THÈSE DE DOCTORAT DE

L'INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

ÉCOLE DOCTORALE Nº 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par
Kwon-Young Choi

Combinaison de modèles génératifs non supervisés et d'une méthode syntaxique pour la détection de symboles musicaux avec peu de données annotées

Thèse présentée et soutenue à Rennes, le date
Unité de recherche : IRISA - UMR6074
Thèse Nº : 21ISAR 19 / D21 - 19

ACKNOWLEDGEMENT

Je voudrais tout d'abord remercier mon directeur de thèse, Bertrand Coüasnon, pour son soutien indéfectible et sans qui cette thèse n'aurais jamais pu voir le jour. De même, je voudrais remercier mes co-encadrant de thèse, Richard Zanibbi et Yann Ricquebourg, pour leur précieux conseils et plus particulièrement Richard pour son accueil très chaleureux au RIT.

Je n'ai pas de mots assez fort pour décrire à quelle point je suis reconnaissant à ma famille, qui m'ont soutenu, conseillé, écouté patiemment et même hébergé en ces temps difficiles ...

Une personne toute spéciale, à qui je dois l'achèvement de cette thèse, est ma fiancée Youna. Je réalise maintenant à quelle point il est important de pouvoir partager l'expérience difficile de la rédaction d'une thèse. Moi aussi, je serais là pour l'aider à finir la sienne.

Enfin, ces remerciements n'aurais pas de valeur si je ne mentionnais pas l'équipe Intuidoc, collègues et amis, pour nos débats effrénés et nos échanges si enrichissants.

Richard, thank you again for your advice, knowledge and wisdom. The warm welcome you gave me in Rochester is still one of my best memory of my PhD.

항상 저를 믿어주시고 응원해주시는 사랑하는 가족분들께 감사드립니다.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

List of Algorithms

RÉSUMÉ EN FRANÇAIS

Introduction

Une partition musicale est un type spécial de document permettant de retranscrire par écrit une chanson, une mélodie ou une musique. La reconnaissance et le traitement automatique de telles images de documents est appelé la reconnaissance optique de la musique (Optical Music Recognition ou OMR). Ce sous-domaine de la reconnaissance de document a été le sujet de travaux de recherche depuis les années 60 avec le travail de PRUSLIN [Pruslin 66] et a récemment vu un intérêt renouvelé à cause d'avancées en matière de vision par ordinateur principalement dans le domaine du Deep Learning.

Le processus de reconnaissance d'une partition musicale, en partant d'une image pour arriver à l'extraction d'informations musicales de haut niveau comme la hauteur et la durée des notes de musique, est composé de multiples étapes successives. Traditionnellement, les états de l'art comme celui de FORNÉS et al. [Fornés 14] distinguent trois étapes principales :

1. le prétraitement des images de partitions musicales ;
2. la localisation et la reconnaissance des symboles musicaux ;
3. la reconstruction de la notation musicale.

Dans ce travail, nous nous intéressons à la détection de symboles musicaux, étape charnière intervenant avant la reconstruction de la notation musicale, dans le but de localiser précisément et de reconnaitre la nature ou la classe des symboles. Cette étape de détection est d'autant plus importante que la reconstruction de la notation musicale utilise la position relative d'un symbole par rapport à un autre pour en déduire leur relation. De plus, cette étape de détection de symboles musicaux étant une tâche principalement graphique, elle a le plus à gagner des dernières avancées de Deep Learning dans le domaine de la vision par ordinateur.

Le type de partition sur lequel nous allons principalement nous intéresser dans ce travail sont les partitions imprimées historiques allant du XVIII^{e} au XX^{e} siècle. Nous nous intéressons plus particulièrement à ces partitions car elles présentent souvent une grande complexité dans l'utilisation de la notation musicale moderne avec parfois une grande densité de symboles dans des espaces très restreints. Ces documents ont aussi la caractéristique d'avoir été imprimés en utilisant un processus de gravure manuelle sur plaque de cuivre présentant des défauts de placements uniques à ces documents. Enfin, les partitions historiques ont souvent des défauts et du bruit dûs à l'âge de la partition et à la manière de numériser l'image de la partition. Toutes ces caractéristiques contribuent à des difficultés de segmentation car beaucoup de symboles musicaux se touchent et présentent des défauts de formes et de bruits. Beaucoup de travaux de recherche en OMR se sont concentrés sur la reconnaissance de partitions manuscrites et imprimées par ordinateur mais assez peu sur des partitions historiques imprimées.

