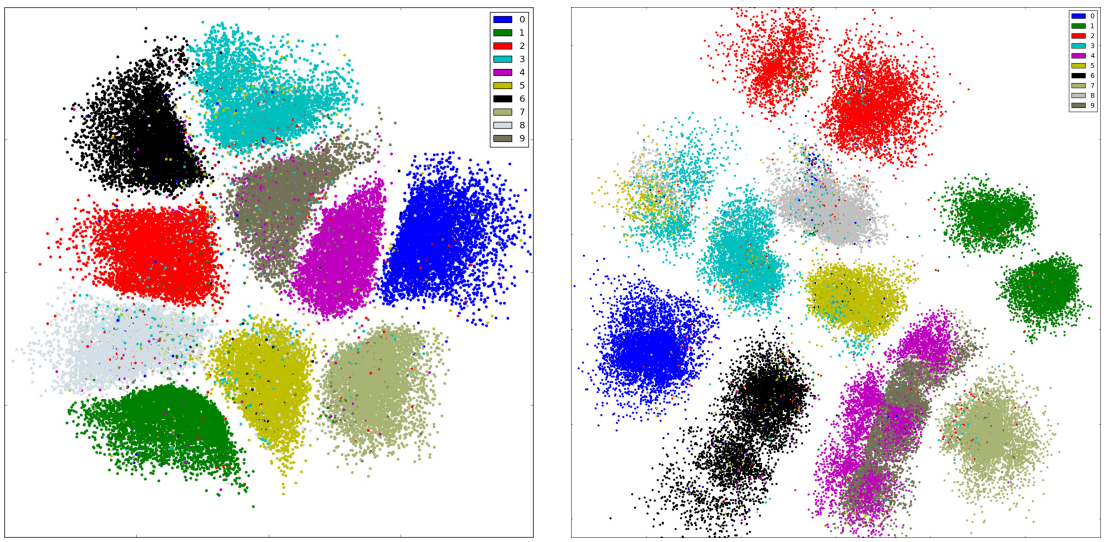
2.3 Generative Semi-supervised Learning

The article *Adversarial Autoencoders* (Makhzani et al. [2015]) proposes an autoencoder architecture, where the latent space distribution is shaped by an adversarial mechanism. A separate discriminator network is trained to distinguish between real embeddings from the dataset and fake embeddings drawn from the desired distribution. This process forces the encoder to produce embeddings that match the desired distribution (e.g. gaussian). The resulting latent space is tightly filled with dataset embeddings and there are no holes present. This makes the decoder a very good generative model, as any random sample from the desired distribution will be correctly decoded into a realistic data point.

The article compares the latent space of an adversarial autoencoder (AAE) to a variational autoencoder (VAE) (Kingma and Welling [2014]), and it is clearly visible how an AAE better matches the desired distribution. Most of these visualizations are performed on the MNIST dataset (Deng [2012]).

The article then describes modifications of the architecture, related to splitting the latent vector into a continuous and a categorical part and treating the categorical part as class information. These modified architectures are then used to perform many interesting tasks, such as semi-supervised classification, unsupervised clustering and style-content disentangling.

The article is referenced here as a demonstration of the capabilities of generative semi-supervised models.



(a) semi-supervised, 1000 labels (b) unsupervised

Figure 2.3: Semi-supervised and unsupervised dimensionality reduction results for an adversarial autoencoder on the MNIST dataset. Images taken from Makhzani et al. [2015].

3. Optical Music Recognition

This chapter contains an introduction to optical music recognition (OMR). Most of its content is based on the article *Understanding Optical Music Recognition* (Calvo-Zaragoza et al. [2020]).

The article defines OMR as *a field of research that investigates how to computationally read music notation in documents*. This definition attempts to clearly define the term. OMR is a field of research, therefore it encapsulates the investigation of many problems – it is not a single task. OMR attempts to computationally read music; it applies existing understanding of various music notations and does not study these notations as such. Recognized documents may be of many types (scanned, digitally engraved, printed, handwritten, captured from a stylus) and be represented in different music notations (common western, mensural, tablature). This thesis focuses on reading entire pages of handwritten music, expressed in the common western music notation.

The recognition process on its own is also not very useful. It serves as one phase of solving some larger problem, like metadata extraction, archive searching, audio replay, editing, or conversion to some other music data format. Each one of these applications has a different set of priorities regarding recognition. If the goal is to search by melody, we do not need to recognize rhythm; if the goal is to replay audio, we do not need to read lyrics.

It should be noted, that there is a difference between recovering musical semantics and recovering music notation. This stems from the specifics of how music notation encodes musical ideas. Musical symbols on a staff compete for space and this influences their relative position. In music notation, there is also a lot of freedom in how a musical idea is expressed. Some rhythmic phrases may be expressed using either duration dots or using ties and the choice may depend on the writer's preference or on the rhythmic context. Recovering music notation is the easier task of reading individual notation symbols and their relationships. Recovering musical semantics requires an additional step of interpreting the parsed musical notation.

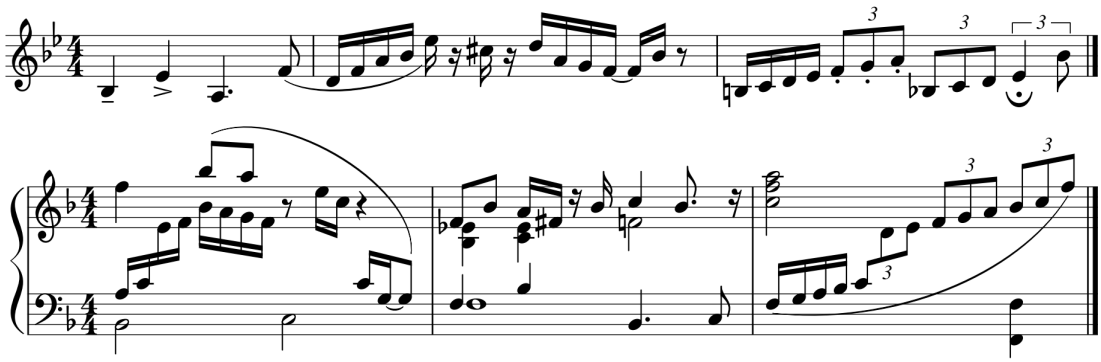


Figure 3.1: Comparison of monophonic music (top) to a pianoform (bottom). Monophonic music contains only a single voice and can be represented sequentially. Pianoform contains multiple voices spread out over two staves, making it behave more like a graph. The image is taken from (Calvo-Zaragoza et al. [2020]).

It is only natural to ask about the relationship between OMR and text recogni-

tion. Both fields attempt to read documents containing sequential data and there are many other similarities. However, music recognition is much more difficult for multiple reasons. Music notation is contextual, meaning that interpretation of some symbols depends on other symbols around it. The pitch of a note is one example. We need to know both the key signature (correctly read the clef and accidentals at the beginning of the staff) vertical note position (in relation to staff lines) and possibly an additional accidental in front of the notehead. The size and shape of symbols vary from very small (a dot) to relatively large (beams, slurs, hairpins) to spanning the entire page (tall barlines or braces). Lastly, a single staff of music can contain multiple voices and a pianoform may contain two to three voices on two staves and some notation symbols may be present on both staves simultaneously. Text recognition usually does not need to deal with such a level of complexity.

3.1 Approaches to OMR

Traditional recognition systems relied on the detection of individual symbols and their classification. Stafflines intersect the majority of all symbols, which posed a problem to most object detectors that relied on finding connected components. For this reason, staffline removal was an important preprocessing step.

In recent years, deep learning has been applied to many OMR problems with great success. Especially symbol classification and staffline removal have been considerably improved (Calvo-Zaragoza et al. [2017], Pacha and Eidenberger [2017]). Convolutional neural networks are well capable of learning handwritten symbol classification despite the variance in handwriting styles. In fact, they are able to perform classification without the need for staffline removal – aggregating multiple phases of a traditional pipeline and greatly simplifying the task.

The pipeline for music recognition has been refined over the years (Bainbridge and Bell [2001], Rebelo et al. [2012]) and has currently reached a form with four distinct stages:

- **Preprocessing** Methods that enhance the raw scanned or photographed document for easier processing. These contain perspective compensation, page cropping, contrast enhancement, binarization, and noise removal. Basic layout analysis (such as staff detection) also belongs here.
- **Music Object Detection** Finding and classifying all notation symbols.
- **Notation Assembly** Identifying relations between detected symbols and constructing a machine-readable representation (a sequence or a graph).
- **Encoding** Reading and interpreting the recognized notation to fulfill the given task (conversion to another format, audio replay, ...).

The introduction of deep learning has allowed an alternative approach to be developed, where all of these stages are performed by a single model. This end-to-end approach has yielded state-of-the-art results in other fields, such as handwritten text recognition (Scheidl [2018]). It has also been tried for music recognition (Calvo-Zaragoza and Rizo [2018], Mayer and Pecina [2021]), but

modeling the complex graph-like structure of music notation is problematic, and so this approach has been limited to monophonic music (which can be represented sequentially).

This thesis focuses on semantic segmentation, which is used as the first step of the object detection stage.

3.2 Datasets

With the shift towards machine learning, several datasets have been created to allow machine learning models to be trained. This thesis relies on three of these datasets:

- CVC-MUSCIMA (Fornés et al. [2011])
- MUSCIMA++ (Hajič jr. and Pecina [2017])
- DeepScores v2 (Tuggener et al. [2021])

These are the only datasets in the OMR field that contain semantic segmentation labels (for Common Western Music Notation). A comprehensive list of other available OMR datasets is maintained by Alexander Pacha on his GitHub page[1].

3.2.1 CVC-MUSCIMA

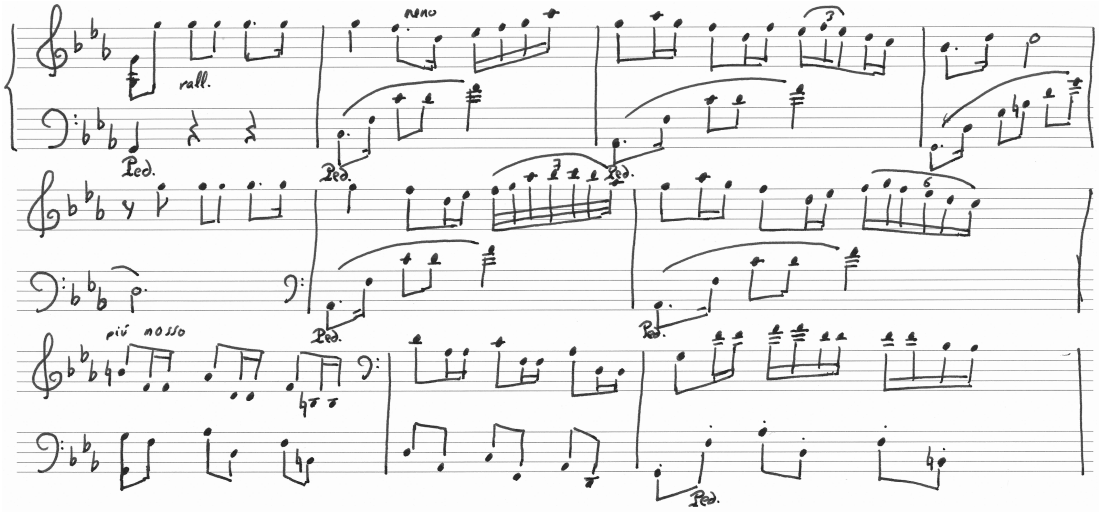


Figure 3.2: One page from the CVC-MUSCIMA dataset. The image is taken from the website `http://www.cvc.uab.es/cvcmuscima/index_database.html`

This dataset was introduced in the article *CVC-MUSCIMA: A ground truth of handwritten music score images for writer identification and staff removal* (Fornés et al. [2011]). It contains 1000 pages of handwritten music, created by having 50 writers transcribe 20 unique music pages. It was designed for the tasks of

[1] `https://apacha.github.io/OMR-Datasets/`

staff removal and writer identification. It is also the only handwritten music dataset, consisting of entire music pages – all other handwritten datasets contain only individual symbols. This makes it very important for research focusing on symbol detection.

Since the dataset does not contain segmentation labels, we will use it primarily as a source of unlabeled training data.

3.2.2 MUSCIMA++

MUSCIMA++ was created by Jan Hajič jr. and Pavel Pecina as a general-purpose OMR dataset. It was introduced in the article *In Search of a Dataset for Handwritten Optical Music Recognition: Introducing MUSCIMA++* (Hajič jr. and Pecina [2017]). The dataset builds on top of the CVC-MUSCIMA dataset, providing rich annotations for 140 selected pages. The annotation scheme was designed to be sufficiently low-level for tasks such as object detection (bounding boxes, symbol classes, segmentation masks), while also having relationship data in the form of an oriented graph, that lets a user extract semantic information about the music. Dataset authors call this annotation scheme the *Music Notation Graph* (MuNG).

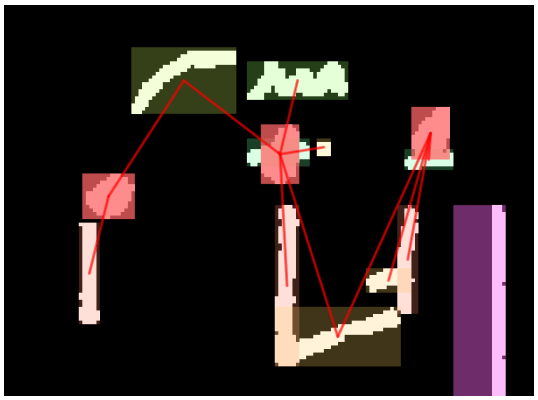


Figure 3.3: Annotations present in the MUSCIMA++ dataset (bounding boxes, segmentation masks and the notation graph). The image is taken from Hajič jr. and Pecina [2017]

The dataset was updated in 2019, fixing bugs and modifying class names to be aligned with the SMuFL[2] standard. A similar update was also performed on the DeepScores dataset (Tuggener et al. [2021]), making it easier to use both datasets simultaneously. The latest dataset description and accompanying tools can be found on the GitHub page[3] of the OMR Research group[4].

We will use this dataset as a source of labeled data for semantic segmentation.

3.2.3 DeepScores

Version 2 of this dataset was introduced in the article *The DeepScoresV2 Dataset and Benchmark for Music Object Detection* (Tuggener et al. [2021]). The dataset

[2] `https://www.smufl.org/`
[3] `https://github.com/OMR-Research/muscima-pp`
[4] `https://omr-research.net/`

contains entire pages of printed music, with annotations best suited for object detection, semantic segmentation, and object classification. It was created from MusicXML documents taken from the MuseScore[5] website and engraved using the LilyPond[6] tool. The dataset contains 255,386 pages of music, but also provides a dense and diverse subset, having only 1,714 pages. Version 2 also introduced a MUSCIMA++ compatibility mode, making it easier for the two datasets to be used simultaneously.

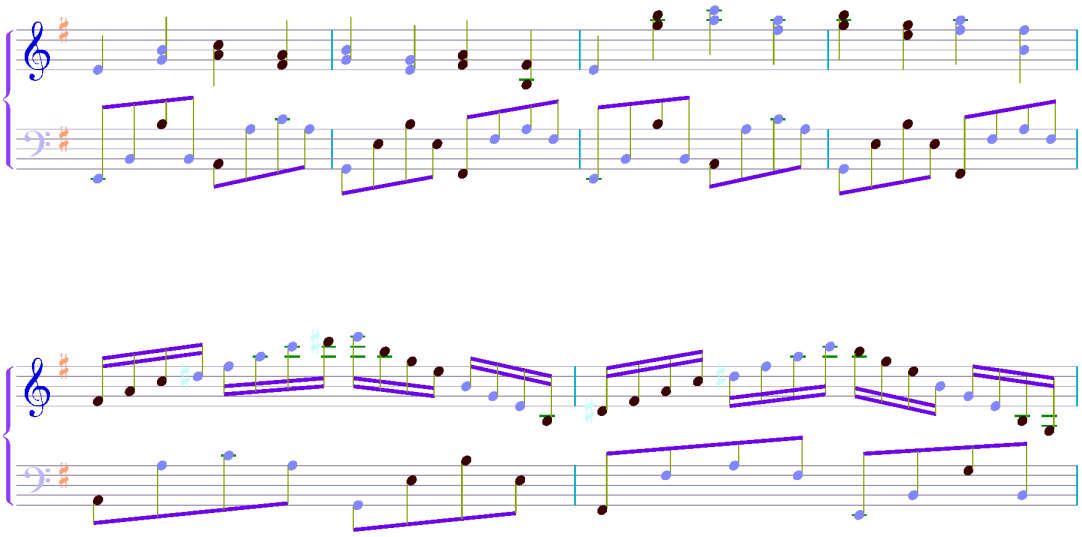


Figure 3.4: An example of semantic segmentation labels from the DeepScores v2 dataset. The dataset is very large, but digitally engraved.

We will use this dataset as a source of labeled data for semantic segmentation.

[5] https://musescore.com/sheetmusic
[6] https://lilypond.org/

4. Semi-supervised Learning

This chapter provides an overview of semi-supervised learning (SSL). The following text draws information predominantly from the article *An Overview of Deep Semi-Supervised Learning* (Ouali et al. [2020]) and from the book *Semi-supervised learning* (Chapelle et al. [2009]).

In deep learning, the goal typically is to learn some function $f : X \to Y$ by training a neural network on pairs of data points (x, y). Deep neural networks have a large number of trainable parameters, which require adequately large training datasets. Not having enough training data causes the model to overfit and then be unable to generalize to unseen data. Producing training datasets usually requires a lot of manual labor. Obtaining input data from the domain X is typically easy, the difficult task is assigning correct outputs from Y (called labeling). For this reason, many niche domain problems (e.g. handwritten music recognition or cuneiform recognition) lack sufficiently large labeled datasets for training deep learning models (Hajič jr. and Pecina [2017], Brandenbusch et al. [2021]). These domains, however, have a relative abundance of unlabeled data.

Semi-supervised learning is a set of methods that aim to utilize this unlabeled data in addition to the available labeled data. The goal of these methods is to produce models that perform better, than models trained on the labeled data alone. The field of SSL emerged in the 1970s and has since accumulated many techniques. This thesis focuses on a subset of these techniques, that are based on generative models. However, this chapter provides a short overview of other available techniques as well.

The following text uses terminology, that should be explained first. We will do that with an example:

Figure 4.1 shows the latent space of a variational autoencoder trained on the MNIST dataset (Kingma and Welling [2014], Deng [2012]). This dataset contains grayscale images of handwritten digits, 28x28 pixels in size. In machine learning, the *manifold hypothesis* states, that *real-world high-dimensional datasets lie along low-dimensional manifolds inside that high-dimensional space.* The space of all 28x28 images is such a **high-dimensional space** (784 dimensions). Manifold, mathematically speaking, is a continuous, locally-euclidian space (e.g. a plane, sphere, 3D space, Möbius strip). The latent space of the autoencoder is a 2D plane (as seen in Figure 4.1) and it is the stated **low-dimensional manifold** containing all the data. The data points lie on this manifold, the model has only learned, how is this 2D manifold embedded in the 784-dimensional space. (Note that it is not the only manifold the data lies on, this is just the one discovered during training.)

The data points are not evenly scattered throughout the high-dimensional space. They coalesce into groups called **clusters**. Points within these clusters are located relatively close to each other and usually share the same class. This is a similar statement to saying that all digits 7 look alike. These clusters are clearly visible in the latent space (Fig. 4.1), where they form very distinct blobs. The space, where clusters sit, is referred to as **high-density regions** – many items from the dataset are located here. Conversely, the space between clusters,

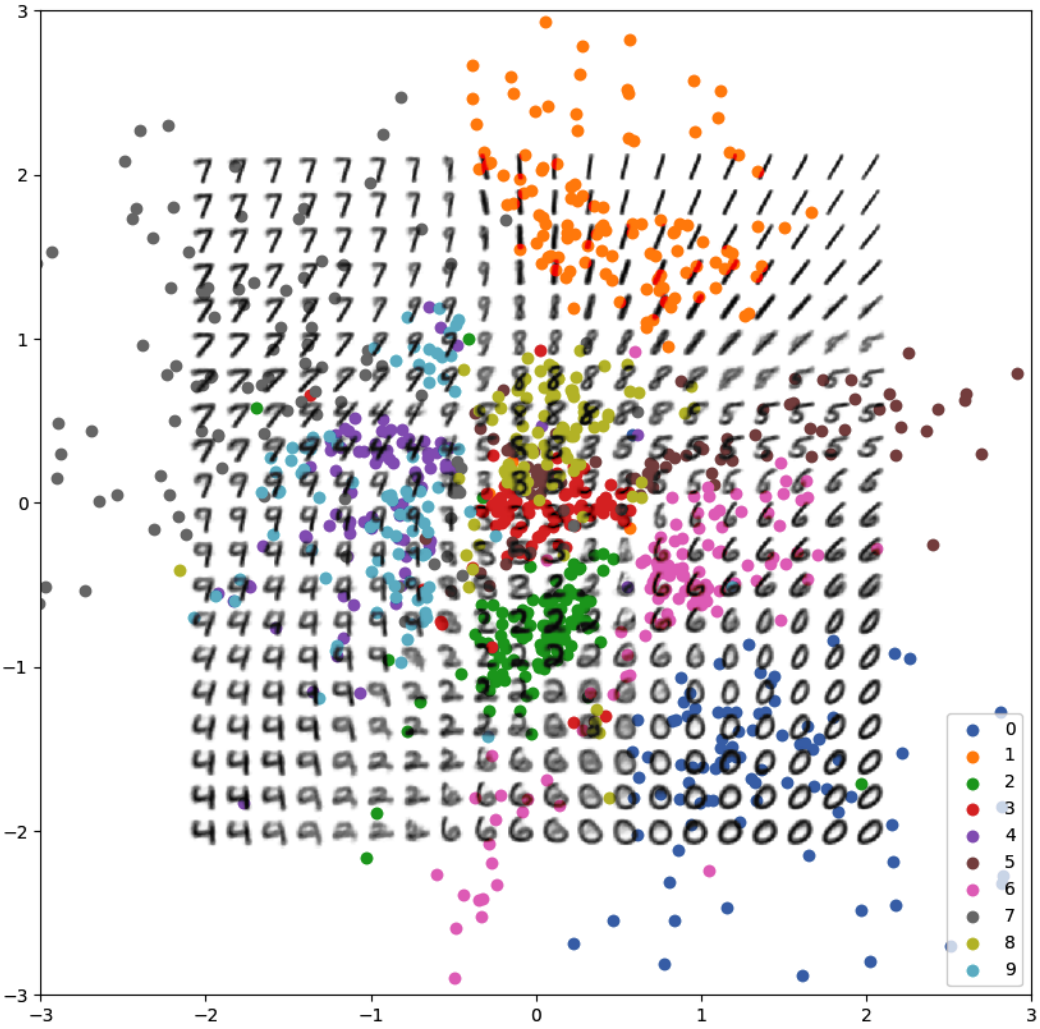


Figure 4.1: Variational autoencoder learns to project 784-dimensional space of MNIST images down to only 2-dimensional space. We can see clusters of individual classes, forming high-density regions and being separated by low-density regions.

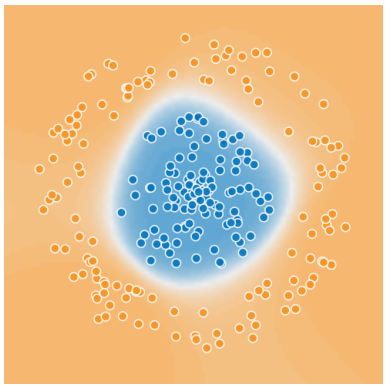


Figure 4.2: Visualization of a classification task, with the decision boundary visible in between the two classes (the while region). Image is taken from the TensorFlow Playground at `https://playground.tensorflow.org/`.

where almost no data points are present, is referred to as **low-density regions**.

A classification task learns a function that assigns a label to each point of the input space. This label is discrete. We can color the input space according to the assigned label and that would produce regions, where all points have the same label. The place where these regions meet is called the **decision boundary**. A decision boundary can be seen in Figure 4.2. In a well-trained classifier for a well-defined classification problem, this decision boundary should lie in the described low-density regions.

4.1 Assumptions

Before we start using semi-supervised methods, we should first understand the assumptions that underpin them. These assumptions are mostly intuitive and are satisfied in almost all real-world problems, however stating them explicitly yields a better understanding of these methods.

- **The Smoothness Assumption.** *If two points x_1, x_2 reside in a high-density region and are close together, then their corresponding outputs y_1, y_2 should also be close together.* For example, if we have a picture of digit 7, then small variations in its shape and color should still be interpreted as digit 7. The opposite statement also holds; if the two input points are separated by a low-density region, the outputs must be distant from each other. This assumption is primarily helpful in a classification task, not so much in a regression task.

- **The Cluster Assumption.** *If points are in the same cluster, they are likely to be of the same class.* This assumption connects the classification task to the smoothness assumption. It indirectly states, that the decision boundary is located in low-density regions, because otherwise, it would cut a cluster in half, causing close points to fall to different classes (which is a violation of the assumption). This assumption can be used as a motivation for methods that push the decision boundary away from data points, into the low-density regions.

- **The Manifold Assumption.** *The (high-dimensional) data lie (roughly) on a low-dimensional manifold.* The problem with high-dimensional data is that as the number of dimensions increases, the volume of the space grows exponentially. This means that most of the space is not covered by any data points in our dataset and that makes it difficult to learn to classify. This assumption states, that our data points lie on a small subspace (manifold) of the entire space and that a projection can be learned, that maps this manifold onto a low-dimensional space. Learning the classification task for this low-dimensional space should be much easier.

4.2 Methods

The following section lists major SSL methods. These methods are often directly based on previously stated assumptions and are not mutually exclusive, in fact,

most of them can be used simultaneously as so-called holistic methods. These are, however, not covered in this chapter.

4.2.1 Consistency Regularization

The core idea of this method is that a small neighborhood of each data point should have the same label as that datapoint. This idea follows directly from the cluster assumption. In this method, we train the model to minimize distance between a known datapoint x and a perturbed datapoint $\hat{x}$. This training is performed in addition to the usual supervised training. This method does not rely on the corresponding label y, instead it trains to minimize the distance between $f(x)$ and $f(\hat{x})$. This lets us utilize all unlabeled data points as well.

The method can be seen as an extension of supervised learning. In supervised learning, we train on individual points and learn the overall shape from them. Here, we train on small neighborhoods, which makes sure the decision boundary will not come close to any individual data point. Further utilization of unlabeled data points in the proximity of labeled data points should force the decision boundary even further away, into low-density regions (Zhu [2005]).

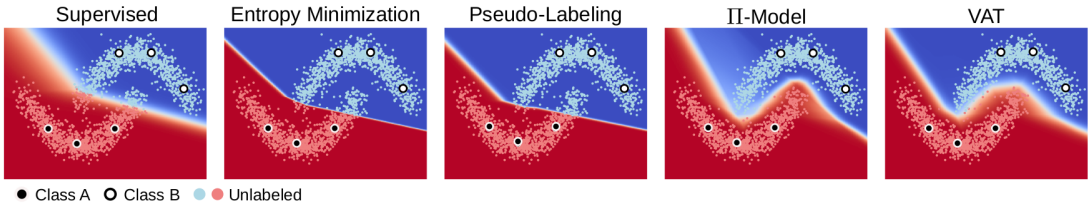


Figure 4.3: SSL toy example – the *two moons* dataset with three labeled examples for each class. Π-model and VAT are both consistency regularization techniques. The image is taken from Ouali et al. [2020].

4.2.2 Proxy-label Methods

This is a set of methods, that train a supervised model on the labeled data and then use it to label a portion of the unlabeled data. These methods are also called *bootstrapping*. To distinguish these later-added labels, they are called *pseudo-labels*.

One of these methods is *self-training*. In self-training, labeled data is used to train a supervised classification model. This model is then used to classify all unlabeled data points and since its output is a softmax layer, we can not only pick the most likely class but also measure the confidence in that class. We define a threshold τ and only assign pseudo-labels to data points above this threshold. This process can be repeated, until a stopping condition is met (e.g. there are no more unlabeled points with confidence above τ). The main problem of this method is the inability to correct mistakes made during the labeling step.

Pseudo-labeling is a similar method, where the unlabeled data points are also assigned pseudo-labels. In this case, these pseudo-labels are treated as trainable parameters, similar to model parameters, and are optimized together with the model during training. The difficult part is designing a good loss function for

pseudo-labels, because applying this method naively causes the pseudo-labels to overfit, due to so-called confirmation bias.

Self-training has been used for semantic segmentation in the work of Babakhin et al. [2019].

4.2.3 Graph-based Methods

These methods frame the problem in the language of graphs (Zhu [2005]). Data points are represented as vertices of a graph and weighted edges are present between pairs of vertices, where the weight corresponds to some similarity between the two points. The labeling task can be viewed as the propagation of information along the edges of the graph.

This propagation resembles proxy-label methods, however, due to the graph framing, we can consider things like vertex neighborhood and its impact on the examined node, or we can leverage algebraic structures like adjacency matrix.

A different set of graph methods aims to learn data point embeddings, which preserve the structure of the input graph. The goal is to represent each vertex (data point) as a low-dimensional vector, where a simple similarity function (e.g. inner product) can be used. This again resembles deep learning techniques, such as autoencoding (see Figure 4.1), however here, methods derived from graph theory are used to produce these embeddings (e.g. Laplacian Eigenmaps or Locally Linear Embeddings).

4.2.4 Entropy Minimization

Consistency regularization methods attempt to push the decision boundary away from clusters and into the low-density regions by stabilizing the model output in a neighborhood around each data point. Entropy minimization is a different technique that attempts to do the same thing. In entropy minimization, we penalize the model for being unsure about its predictions. Low-confidence predictions are predictions with more than one class having non-zero probability. Such probability distributions have higher entropy than one-hot distributions. By adding a loss term that minimizes entropy of predictions during training, we can make areas around datapoints more stable (Grandvalet and Bengio [2004]). This method can not, however, be used alone for high-capacity models (deep neural networks), as it causes the training to overfit. Instead, it may be used as a supplementing technique to another SSL technique.

4.3 Generative Methods

Generative semi-supervised learning leverages generative models for the classification task. A generative model learns to describe the distribution of data $p(x)$. The assumption here is that this understanding helps the model with the classification task $p(y|x)$. Generative SSL can be viewed either as an extension of supervised learning (classification with the addition of $p(x)$ modeling) or as an extension of unsupervised learning (modeling, extended by label information).

4.3.1 Variational Autoencoders

A variational autoencoder (VAE) (Kingma and Welling [2014]) is a model that attempts to learn a mapping between the input space and a smaller latent space. The latent space is also required to roughly follow a unit gaussian distribution, which causes the space to be filled and smooth, such that sampling any point of this space yields a reasonable corresponding data point. Moreover, the model architecture forces clusters of similar data points to occupy roughly gaussian-shaped regions in the latent space. A visualization of this latent space can be seen in Figure 4.1. The variational autoencoder consists of an encoder and a decoder. The encoder learns the mapping from the input space to the latent space and the decoder learns the inverse mapping. The decoder is then considered the generative model, as it can generate a reasonable data point for an arbitrary latent space point. Input data points are typically denoted x and the latent vectors are denoted z.

Kingma et al. [2014] used variational autoencoders in a semi-supervised setting. They propose three ways of extending VAEs:

M1 Model First, an unsupervised VAE model is pre-trained using all available labeled and unlabeled samples. The goal is to learn a low-dimensional latent space and then transform all data points to this space (create embeddings). After this, a supervised classifier is trained on these embeddings using only the labeled portion of the data. This setup leverages the manifold assumption and the assumption that learning classification in a low-dimensional space is easier than in a high-dimensional space. Also, since the variational autoencoder forces the creation of gaussian-shaped clusters in the latent space, the classifier has an easier job separating these clusters.

M2 Model In the previous setup, labels y were ignored when training the autoencoder. Here, the latent vector z is extended by an additional vector y. This vector will be set to the label if the label is known and will be left unconstrained for unlabeled data. We can view this as splitting the original latent vector z into two parts, one is left free to be trained as usual, and the second (categorical) part is sometimes trained to match the label y of the learned datapoint x. If the label is not known, this part is trained in the unsupervised mode only, like the original latent variable z. When the training finishes, the encoder can be immediately used as a classifier, by interpreting the latent vector y as the output label. The two parts of the latent representation (z and y) can be interpreted as style and content embeddings respectively.

M1 + M2 Model The last setup involves training the model M1 in an unsupervised way and then training the model M2 in a semi-supervised way on embeddings produced by M1. It can be viewed as the M2 model, utilizing dimensionality reduction by M1. This stacked setup yields the best results.

4.3.2 Adversarial Autoencoders

A newer article by Makhzani et al. [2015] introduces an architecture, called an Adversarial Autoencoder (AAE). The architecture is in many ways similar to the variational autoencoder, but it differs in the way it regularizes its latent space. Whereas a VAE uses KL-divergence loss to force the latent space into a gaussian distribution, an AAE uses a discriminator network (inspired by generative adversarial networks). This architecture can be used and extended in similar ways to the M1 and M2 models and yields even better results for tasks, such as semi-supervised classification, unsupervised clustering, and style-content disentangling.

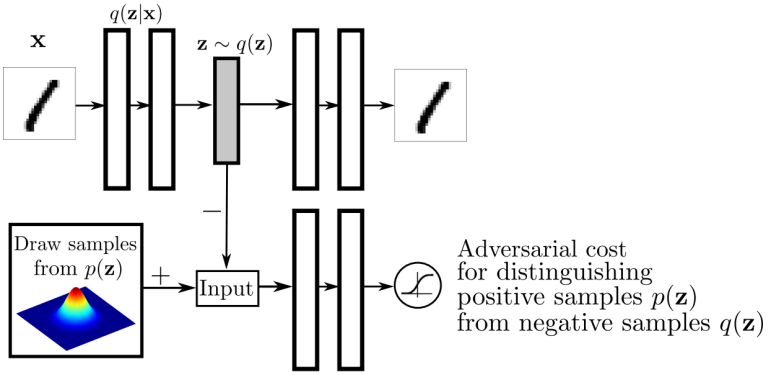


Figure 4.4: The architecture of an adversarial autoencoder. The top two networks form a typical autoencoder with the embedding vector in the middle. The bottom model is a discriminator, forcing the encoder to produce embeddings matching the distribution $p(\mathbf{z})$. The image is taken from Makhzani et al. [2015].

4.3.3 Generative Adversarial Networks

Generative adversarial network (GAN) is a generative architecture consisting of two parts: a generator and a discriminator (Goodfellow et al. [2014]). They are trained together in an alternating fashion, where the generator is trained to produce realistic-looking data (to fool the discriminator) and the discriminator is trained to distinguish this generated data from the real-world data. The tension between these two models causes both of them to get increasingly better and the generator ends up producing well-looking synthetic samples. Input to the generator is a latent vector z, drawn from a prior distribution (e.g. a gaussian). The generator uses this vector as a seed for the generation. The discriminator outputs a single value, interpreted as the probability of the shown sample being real or generated.

CatGAN Categorical generative adversarial networks (CatGAN) replace the discriminator with a classifier (Springenberg [2016]). The output of this classifier is a softmax layer with C neurons. The discriminative task is transformed onto the classifier by considering its certainty of prediction. In the section on entropy minimization (Section 4.2.4), we described how certainty of prediction is related to entropy of the predicted distribution, and also how the entropy can be minimized

(or maximized) to force the model to be certain (uncertain). In the unsupervised mode, the classifier is trained to output one (any) of the classes for real data points with maximum certainty – to recognize real samples. The generator is then trained to confuse the classifier to produce uncertain predictions (uniform distribution).

In SSL, the classifier is trained to produce confident predictions of any class for unlabeled data and confident predictions of known class for labeled data. The supervised loss is a categorical cross-entropy (like in a typical classification setup) and the unsupervised loss is an entropy minimization loss (to force confident predictions of any class).

DCGAN, SGAN Deep Convolutional GANs attempt to learn intermediate representations by training convolutional GANs on unlabeled data (Radford et al. [2016]). Parts of the generator and discriminator are then re-used as feature extractors for supervised classification task. Semi-supervised GANs (SGAN) address and fix issues with DCGAN, such as the fact that the classifier is trained after the GAN has finished training (Zhang et al. [2019]). SGAN is able to train the generator and classifier simultaneously, substantially improving classification performance.

BiGAN In a typical GAN, the latent variable z is only used as a seed for the generator and the generator acts as function $G : Z \to X$. In a variational autoencoder, there exists a full bijection between spaces X and Z. BiGANs extend the basic GAN framework by adding an encoder, that acts as a mapping $E : X \to Z$ (Donahue et al. [2017]). This gives us access to the latent vector z of an existing real-world sample x. The discriminator is extended to have access to both the data points and the latent points, and so it discriminates between pairs of $(x, E(x))$ and $(G(z), z)$. BiGAN therefore serves a similar role to a variational autoencoder and can be extended for use in semi-supervised learning.

4.4 Related Methods

Transfer Learning In transfer learning, the goal is to utilize the knowledge about one problem to solve another different but related problem. For example, utilizing layers of an already trained network (e.g. an autoencoder) as feature extractors for a different task (e.g. classification) is an example of transfer learning.

Domain Adaptation In the previous example of transfer learning, the two tasks have different feature space. Domain adaptation is a subset of transfer learning where both tasks have the same feature space, but different data distribution. For example, training a classifier of handwritten digits on augmented printed digits (handwritten and printed digit images are both images, just with different distributions). Semi-supervised learning differs from domain adaptation in that both problems have the same data distribution (the training and test data come from the same process).

Muti-task Learning Multi-task learning is the process of training one model to solve two different tasks simultaneously. By having one model learn both tasks, the model can share intermediate representations for both of them, achieving better performance, than by having a dedicated model for each task.

5. Methodology

This chapter describes the motivation and process behind performed experiments. It starts by describing our modified U-Net architecture and the challenges related to its training. The two following sections describe the preparation of the training data. The last section is a brief overview of object recognition metrics and it describes our motivation for using pixelwise F1 score as the evaluation metric.

5.1 Architecture

In the *generative model* approach to semi-supervised learning, one usually starts with a model that can be trained in an unsupervised manner (either an autoencoder or a generative adversarial network; Bank et al. [2020], Goodfellow et al. [2014]) and then extends it to also perform the supervised task. For example, when starting with an autoencoder, one can use the encoder part as a dimensionality reduction mechanism and then build a supervised classification network that classifies the learned embeddings (Kingma et al. [2014]).

In our context of music recognition, we are highly motivated to build on top of the U-Net architecture (Ronneberger et al. [2015]) (Figure 5.1). It has first been used for biomedical image segmentation, however, its superiority for object detection in music recognition has been clearly demonstrated by Pacha et al. [2018].

The architecture can be viewed as a fully convolutional autoencoder with residual (skip) connections added between the encoder and decoder on every resolution level. The U-Net encoder is a typical fully-convolutional encoder that gradually reduces image dimensions while increasing the channel count. Such an architecture can learn abstract representations of symbols present in the input image. The decoder then tries to go from these abstract representations back to specific ones, while at the same time modifying the reconstruction to fit the learned segmentation task. The core idea behind this architecture is that the decoder can utilize skip connections during upsampling, thereby producing a pixel-perfect segmentation.

We choose to share the entire U-Net model for both the supervised and unsupervised tasks. The two tasks are differentiated only at the very last layer. The original U-Net architecture ends with a 1x1 convolution layer with sigmoid activation. It can be viewed as a pixelwise softmax for two output classes (like in a typical classification architecture). We decided to fork the architecture here, having one sigmoid convolution for the supervised segmentation task and one for the unsupervised reconstruction task.

The supervised task can be trained in almost the same way as the original U-Net architecture:

- A batch (x, y) of input images and expected segmentation masks is taken from the dataset.
- Input images x are fed through the model, producing a prediction for the segmentation mask $\hat{y}$ and for the reconstructed image $\hat{x}$.

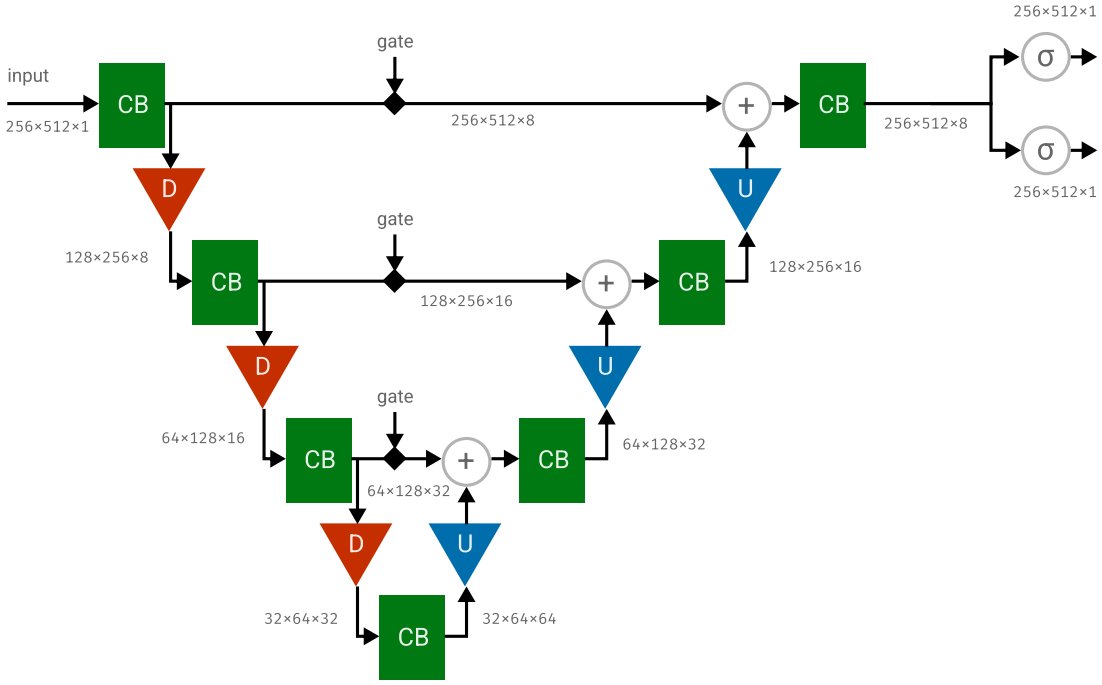


Figure 5.1: Block diagram, depicting the used architecture. It has 4 levels, with gated skip connections on each level. The network is forked after the second to last layer to provide separate segmentation and reconstruction outputs. A detailed explanation of each block is provided in Figure 5.2.

- A loss function is used, computing the distance between y and $\hat{y}$ and generating gradients for the network.
- Reconstruction output $\hat{x}$ is ignored and no gradients for the branch are computed.
- An optimizer uses computed gradients to update model parameters.

The unsupervised task can be trained in the same way, using the other output branch of the model. The model would receive an image on the input and would be trained to produce an identical image on the reconstruction output. This setup requires no labels, only the music images.

The unsupervised training, as described so far, would work for a typical autoencoder, but since the U-Net architecture contains skip connections, the model could learn an identity function without using any of the abstraction-learning layers. The goal of unsupervised learning is learning these abstract features, so this straightforward training scheme is infeasible in our context.

We propose two ways of overcoming this challenge:

- gated skip connections
- denoising

One option is to disable skip connections during reconstruction training and keep them enabled during segmentation training. This should force the model to

learn abstract features during reconstruction, while also being able to utilize skip connections during segmentation.

This gated scheme performs better than a plain autoencoder without any skip connections, but the experiment in Section 6.2.3 suggests that having skip connections permanently enabled works better still. Despite this, we choose to use gated skip connections for reasons explained in that section. We combine this gating with the second technique of adding noise.

The denoising option refers to the process of adding noise to the input image when training the reconstruction task. The model learns to not only reconstruct the input image but also to remove the added noise. We took inspiration from denoising autoencoders (Mao et al. [2016]). The difficult part is designing the noise function such that it would cause the model to learn abstract representations (see Section 5.3).

We consider the described scheme for unsupervised U-Net training to be the main contribution of this thesis. The proposed experiments try to assess the viability of the scheme in the context of semi-supervised learning.

Both the segmentation and reconstruction tasks are trained jointly on composite batches and a single composite loss function. A single optimizer step is used to update model parameters for both tasks simultaneously. The specifics of combining supervised and unsupervised training are described later in Section 6.1.

The last sigmoid layers output images with only one channel. For the reconstruction output, this data is interpreted as a grayscale image. For the segmentation output, it represents the probability the model assigns to each pixel of being in the target class. This means that the model can learn to segment only one symbol class.

Hajič jr. et al. [2018] have explored the option of having multiple segmentation output channels. The hypothesis is that having the model learn multiple classes would improve its accuracy since some internal representations can be shared. An added advantage is the reduced inference time as multiple segmentation masks can be inferred in a single pass. Such a training however introduces problems related to unbalanced class distribution in the training data. Even if these can be overcome, their results suggest minimal accuracy benefits and risk of decreasing accuracy in some cases.

Despite the fact that our experiments' source code lets us easily train multiple classes, we choose not to explore this path as we are interested in the semi-supervised effects only.

Figure 5.2 contains a recursive block diagram of our extended U-Net architecture. The model takes a single image as the input and produces both the segmentation mask and the reconstructed image. When training segmentation, the reconstructed image is ignored and when training reconstruction, the input image is covered in noise, and the output segmentation is ignored.

Each level of the network begins and ends with a convolutional block (CB). It consists of two identical 3x3 convolutional layers with an optional dropout layer in between them. Both convolutional blocks on the same level output the same number of feature channels. The number of feature channels is dictated by a hyperparameter we call *inner features*. If the network has 4 inner features, the zeroth level convolutions output 4 feature channels. Each successive level then

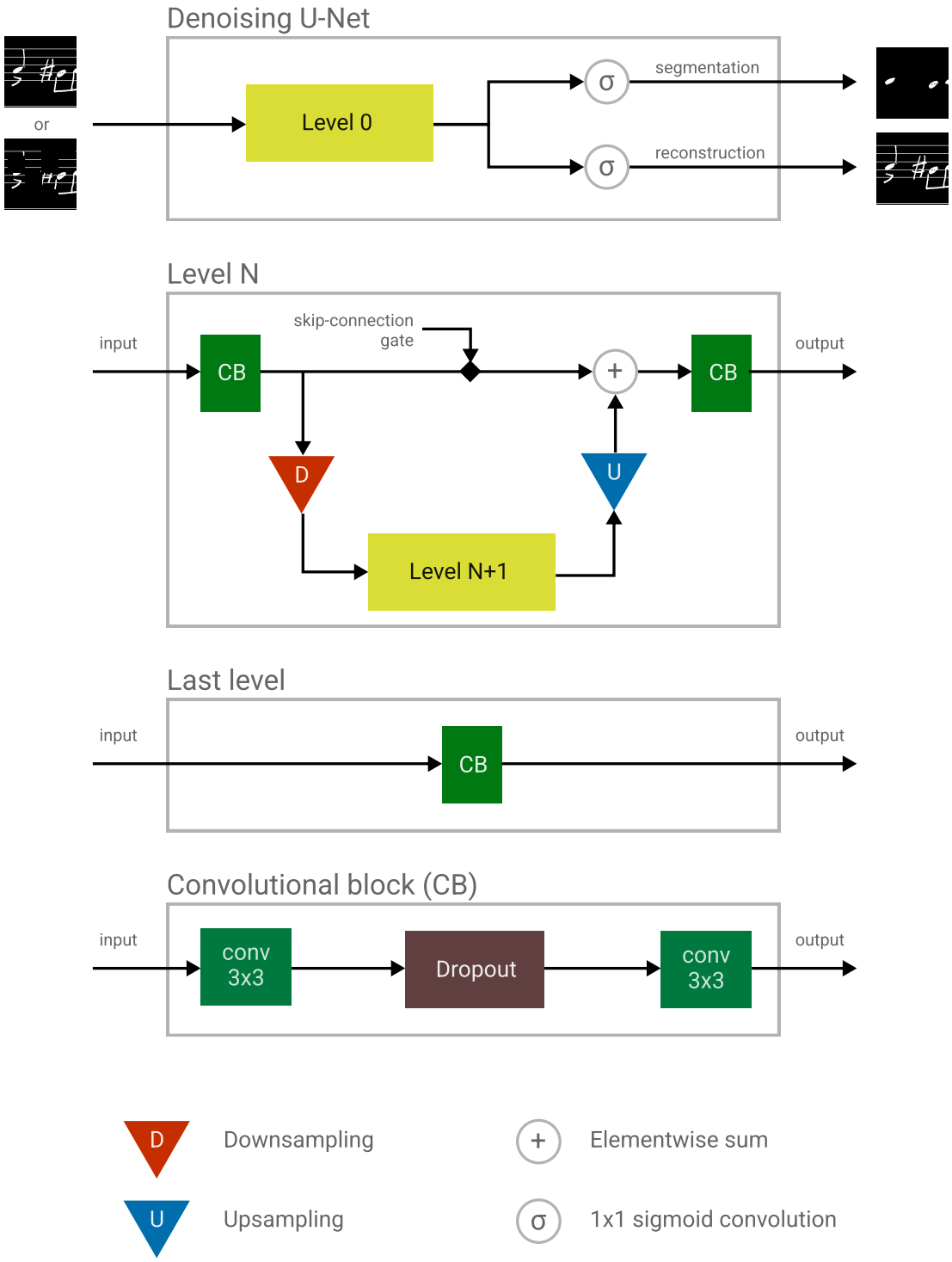


Figure 5.2: A recursive block schema of the used architecture. When training and inferring segmentation, the input images are cropped directly from the given dataset. When training reconstruction, the input images are overlayed with large, tiled noise that zeroes out some pixels.

has twice as many feature channels as the one above it (e.g. 4, 8, 16, 32). Figure 5.1 contains image dimensions and channel counts for each edge in the network. The image dimensions are taken for an input image tile of size 512x256.

The downsampling block is implemented as a 2D max-pooling layer (Goodfellow et al. [2016]). The number of feature channels is preserved. If we track the feature count through the encoder, we always first double the number of channels in the convolutional block and then reduce the spatial dimensionality in half in the downsampling block. This way the model can learn to extract higher-level features before the spatial resolution is lowered. Since the image is two-dimensional, halving the resolution shrinks the number of pixels to one quarter. Combined with the doubling of channel count, one encoder level effectively discards half of the image data. This acts as a bottleneck, forcing the network to learn relevant high-level representations.

The upsampling block has to perform two operations:

- double the spatial dimensions
- halve the channel count

The resolution increase is performed by nearest neighbour interpolation (each pixel is copied to form a 2x2 region of the output image). The reduction in channel count is performed by a 1x1 convolutional layer, which can be trained to select or combine specific input channels. The upsampling output image may also be padded by zeros, if the desired output resolution is not even (we need to exactly match the resolution of the skip connection).

The gate on the skip connection is implemented as a multiplication by one or zero. This also causes any back-propagating gradients to be zeroed out, when the gate is closed. Implementation of this gating mechanism is inspired by LSTM and GRU cells used in recurrent neural networks (Hochreiter and Schmidhuber [1997], Cho et al. [2014]).

The output of the upsampling block and the skip connection is merged in an elementwise sum operation. The original U-Net paper uses concatenation (Ronneberger et al. [2015]), however Dorfer et al. [2017] have shown that using a sum instead speeds up training, while having minimal impact on model accuracy.

In all layers (except for the two output sigmoid layers) the exponential linear unit (ELU) activation function is used. Reasons for this are described in Section 6.2.6.

5.2 Combining Datasets

We need to slice and combine described datasets in various ways, depending on the performed experiment. There are two operations we want to perform:

- Split an existing dataset into multiple slices (e.g. validation, supervised and unsupervised slices).
- Combine two different datasets together.

The splitting is needed for an experiment, that gradually increases the amount of unlabeled data and measures the change in model accuracy. This means the splitting logic has to be stable with respect to the size of the unlabeled slice:

- We do not want the content of other slices to change, when we change the size of the unlabeled slice.
- We want the unlabeled slice to only gain new data, not to be completely resampled.

This stability is achieved by giving consideration to the splitting logic (sampling the unlabeled data as the very last slice) and to the shuffling logic (the data is shuffled prior to splitting). All the randomness in the process is controlled by a parameter called *dataset seed*, which is distinct from the plain *seed* used for other initialization (model weights and noise generation). The introduction of the *dataset seed* is essential for isolating our experiments from variability introduced by differential dataset sampling.

One additional consideration has to be made when sampling datasets CVC-MUSCIMA and MUSCIMA++. Because their content consists of music transcribed by 50 different writers, we want our splits to keep these writers separate. Another words, we do not want a single writer to be present in both the training set and the validation set. We enforce this constraint to better measure the ability of the trained model to generalize. The official test set of MUSCIMA++ built with this approach is called the *writer-independent test set*, therefore we will call this property of our splitting logic *writer-independence*.

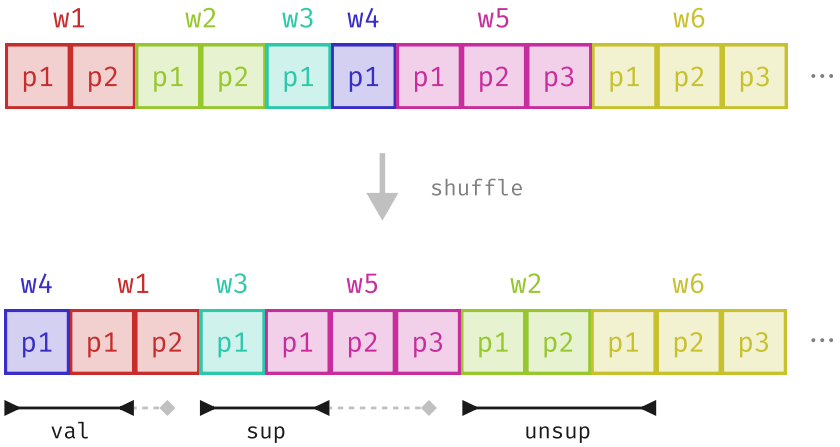


Figure 5.3: Visualization of how a dataset is split in the writer-independent fashion, if we request 2 validation, 2 supervised and 3 unsupervised pages. (w) writer, (p) page

When combining datasets, we run into the problem of image resolution. Since our proposed architecture learns visual features in a scale-dependent way, we need to bring all datasets to the same resolution. Datasets CVC-MUSCIMA and MUSCIMA++ have the same source images, so the problem is not present here, however, the dataset DeepScores is sampled at a lower resolution. In paper printing, a unit of DPI (dots per inch) is used. We cannot, however, use this unit,

since it assumes the music from both datasets has the same size when printed on paper. A more general approach is to define a unit, that could be derived from the raster image itself, without any knowledge of the scanning/printing DPI. The open format MusicXML[1] uses a spatial unit defined as *one tenth of interline space* (the distance between two adjacent stafflines). We took inspiration from this and defined a unit we call DPSS (dots per staff space). DPSS of a raster music document is the number of pixels between two neighboring stafflines. We measured the DPSS of CVC-MUSCIMA to be `28.75 px` and of DeepScores to be `16.0 px`.

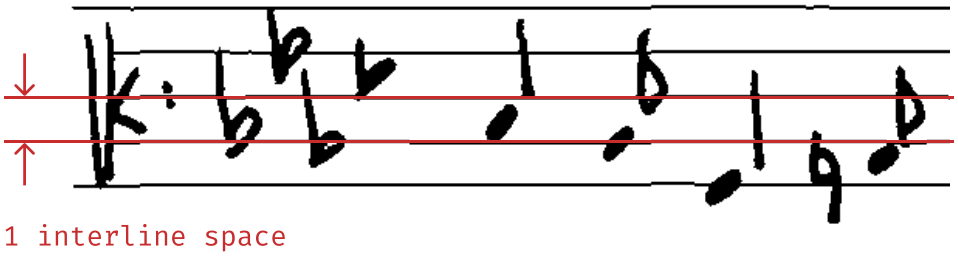


Figure 5.4: The spatial unit *interline space*, size of which in pixels we call DPSS (dots per staff space).

When matching the resolution of both datasets, we need to perform image resizing and we need to choose the target size and the interpolation method. We ended up upscaling the DeepScores dataset to match the resolution of the CVC-MUSCIMA dataset with the bilinear interpolation method.

In the beginning, we tried downscaling CVC-MUSCIMA with the area interpolation method. We chose the area method because it does not produce aliasing artifacts (it works by averaging pixel values over an area). It, however, introduced problems with thin lines (such as stafflines and stems) that faded below 50% intensity on most pixels. That caused the pixelwise F1 score we use to evaluate the experiments to become very noisy and difficult to interpret.

5.3 Noise Generation

Typical denoising autoencoders are used either for removing noise from existing images (say, an overlayed text, sensor noise, or aliasing artifacts) or for learning abstract features (Mao et al. [2016]). If noise is sufficiently fine, so as to not obstruct the high-level structure of the image, the autoencoder can learn this high-level structure in an unsupervised way. This setup can be used for feature extraction, similar to how variational autoencoders are used (Kingma and Welling [2014]).

Our goal is to leverage this training scheme to force our U-Net model to learn high-level representations. We chose to use noise in the form of dropping out parts of the input image – setting the image pixels to zero. The parameter that specifies the percentage of the zeroed-out image area is called *noise dropout*.

[1] `https://www.musicxml.com/`

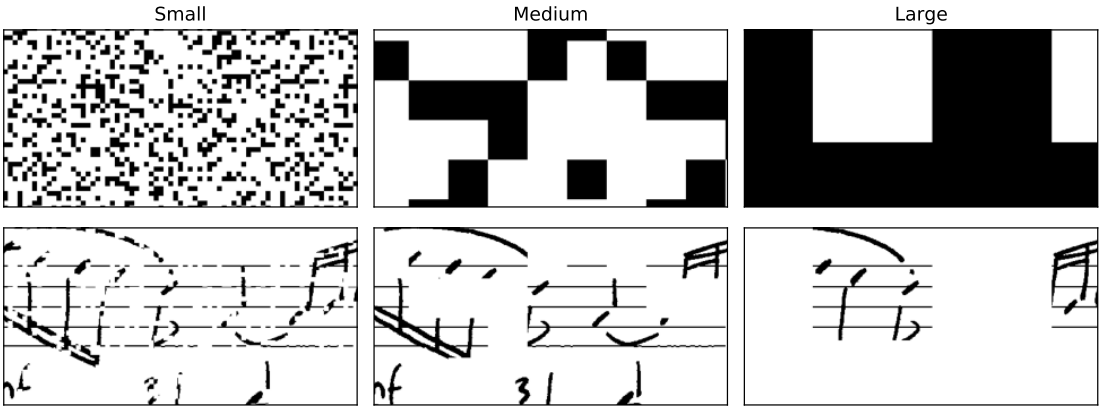


Figure 5.5: Comparison of various noise sizes. Small noise is easy for the model to remove and large noise leaves too many possibilities of reconstruction.

We could decide for each pixel independently, whether it should be dropped, but that would produce a very fine noise. We believe that such a noise would not force the model to learn large symbols, as they are unnecessary for removing the noise. Therefore we chose to increase the resolution of the noise to form a square grid, where one square covers approximately two staff spaces. At this size, dropped regions are large enough, so simple operations like dilation will not reconstruct them. These regions are also small enough that the number of plausible reconstructions within the dropped space is reasonably limited. An extreme situation would be dropping the entire input image, but there the space of possible reconstructions would be so large, that it would be impossible for the model to guess the correct one (even among the space of plausible reconstructions).

We got the idea of dropping image squares after reading the article by Brandenbusch et al. [2021]. The authors of the article present a method for reconstructing parts of an input image using a generative adversarial network. Their goal is to create synthetic training data for the task of cuneiform sign recognition.

5.4 Evaluation Metrics

Semantic segmentation is closely related to object detection and recognition. It is often performed as the first step from which object bounding boxes are computed. The segmentation output is first binarized by applying a threshold. Adjacent positive pixels are merged to form connected components, where each can then be enclosed in a bounding box. The described process is very basic and there are ways to improve it. A slightly more advanced process in the context of notehead recognition is used in the article by Dorfer et al. [2017].

There are many metrics that evaluate the correctness of predicted bounding boxes, comparing them to ground-truth bounding boxes. A metric called intersection over union (IOU) can be used to calculate the overlap of two bounding boxes. Setting a threshold on IOU and computing it between all pairs of predicted and ground-truth bounding boxes lets us compute the binary classification confusion matrix. From this matrix, additional metrics could be computed, such as precision, recall, and F1 score. By introducing a confidence level for each bounding box, or considering the performance over multiple classes, we get a large number

of metrics that can be used to assess various aspects of the recognition. An article by Padilla et al. [2021] provides an extensive comparison of such metrics.

Even though these object detection metrics tell more about the actual usefulness of the model (we care about detected objects, not pixels), we chose to evaluate our experiments directly at the pixel level. Adding these evaluation metrics adds unnecessary complexity to our experiments and it is not needed for our goal – measuring the impact of unsupervised data.

We chose to use the pixelwise F1 score at 0.5 thresholding. The F1 score is computed as the harmonic mean of precision and recall, so that it is high when both precision and recall are high. It is computed from the confusion matrix via the following formula:

$$F1 = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}$$

The confusion matrix values are computed over all pixels of the entire (validation) dataset.

It is important to note, that the two articles regarding music symbol detection (Hajič jr. et al. [2018], Dorfer et al. [2017]) use object detection F1 score, which is not directly comparable to our pixelwise F1 score.

6. Experiments and Results

This chapter begins with a detailed description of the training process. Next, there is an in-depth exploration, discussion, and experimentation regarding major hyperparameters, where each section concludes with the choice of a default value for that parameter. Finally, each of the last three sections presents an experiment we performed:

- Measure the impact of adding different amounts of unsupervised data during training (Section 6.3).
- Measure the impact of model size (number of synapses) on the task performance, when training on all available handwritten data (Section 6.4).
- Attempt to learn the segmentation task on printed music and evaluate it on handwritten music, while using unlabeled handwritten music as regularization (Section 6.5).

6.1 Training

There are many ways in which a semi-supervised generative model can be trained. One option is to pre-train a model in the unsupervised mode and after that train a separate supervised part of the model (Kingma et al. [2014]). Another option is to run the supervised training as a fine-tuning step after the unsupervised training, similar to the transfer learning approach. We chose to train both the supervised and the unsupervised tasks simultaneously, using a single, composite loss function.

One training step has the following structure:

- An input image is passed through the network and the produced segmentation is compared to the gold one, producing the supervised loss.
- An input image with noise added is passed through the network and the produced reconstruction is compared to the clean input image, producing the reconstruction loss.
- The reconstruction loss is multiplied by a constant hyperparameter we call *unsupervised loss weight*. Both losses are summed to produce the composite loss.
- The composite loss is used to calculate model gradients.
- An optimizer uses these gradients to update model weights.

The input to the training step is a composite batch of (x_l, y_l) labeled data pairs and (x_u, y_u) unlabeled data pairs.

- x_l are input images for segmentation
- y_l are gold segmentation masks

- x_u are noised input images for reconstruction
- y_u are cleaned target reconstruction images

The number of labeled and unlabeled pairs in a composite batch differs and their ratio is related to the ratio of labeled to unlabeled data in the dataset. The hyperparameter *batch size* dictates the total number of labeled and unlabeled pairs in one composite batch.

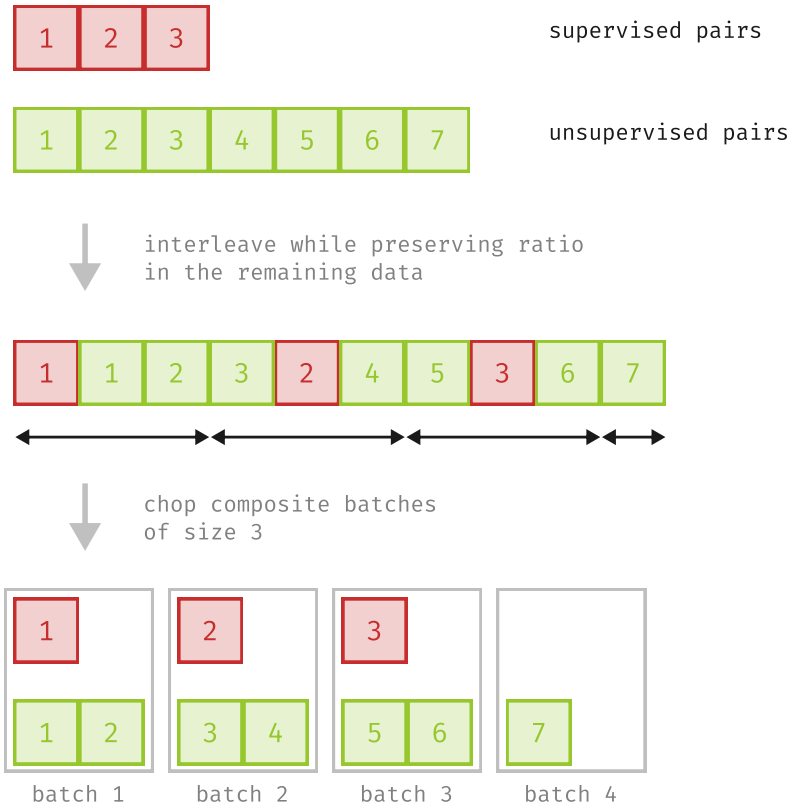


Figure 6.1: Illustration of how are supervised and unsupervised training pairs combined to form composite training batches.

Composite batches are created by taking training pairs from the labeled and unlabeled splits of the dataset. When the next pair is taken, we either take it from the labeled or the unlabeled split, whichever is larger, with respect to their target ratio. For example, if their ratio in the whole dataset is 1:5 (labeled to unlabeled) and we have 10:201 remaining pairs, we will take an unlabeled pair next. We take pairs until we have *batch size* number of them and then we send them to the training logic as a new composite batch.

To be able to batch together the training images, they need to have the same spatial dimensions. For this reason, we do not train on the entire music pages, but instead on tiles of fixed size. The size is set to 512 pixels wide and 256 pixels high for all experiments. We took inspiration for this from the article by Hajič jr. et al. [2018]. Tiles are sampled from a music page at random locations and the number of tiles is equal to the area of the page, divided by the area of the tile. The article also employs a technique called oversampling, where a tile is

sampled up to five times if it contains no pixel of the target segmentation class. This technique helps with training on rare symbols (e.g. clefs), where most tiles would contain no instance of the target class. Our training logic also implements this oversampling technique.

The semantic segmentation task is usually framed as a pixelwise classification problem. For this reason, we use the pixelwise binary cross-entropy loss function (Goodfellow et al. [2016]). To make the loss value invariant to changes in *batch size* and *tile size*, we compute the average over all pixels in a batch. This allows us to sum the supervised and unsupervised losses, as they are of a similar magnitude, regardless of the number of labeled and unlabeled training pairs used. The reconstruction task will also use the same pixelwise binary cross-entropy loss function since the reconstruction of a single pixel can be understood as the probability of that pixel being black.

We use the Adam optimizer for all experiments, with learning rate 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\varepsilon = 10^{-7}$ (Kingma and Ba [2014]). Also, for all experiments, we use the validation dataset to assess the performance of the model and train for as long as it is improving. Any evaluation of any trained model is performed by the model with the optimal validation dataset performance (unless the entire validation training curve is shown).

6.2 Understanding Hyperparameters

6.2.1 Batch Size

In the deep learning field, it is known that having a small batch size makes the training fast and noisy, whereas a large batch size makes it more stable at the cost of being slower (Goodfellow et al. [2016]). Since our model is fully convolutional and we train it on image tiles of fixed size, we can consider the size of these tiles to be a parameter similar to batch size. It also regulates the amount of data used for gradient estimation. If these tiles are large enough, we can get away with a batch size of 1 (this is what the original U-Net article does (Ronneberger et al. [2015])).

Using such a small batch size is, however, not possible in our case. Our training process expects batches containing both labeled and unlabeled data. The batch size determines the total number of these two kinds of data items in the entire composite batch. The ratio of these two item types within the composite batch is dictated by the ratio within the whole dataset. So if our dataset has, for example, 1:5 labeled to unlabeled data, the batch size has to be at least 6. Otherwise, we will start getting batches that contain only unlabeled data. This rule is not as strict, since the model would probably learn both tasks even if half of all batches were missing labeled data, however, if the imbalance becomes too severe, the training fails.

An example of such a failing training can be seen in Figure 6.2. The model learns to perform reconstruction even for the segmentation task. This is understandable since the two tasks are differentiated only at the last layer (1x1 sigmoid convolution). If all second-to-last layer activations contain image reconstruction data, then any 1x1 convolution combination of them will do as well.

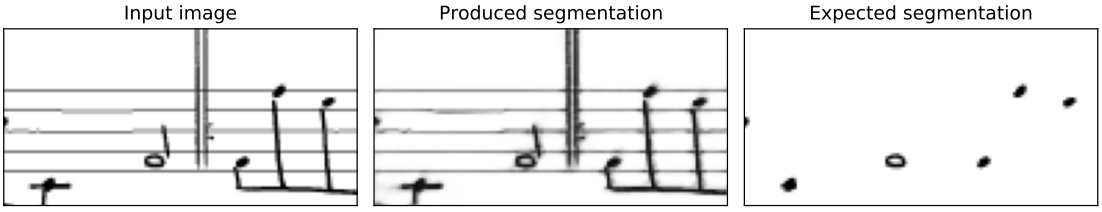


Figure 6.2: When batches are too small, some of them will lack labeled data and the model will fail to learn the supervised segmentation task. Instead, reconstruction output leaks into the segmentation output. This example has batch size 2 and a 1:10 ratio.

Since all of our experiments have training data ratios between 1:0 and 1:10, we chose to set the batch size parameter to 10.

6.2.2 Dropout

The network architecture, as described in Section 5.1, contains a dropout layer within every convolutional block (Srivastava et al. [2014]). We use the dropout when training on small datasets (experiment 6.3), because the training is otherwise very noisy and it is difficult to infer any measurable differences between runs. We use dropout as a means to stabilize the training, which in this setting also slightly increases model accuracy.

When training on larger datasets, the training is no longer unstable and dropout is not needed (Section 6.4 and 6.5). In fact, it causes the training process to converge much slower (2x or more) and it does not increase model accuracy anymore.

Both the original U-Net article (Ronneberger et al. [2015]) and the article by Hajič jr. et al. [2018] use the U-Net architecture without any dropout. In fact, an article by Tompson et al. [2015] argues, that using traditional dropout on convolutional layers may not be ideal. This agrees with our findings, that dropout helps only in very specific circumstances.

It may be the case, that using batch normalization instead or dropout (like Hajič jr. et al. [2018]) has the same effect of regularizing the network. However our goal is not to find the optimal architecture, but to measure the impact of unsupervised data. For that reason, we did not explore this option further.

From all this, we conclude that dropout should be disabled by default.

6.2.3 Skip Connections

In the chapter about our architecture (Section 5.1) we described a method for disabling skip connections during reconstruction training. We present three modes for skip connections:

- solid – skip connections are always enabled, like in a typical U-Net architecture
- gated – skip connections are enabled for the supervised task (training and inference) and disabled for the reconstruction task (training and inference)

- none – skip connections are always disabled, like in a typical autoencoder

The motivation behind gating skip connections is to bring the U-Net architecture (Ronneberger et al. [2015]) closer to a more traditional fully-convolutional autoencoder architecture (Bank et al. [2020]).

To measure the effect of skip connections, we propose the following experiment. We take the 16 *inner feature*, semi-supervised model from experiment 6.4 and train it with various skip connection settings. Training curves are shown in Figure 6.3.

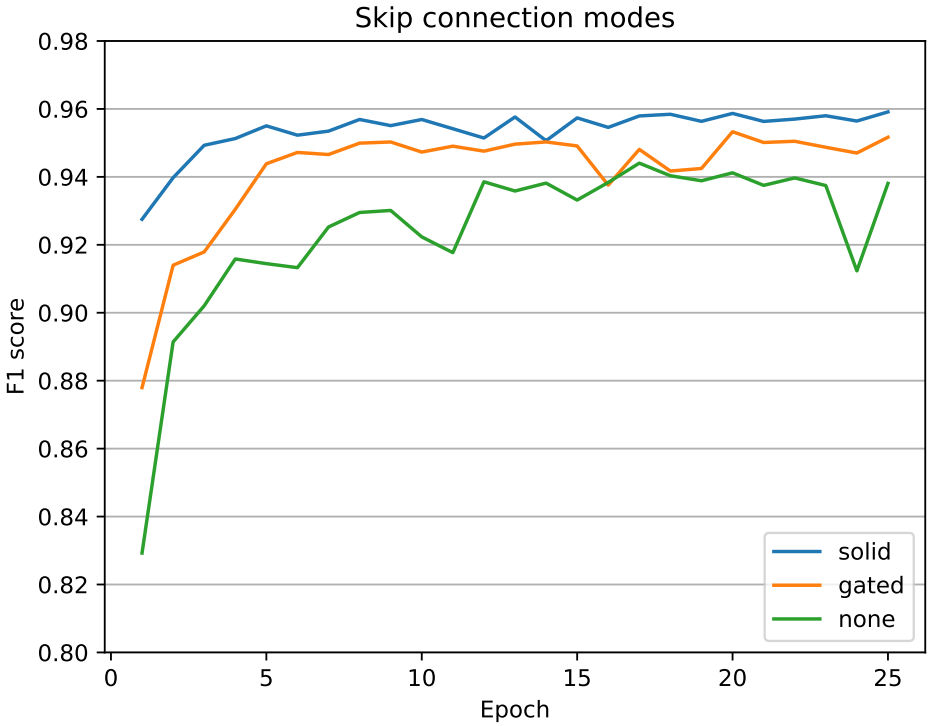


Figure 6.3: Comparing various skip connection modes with a model from experiment 6.4 (16 inner features, semi-supervised). The permanent skip connection mode performs the best, although all runs are below the supervised baseline. The order could be different, should the baseline be exceeded.

We can see that solid and gated modes have the best performance. The model with no skip connections is clearly worse at the segmentation task. This result validates the superiority of the U-Net architecture for semantic segmentation.

The gated mode seems to perform worse than the solid one. Despite this, we chose to train all the experiments in this gated mode as we believe it helps the model to learn representations during unsupervised training.

We have to point out, that all of our semi-supervised experiments never exceeded their supervised baselines. This means that bringing the model closer to its supervised setting (using solid skip connections) should increase its performance because we are bringing it closer to the baseline. For this reason, we cannot conclude, what is the best skip connection mode, should the baseline be exceeded. The order of modes may be in such a case reversed. This is another

reason why we chose to use gated connections, despite not being optimal in this experiment.

6.2.4 Unsupervised Loss Weight

This parameter determines the relative magnitude of the segmentation and the reconstruction loss. We first thought this parameter would have a significant impact on the training and would need to be tuned carefully. It turned out not to be the case. We ended up setting it 1 and not optimizing it.

Varying this parameter from 10 to 0.1 has minimal impact on model performance. When we examined inferred segmentation masks and reconstructed images, as they evolve throughout the training process, we discovered that this parameter primarily impacts the speed at which the two tasks are learned relative to each other. When the parameter is 10, the reconstruction task is learned quickly and the segmentation task is learned slowly. When we set it to 0.1 the situation is reversed. If we let the training converge, it ends up learning both tasks equally well.

Interesting behavior happens only at the extreme. When the parameter is set to zero, the reconstruction task produces no gradients, and the training collapses to the fully-supervised regime. We performed an experiment that verified this behavior.

6.2.5 Noise Parameters

There are two major noise parameters that can be varied:

- Noise resolution (size of dropout squares)
- Noise dropout ratio (ratio of erased input pixels)

In the section on noise generation (5.3) we gave reasons for using a larger noise resolution (but not too large). We believe large tiles force the model to learn larger symbols and thus the corresponding high-level representations. This hypothesis is supported by images shown in Figure 6.4.

The first two images show, how the model learns to remove small noise easily. The model, however, does not produce satisfying reconstructions when applied to an image with larger noise tiles dropped (as seen in the second pair of images). We can see that:

- Edges of noise tiles are still visible in the reconstruction.
- Reconstructed stafflines fade when they are close to other symbols.
- Most reconstructions are based on relatively low-level features. The beam corner is extended upwards, which is reasonable in the sense of drawing straight lines, but not in any musical sense.

Comparatively, the lower two images show a much better reconstruction. The lower reconstruction was trained with a noise size equal to the size seen in the lower-left image. Reconstructed stafflines are sharp even when intersecting the

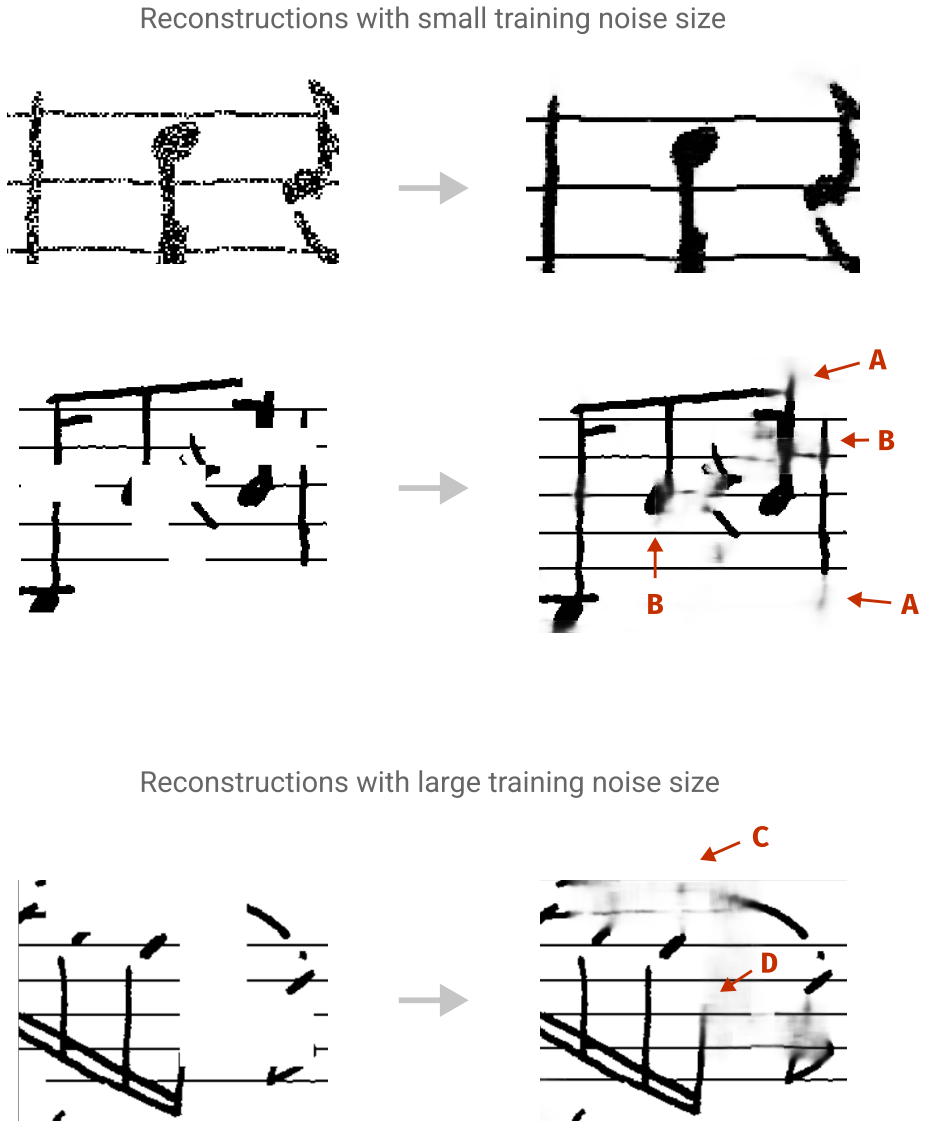


Figure 6.4: Comparing reconstructions performed by models trained with small and large noise tiles. (A) Low-level reconstructions, make no sense in the musical context. (B) Edges of dropped tiles leaking into the reconstruction. (C) High-level reconstruction with staffliens and stems visible. (D) Forgotten notehead.

eighth note and there are no tile edges visible (reconstructions blend well with the known image). The model has even managed to conjure up a fake set of stafflines and stems, that connect well to visible noteheads (pointer C). This shows that the model learns much higher-level features than just simple geometric shapes (lines, circles).

We ended up training all experiments with noise tiles of size equal to two staff spaces and a noise dropout ratio of 25%. We experimented with noise dropout of 50%, but the performance seemed not to change. This chosen size and ratio can be seen in the last row of Figure 6.4.

6.2.6 Activation Function

We first used the ReLU activation function (rectified linear unit) in all convolutional layers (except the final sigmoid layer), just like it is used in the original U-Net article (Ronneberger et al. [2015]). However, we occasionally encountered problems with convergence. Replacing the activation function with ELU (exponential linear unit) solved these issues. We took inspiration from Dorfer et al. [2017], who also use the ELU activation function. The difference between the two can be seen in Figure 6.5.

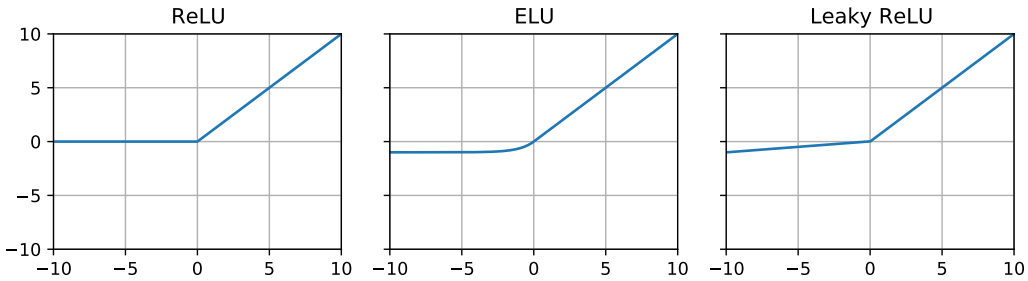


Figure 6.5: Visualization of explored activation functions. ReLU is flat in negative values and therefore has no gradient there. ELU has an exponentially decaying gradient and leaky ReLU has a constant gradient. The parameter for the displayed leaky ReLU is 0.1 to make its shape more apparent.

The convergence problems were happening at the very beginning of training. The model quickly learned to output a completely black image and never recovered from that state. We think it was an instance of the ”dying ReLU” problem (Lu et al. [2020]). When the model is first initialized, it outputs a gray-ish image, since model weights are drawn from a uniform distribution centered on zero and the final sigmoid layer turns that into a 0.5 gray. Because our target images have a black background, the model first learns to produce mostly black images. Only then does it learn to output white pixels as well (see Figure 6.6). With ReLU, the first training phase probably overshoots into the negative range of most synapses and that causes the model to get stuck in that negative range with zero gradients.

Interestingly enough, this problem happens only when training in the fully-supervised mode. We have never encountered it when training in the semi-supervised mode. This again suggests that unlabeled data act as regularization, damping any extreme gradients, and stabilizing the training.

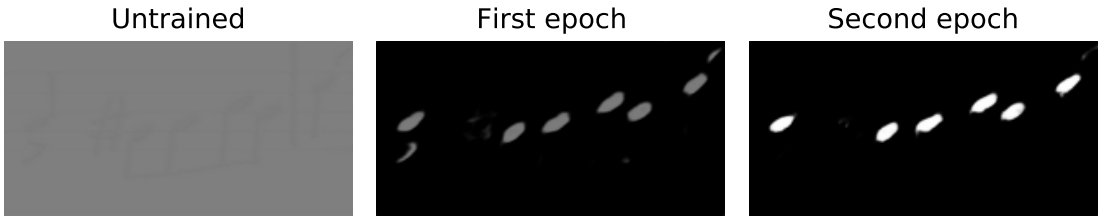


Figure 6.6: The training process starts by learning to output a mostly black image, which probably causes the model to overshoot during the fully-supervised training and get stuck in the "dying ReLU" problem.

We also tried using the leaky ReLU function with parameter $\alpha = 0.01$, however the problem still remained. Maybe a larger value for α could help, although we already knew that ELU works, so we haven't explored this further.

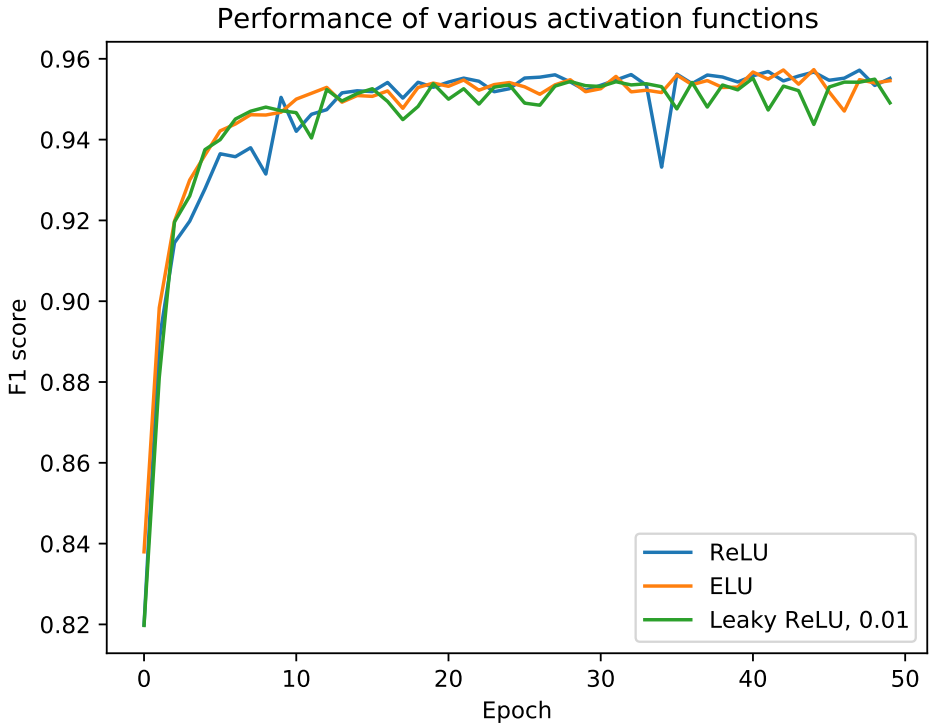


Figure 6.7: An experiment from Section 6.4 with 8 inner features, trained in fully-supervised mode with various activation functions. All runs show the same performance.

We run one of the experiments from Section 6.4 with all proposed activation functions to see what impact it has on model performance (Figure 6.7). We can clearly see that they all perform equally well, so we choose ELU as the only activation function that does not suffer from the convergence problem. We thereby validate the work of Dorfer et al. [2017] (their article does not provide an explanation for the use of ELU, but we believe they must have encountered these exact same problems).

6.3 Semi-supervised Improvements

The main hypothesis this work is attempting to validate is that adding unlabeled data to the training process helps. We primarily want to improve model accuracy, but as we will see, this is not what our experiments suggest. They do, however, show improvements in other areas, such as training stability and reduced overfitting (Section 6.4).

In this first experiment, we test how various labeled to unlabeled data ratios affect the training process. The experiment uses the MUSCIMA++ dataset (Hajič jr. and Pecina [2017]):

- 10 pages act as the labeled set.
- 0, 5, 10, and 50 pages act as the unlabeled set.
- 10 pages act as the validation set.
- All of these pages come from the writer-independent train set of MUSCIMA++ and are chosen in a writer-independent manner (all the splits contain pages by different writers).

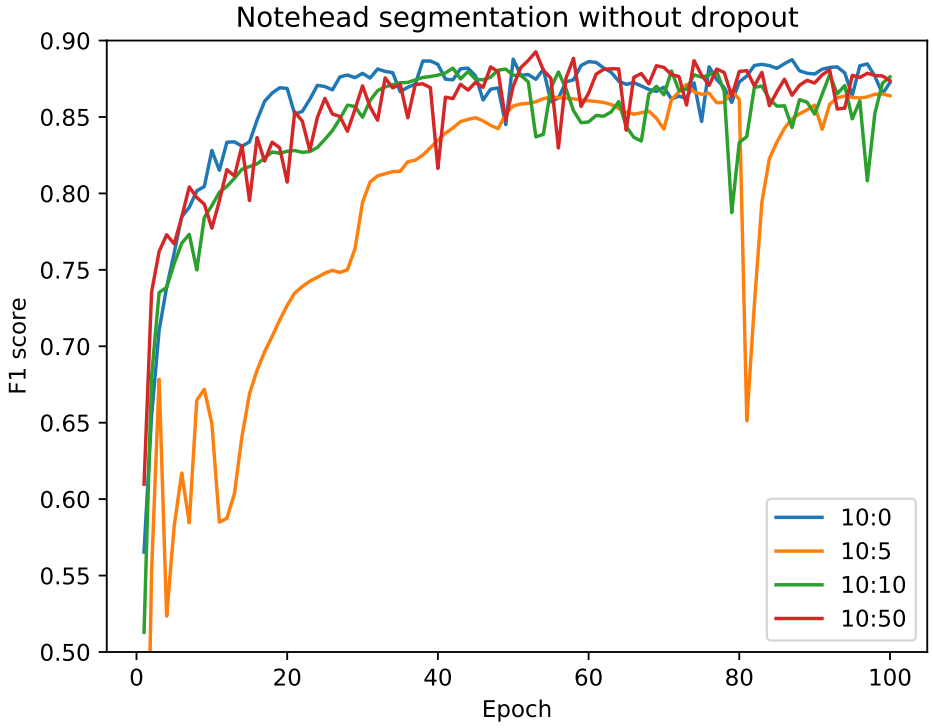


Figure 6.8: Training on a small dataset without dropout is noisy, see the orange line at the beginning and the green line at the end.

The learned task is notehead segmentation (both full and empty noteheads). Noteheads are an ideal symbol for this kind of measurement. Firstly, they are very abundant. Each page of the dataset contains many instances of them and they are evenly scattered over the whole page. If we were to instead detect more

rare symbols (such as clefs or rests), it could skew the results, making it difficult to separate the effects we want to measure. Handwritten noteheads are also very diverse in style, making them more interesting to learn (compared to, say, stafflines).

All model hyperparameters are set to sensible defaults. The derivation of these values has been described in Section 6.2. The model capacity, described by the *inner features* parameter is set to 8, which is useful to know for comparison with the next experiment. The proposed dataset is rather small and so the training is very noisy (Figure 6.8). To stabilize the training we set the dropout parameter to 50% (Srivastava et al. [2014]).

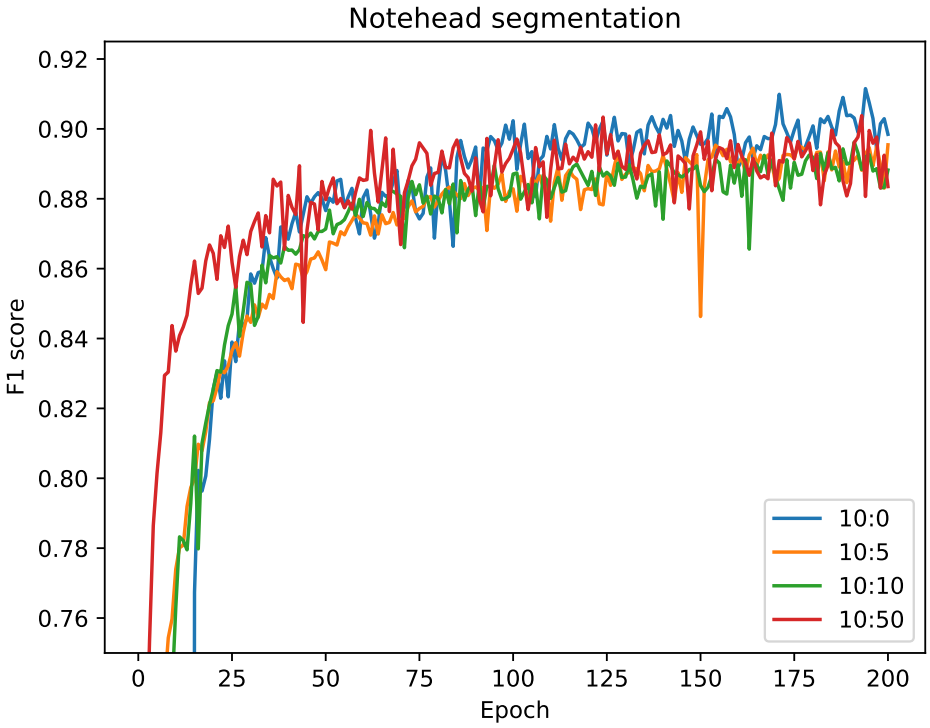
We expect that as we add more and more unlabeled data, the F1 score should reach higher and higher. Or at least not get worse. This is not what we see in Figure 6.9. The fully supervised model outperforms all the others by a clear margin.

Focusing only on the semi-supervised models, it seems that adding more unsupervised data maybe helps here, although the three lines end up on top of each other at the epoch 200. A better idea is to look at Figure 6.10. The chart contains evaluation results on the test set of six runs of each configuration. We can clearly see how the performance rises with more unsupervised data. Unfortunately, it does not reach above the fully-supervised results. We, unfortunately, cannot push the amount of unlabeled data much higher, as it breaks our training process (see Section 6.2.1) and it likely also has diminishing returns. The actual numbers are summarized in table 6.1.

The reason for this drop in performance is probably caused by the fact, that the supervised model has to only learn one task – segmentation. Whereas the semi-supervised one has to also learn the unsupervised reconstruction task. This claim is explored in the next section and is supported by the fact that the performance drop disappears when we increase model capacity.

Page type	**Page count**			
Labeled	10	10	10	10
Unlabeled	0	5	10	50
Seed	**F1 Score (%)**			
0	92.73	90.87	91.08	92.38
1	93.02	90.36	90.13	91.24
2	93.54	90.05	89.88	91.56
3	93.34	90.99	89.65	92.63
4	93.90	90.58	90.99	90.95
5	93.00	90.54	91.43	92.37
mean	93.25	90.56	90.53	91.86
std. dev.	0.39	0.31	0.67	0.64

Table 6.1: Values plotted in Figure 6.10.



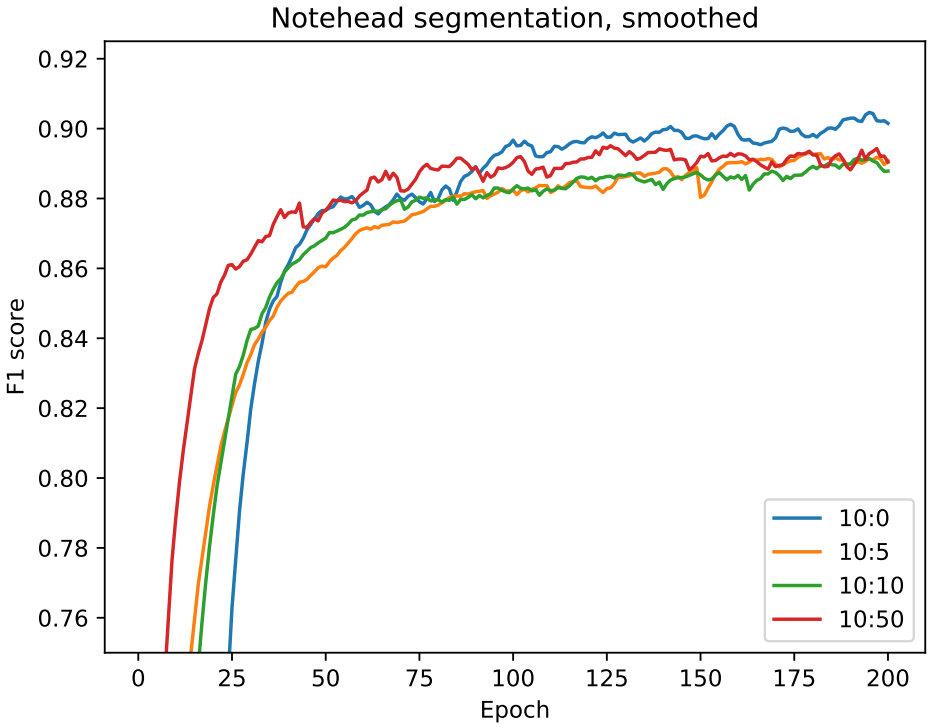


Figure 6.9: Training curves of semi-supervised models with different ratios of supervised and unsupervised data. The bottom image is smoothed using an exponential moving average. The fully-supervised baseline (blue) clearly outperforms all semi-supervised models.

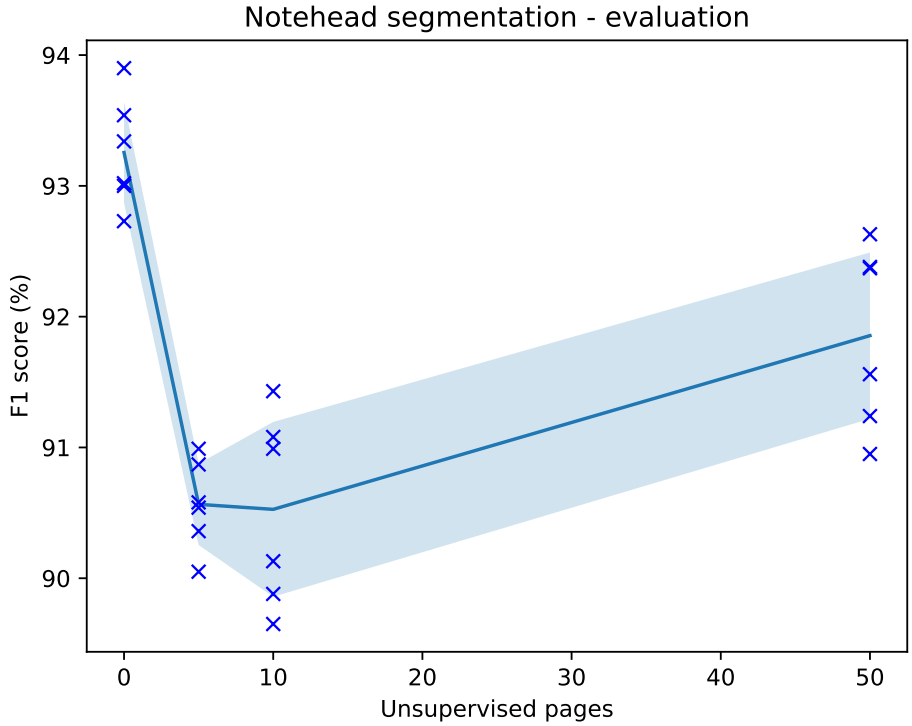


Figure 6.10: Notehead segmentation performance on the MUSCIMA++ writer-independent test set, when training with 10 labeled pages and a variable number of unlabeled pages. Adding more unlabeled data helps, but never outperforms the fully-supervised baseline.

6.4 Utilizing CVC-MUSCIMA

This experiment attempts to address issues of the previous experiment:

- fixed model capacity
- small dataset

In Section 3.2 we described the two major datasets for handwritten music recognition: CVC-MUSCIMA (Fornés et al. [2011]) and MUSCIMA++ (Hajič jr. and Pecina [2017]). The dataset MUSCIMA++ is a highly annotated subset of CVC-MUSCIMA. We can view both datasets together as a single semi-supervised dataset, being 12% labeled and 88% unlabeled. To the best of our knowledge, nobody has yet tried to utilize both datasets simultaneously for semantic segmentation.

Dorfer et al. [2017] and Hajič jr. et al. [2018] have used the U-Net architecture (Ronneberger et al. [2015]) for segmentation and they trained it on the MUSCIMA++ dataset. Their results are very impressive. Being able to further build on their work and improve the model by utilizing unlabeled data from CVC-MUSCIMA would be very helpful for the field of OMR. This experiment attempts to do just that.

We take the whole CVC-MUSCIMA dataset, separate writers from the MUSCIMA++ independent test set, separate 20 pages for the validation set, and

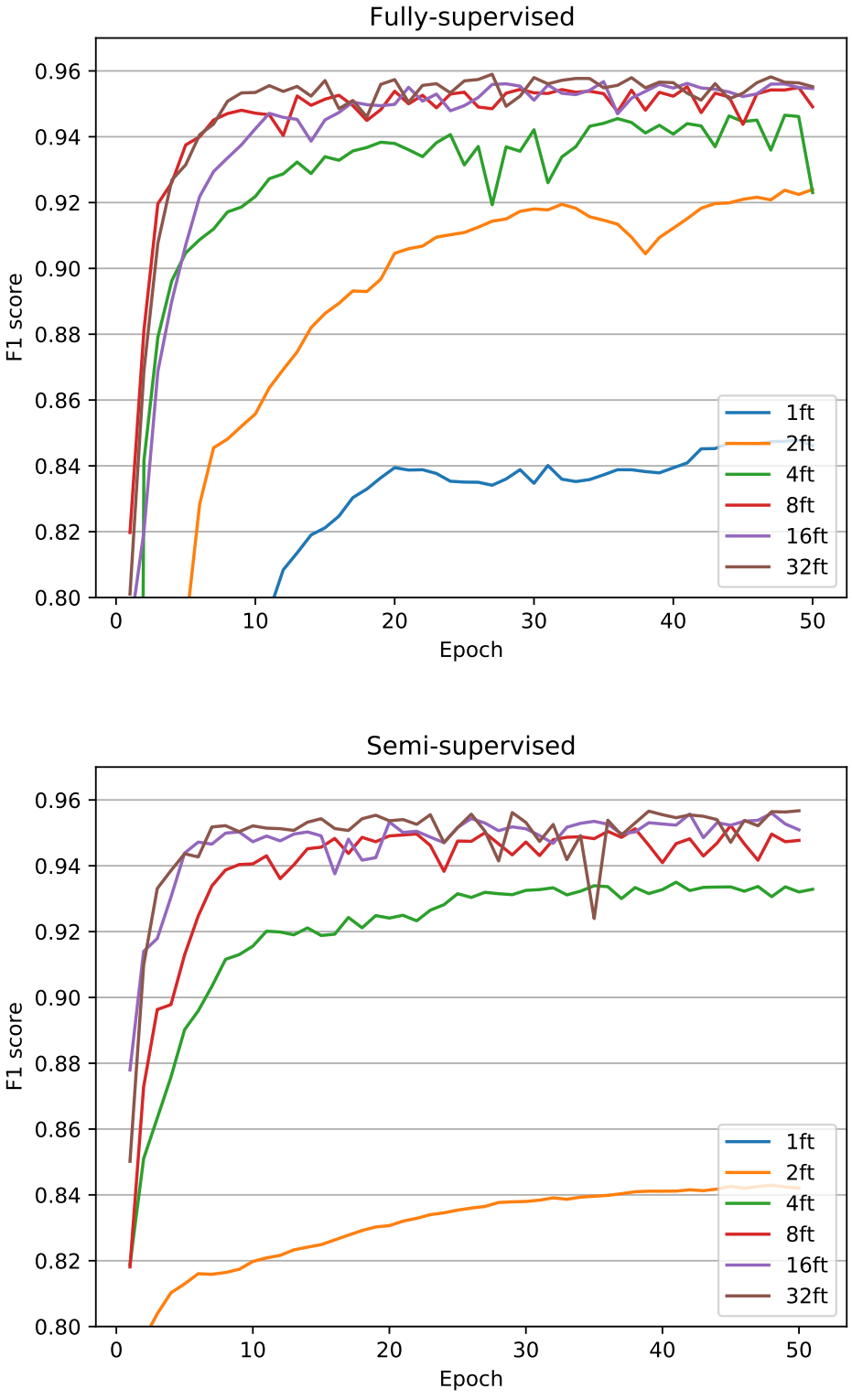


Figure 6.11: Comparison of fully-supervised and semi-supervised training regimes with models of varying capacity (inner features) on the CVC-MUSCIMA dataset. Both regimes get to the same performance, below the 96% line. Model 1 in the semi-supervised regime peaks below 80% line and is not visible in the chart.

remove other pages from these validation writers. Pages that remain are produced by writers not present in both the test set and the validation set. These remaining pages are partially contained in the MUSCIMA++ dataset (99 pages) and all the other pages are used as unlabeled data (551 pages). Therefore we train on 650 out of 1000 pages of the CVC-MUSCIMA dataset.

Since the dataset is now much larger than in the previous experiment (Section 6.3), we no longer need dropout layers. In fact, the training is even more stable without it and individual runs are clearly separated.

This experiment attempts to compare fully-supervised and semi-supervised models regardless of their capacity. We, therefore, train various model capacities (the *inner features* model parameter) and then compare the best ones for each setting.

Another difference to the previous experiment is that the ratio of labeled to unlabeled data is fixed and given by dataset sizes. The ratio of 99 to 551 pages corresponds best with the ratio of 10:50.

The validation dataset F1 score over the course of training can be seen in Figure 6.11. In these charts we can see:

- Models with 1 and 2 *inner features* are clearly underfitting in the supervised mode (compared to other models). When we add unlabeled data, their performance drops significantly, but the training curve gets much smoother.
- Models with 4 and 8 *inner features* worsen much less and also get smoother (especially 4 becomes much more stable).
- Model 16 no longer worsens, it is able to learn both tasks.

Conclusions can be drawn from these observations:

- The reconstruction and segmentation tasks clearly compete for model capacity. The performance drop due to adding unlabeled data decreases, as the model capacity increases.
- The addition of unlabeled data can be used as a regularization technique. This is evident from the fact that training curves get much smoother as we add unlabeled data (especially for models 2 and 4). A regularization effect is also described in the corresponding literature (Ouali et al. [2020]).
- All models come close to the 96% line but never cross it. While the semi-supervised models get as good as the fully-supervised, they never get better. It seems the reconstruction task is not learning any useful representations. But from Section 6.2.5 we know that some abstract representations are learned. Maybe these learned representations are not used during segmentation.

6.5 Knowledge Transfer

The goal of this experiment is to train a model to transfer a learned task from one domain onto another in a semi-supervised manner. The goal is to perform semantic segmentation on handwritten music while providing labels only for printed

music. We use the DeepScores v2 dataset (Tuggener et al. [2021]) as the source of labeled printed music and the MUSCIMA++ dataset (Hajič jr. and Pecina [2017]) as the source of unlabeled handwritten music. Validation is performed on a set of 20 pages from the MUSCIMA++ dataset. We use the remaining 99 pages for unsupervised training. To keep the ratio of labeled to unlabeled data at a reasonable value of 1:5, we use 20 labeled pages from the DeepScores dataset. We also train a supervised baseline with no unlabeled pages to compare against.

We first tried performing notehead segmentation, like in the previous experiments, but the results were extremely noisy and no meaningful difference could be seen. The pixelwise F1 score varied from 0.2 to 0.6 even between different runs of the supervised baseline (with dataset seed fixed). Semi-supervised runs were equally noisy and within the same range. When inspecting the inferred segmentation masks, we concluded that handwritten noteheads are overly varied and have a very different appearance compared to printed noteheads. The trained models could only reliably segment noteheads of writer 9, whose noteheads are very similar to printed ones.

Informed by this failure, we choose to segment stafflines. Firstly, because they are very easy geometric shapes to understand. Stafflines are the first objects that a model learns to repair during reconstruction and does so with great precision. Secondly, their visual appearance in the printed and handwritten datasets is very similar, but not identical. A challenge is still being presented (handwritten stafflines are much thinner and the surrounding symbols are more varied).

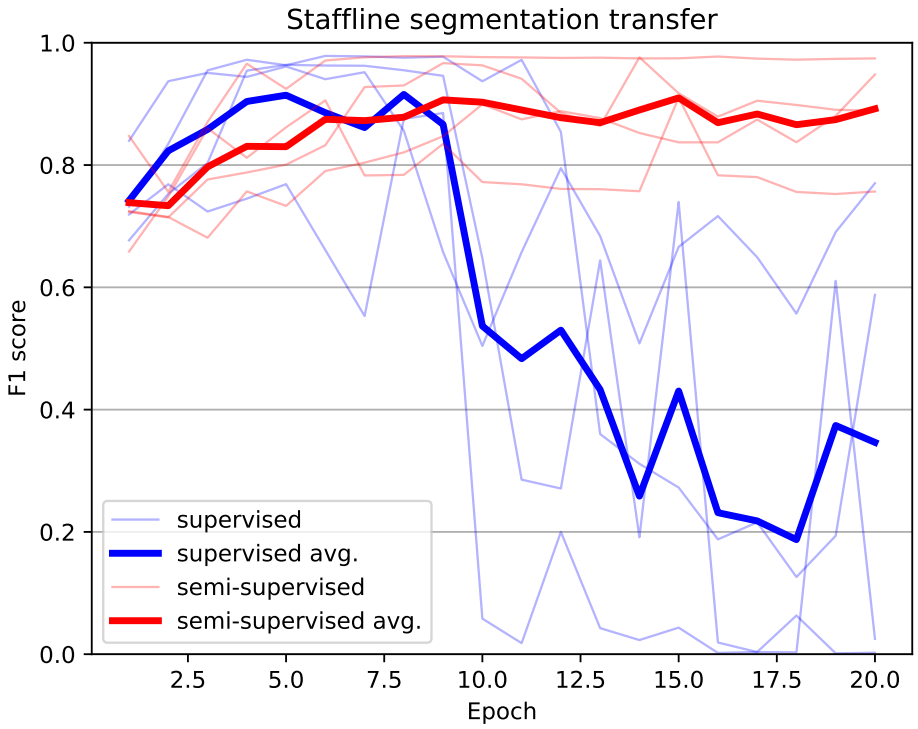


Figure 6.12: Handwritten staffline segmentation F1 score, when training on labeled printed music and unlabeled handwritten music.

We set model capacity to 8 *inner features* and train for 20 epochs. The

training process is still relatively noisy (like the failed notehead attempts), so we train both the supervised and semi-supervised variants 4 times each and compute an average. Results can be seen in Figure 6.12.

During the first 7 epochs, both variants perform equally well, however at about the tenth epoch all fully-supervised models start overfitting and their F1 score drops to very low numbers. Interestingly, all the semi-supervised models stay at a comparably high score. They seem to plateau, instead of overfitting. This is a clear example of how the unsupervised data can regularize the training and prevent the model from overfitting. Unfortunately, if we take the best checkpoints of both variants, we see that both variants perform equally well. The semi-supervised model does not surpass the supervised baseline.

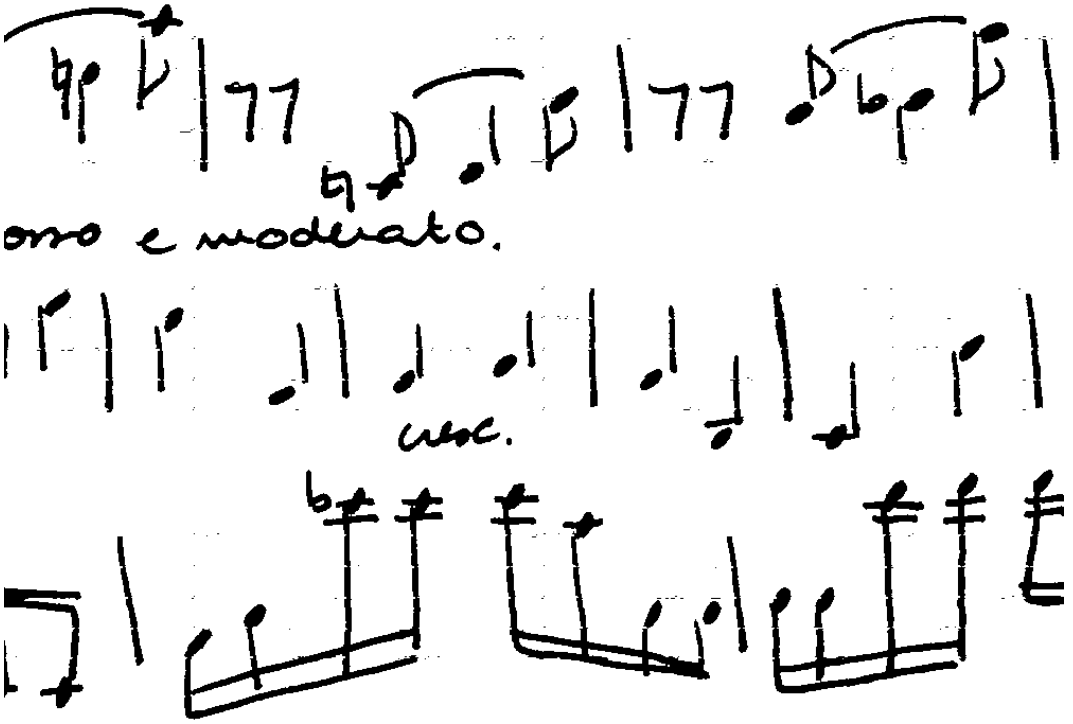


Figure 6.13: Handwritten stafflines were removed by the model trained on labeled printed music and unlabeled handwritten music. Tiny speckles in the image are the result of imperfect segmentation.

Evaluating the best semi-supervised model on the MUSCIMA++ test set (writer-independent) gives us a pixelwise F1 score of 97.70%. The same model peaks on the validation set at 97.81%.

We can compare ourselves to an article by Calvo-Zaragoza et al. [2017], focusing on staffline removal. The article presents a convolutional network trained in a supervised way on a dataset extracted from the CVC-MUSCIMA dataset (Fornés et al. [2011]). Table 4 in the article provides their pixelwise F1 score on undistorted images (99.25%) together with other older approaches (97.59% up to 98.51%). We can see that our results are in a range comparable to older approaches, which is impressive since we did not use a single annotated handwritten image. Our annotated training images were purely digitally constructed, without any attempts at mimicking the handwritten style.

7. Conclusion and Future Work

We have designed a scheme for training the U-Net architecture in an unsupervised manner and used it in a semi-supervised setting to try and improve optical music recognition. The unsupervised training scheme is successful at teaching the model useful representations. These are used during reconstruction to repair masked images of music sheets.

Using the model for semi-supervised learning improves training stability and the added unlabeled data regularizes the model, preventing it from overfitting. The performance of the semi-supervised model was able to reach the supervised baseline but was unfortunately unable to surpass it.

We were also able to replicate and validate the findings of other people, such as using the ELU activation function to prevent the dying ReLU problem.

By looking carefully at learned reconstructions, we can see that the model produces a lot of blurred areas in places where many viable reconstructions can be generated. We believe that if reconstruction is certain (e.g. extending stafflines), then it is not blurred. However, if a reconstruction may contain multiple viable possibilities (e.g. noteheads may have many shapes, a note can have many pitches within the space of the masked area), then the model does not pick one reconstruction but instead generates a blurred average of all of them.

We believe that replacing the simple reconstruction loss function with a discriminative network could force the model to pick a specific, realistic-looking reconstruction, instead of producing a blurred average of many reconstructions. This would reframe the reconstruction task as a generative adversarial network (GAN) problem, rather than an autoencoder problem (as it is framed now). This might be a more appropriate setting since there is no encoded latent representation from which the image is reconstructed. It is being reconstructed out of nothing, with the only condition of matching the surrounding image.

This intuition is further supported by the article (Brandenbusch et al. [2021]), where the authors propose a modified GAN architecture for generating synthetic cuneiform tablets. Their model has two discriminators to force the generated tile to match the surrounding image and an auxiliary classifier that enforces the generation of the correct cuneiform symbol. We will explore this modified scheme in future work.

The goal of the thesis was to explore semi-supervised learning in the context of optical music recognition and to compare SSL results to a supervised baseline. We have fulfilled the goal by adapting a state-of-the-art object recognition architecture to the semi-supervised setting and evaluating it in three distinct experiments. We also explored and described all of its hyperparameters. We have not been able to surpass the supervised baseline, but we have identified possible problems and described the trajectory of our future work.

Bibliography

Yauh Babakhin, Artsiom Sanakoyeu, and Hirotoshi Kitamura. Semi-supervised segmentation of salt bodies in seismic images using an ensemble of convolutional neural networks. In *41th DAGM German Conference on Pattern Recognition (GCPR)*, pages 218–231, Dortmund, Germany, 2019.

David Bainbridge and Timothy Bell. The challenge of optical music recognition. *Computers and the Humanities*, 35:95–121, 2001.

Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders. *CoRR*, 2020. URL `https://arxiv.org/abs/2003.05991`.

Kai Brandenbusch, Eugen Rusakov, and Gernot A. Fink. Context aware generation of cuneiform signs. In *16th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, pages 65–79, Lausanne, Switzerland, 2021.

Jorge Calvo-Zaragoza and David Rizo. End-to-end neural optical music recognition of monophonic scores. *Applied Sciences*, 8, 2018.

Jorge Calvo-Zaragoza, Antonio Pertusa, and Jose Oncina. Staff-line detection and removal using a convolutional neural network. *Machine Vision and Applications*, 28:665–674, 2017.

Jorge Calvo-Zaragoza, Jan Hajič jr., and Alexander Pacha. Understanding optical music recognition. *ACM Computing Surveys*, 53, 2020.

Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. *Semi-supervised learning*. MIT Press, 2009.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *19th Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Quatar, 2014.

Li Deng. The MNIST database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29:141–142, 2012.

Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. In *5th International Conference on Learning Representations (ICLR)*, Toulon, France, 2017.

Matthias Dorfer, Jan Hajič jr., and Gerhard Widmer. On the potential of fully convolutional neural networks for musical symbol detection. In *14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, pages 53–54, Kyoto, Japan, 2017.

Alicia Fornés, Anjan Dutta, Albert Gordo, and Josep Lladós. CVC-MUSCIMA: A ground truth of handwritten music score images for writer identification and staff removal. *International Journal on Document Analysis and Recognition*, 15:243–251, 2011.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *27th International Conference on Neural Information Processing Systems (NIPS)*, pages 2672–2680, Montreal, Canada, 2014.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *17th International Conference on Neural Information Processing Systems (NIPS)*, pages 529–536, Vancouver, Canada, 2004.

Jan Hajič jr. and Pavel Pecina. The MUSCIMA++ dataset for handwritten optical music recognition. In *14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, pages 39–46, Kyoto, Japan, 2017.

Jan Hajič jr., Matthias Dorfer, Gerhard Widmer, and Pavel Pecina. Towards full-pipeline handwritten omr with musical symbol detection by u-nets. In *19th Conference of the International Society for Music Information Retrieval (ISMIR)*, pages 225–232, New York, NY, USA, 2018.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations (ICLR)*, San Diego, USA, 2014.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations (ICLR)*, Banff, Canada, 2014.

Diederik P. Kingma, Danilo J. Rezende, Shakir Mohamed, and Max Welling. Semi-supervised learning with deep generative models. In *27th International Conference on Neural Information Processing Systems (NIPS)*, pages 3581–3589, Montreal, Canada, 2014.

Lu Lu, Yeonjong Shin, Yanhui Su, and George Em Karniadakis. Dying relu and initialization: Theory and numerical examples. *Communications in Computational Physics*, 28:1671–1706, 2020.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian J. Goodfellow. Adversarial autoencoders. *CoRR*, 2015. URL `https://arxiv.org/abs/1511.05644`.

Xiaojiao Mao, Chunhua Shen, and Yu-Bin Yang. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In *29th International Conference on Neural Information Processing Systems (NIPS)*, Barcelona, Spain, 2016.

Jiří Mayer and Pavel Pecina. Synthesizing training data for handwritten music recognition. In *16th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, pages 626–641, Lausanne, Switzerland, 2021.

Yassine Ouali, Céline Hudelot, and Myriam Tami. An overview of deep semi-supervised learning. *CoRR*, 2020. URL `https://arxiv.org/abs/2006.05278`.

Alexander Pacha and Horst Eidenberger. Towards a universal music symbol classifier. In *14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, pages 35–36, Kyoto, Japan, 2017.

Alexander Pacha, Jan Hajič jr., and Jorge Calvo-Zaragoza. A baseline for general music object detection with deep learning. *Applied Sciences*, 8, 2018.

Rafael Padilla, Wesley L. Passos, Thadeu L. B. Dias, Sergio L. Netto, and Eduardo A. B. da Silva. A comparative analysis of object detection metrics with a companion open-source toolkit. *Electronics*, 10, 2021.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *4th International Conference on Learning Representations (ICLR)*, San Juan, Puerto Rico, 2016.

Ana Rebelo, Ichiro Fujinaga, Filipe Paszkiewicz, Andre R. S. Marcal, Carlos Guedes, and Jaime S. Cardoso. Optical music recognition: state-of-the-art and open issues. *International Journal of Multimedia Information Retrieval*, 1:173–190, 2012.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 234–241, Munich, Germany, 2015.

Harald Scheidl. Handwritten text recognition in historical documents. Master's thesis, Vienna University of Technology, 2018.

Jost Tobias Springenberg. Unsupervised and semi-supervised learning with categorical generative adversarial networks. In *4th International Conference on Learning Representations (ICLR)*, San Juan, Puerto Rico, 2016.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. Efficient object localization using convolutional networks. In *28th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 648–656, Boston, MA, USA, 2015.

Lukas Tuggener, Yvan Putra Satyawan, Alexander Pacha, Jürgen Schmidhuber, and Thilo Stadelmann. The deepscoresv2 dataset and benchmark for music object detection. In *25th International Conference on Pattern Recognition (ICPR)*, pages 9188–9195, Milan, Italy, 2021.

Chunlei Zhang, Qian Zhang, and John H.L. Hansen. Semi-supervised learning with generative adversarial networks for arabic dialect identification. In

44th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 5986–5990, Brighton, UK, 2019.

Xiaojin Jerry Zhu. Semi-supervised learning literature survey. *Technical report, University of Wisconsin-Madison Department of Computer Sciences*, 2005.