C'est pourquoi nous nous proposons d'étudier dans ce manuscrit la détection de symboles musicaux dans des partitions historiques imprimées denses, bruitées et complexes. Dans un premier temps, nous étudierons la détection de symboles musicaux avec des modèles de Deep Learning supervisés de détection. Nous proposerons une nouvelle architecture de détection basée sur le Spatial Transformer mieux adapté à certaines situations de détection contrainte et nous comparerons cette nouvelle approche aux modèles de détection de l'état de l'art du Deep Learning.

L'utilisation de modèle de Deep Learning nécessite une large quantité de données manuellement annotées et dans le cadre de partitions historiques imprimées, il n'existe pas de jeu de données pour la détection de symboles antérieur à nos travaux. C'est pourquoi, dans un premier temps, nous proposons un nouveau jeu de données de détection de symboles musicaux et, dans un second temps, nous présenterons une nouvelle méthode de détection de symboles musicaux non supervisée utilisant uniquement des symboles isolés comme source d'information.

Détection supervisée de symboles musicaux

Dans cette première partie, nous explorons la détection supervisée de symboles musicaux car cette étape est centrale dans le processus de reconnaissance d'une partition musicale. Pour cela, nous nous intégrons dans une méthode préexistante de reconnaissance de partition musicale appelée DMOS [Coüasnon 01] basée sur une méthode syntaxique permettant de décrire la notation musicale moderne. En utilisant cette approche hybride de méthode syntaxique et de Deep Learning, nous pouvons diriger l'utilisation de modèles de Deep Learning pour la détection de symboles en contexte. L'utilisation du contexte nous permet de focaliser l'utilisation des modèles de détection dans des régions particulières pour un but précis comme la détection d'altérations, symboles utilisés dans la notation musicale à gauche des notes musicales pour modifier leur hauteur de note. Nous utilisons cette tâche de détection d'altérations comme un cas d'étude pour le développement de notre méthode bien que celle-ci puisse s'appliquer sur d'autres symboles musicaux.

Pour cette tâche de détection d'altérations, nous avons construit un nouveau jeu de données de détection d'altération avec l'information de la boîte englobante et de la classe de chaque altération. Ce jeu de données étant assez restreint, nous proposons d'utiliser une méthode d'augmentation de données en bougeant aléatoirement la région présentant le symbole à détecter. Cela a pour effet de changer de manière aléatoire la position du symbole à détecter dans l'image. Cette méthode d'augmentation de données, en soi très simple, nous a permis d'améliorer radicalement les résultats de détection, surtout avec le nouveau modèle de détection que nous présentons maintenant.

Notre nouveau modèle de détection est basé sur le Spatial Transformer [Jaderberg 15]. Le Spatial Transformer est composé de deux parties : un réseau de localisation et un réseau de classification. L'intention originelle de ce modèle de classification d'image est d'améliorer les résultats de classification en permettant au modèle d'avoir un mécanisme autosupervisé de localisation et d'isolation des zones intéressantes pour la tâche de classification. Pour notre tâche de détection, nous détournons cette localisation autosupervisée et utilisons une fonction d'apprentissage multitâche permettant l'apprentissage simultané de la localisation et de la classification. La localisation est apprise en deux étapes successives permettant à la première localisation de garder suffisamment d'information contextuelle pour la tâche de classification et la seconde localisation de produire une boîte englobante serrée autour du symbole à détecter.

Nous comparons notre nouvelle approche à trois différents détecteurs de l'état de l'art du Deep Learning : le Faster R-CNN, R-FCN et SSD ayant chacun des compromis différents de précision de détection et de rapidité de calcul. Les résultats détaillés sont présentés dans le tableau 2.3 et nous montrons que notre nouveau détecteur produit un mean Average Precision (mAP) de 94,81% tandis que le meilleur détecteur de l'état de l'art, le R-FCN, produit un mAP de 98,73%. Toutefois, notre détecteur est 40 fois plus rapide pouvant traiter 500 images à la seconde tandis que le R-FCN ne peut traiter que 12,5 images à la seconde. Une grande part de cette différence est dûe au fait que nous avons conçu l'architecture du détecteur basé sur le Spatial Transformer par rapport à notre tâche de détection présentant une petite taille d'image d'entrée. Une comparaison plus juste aurait été de réduire aussi la taille des architectures de l'état de l'art pour qu'elle soit plus adaptée à la taille des images d'entrées de notre tâche de détection.

Nous démontrons aussi que l'utilisation d'information contextuelle comme la tête de note associée au symbole à détecter et l'utilisation de technique d'augmentation de données nous a permis d'améliorer les résultats de notre nouveau détecteur de 30,8% de mAP. Ce travail a été publié dans l'article CHOI [Choi 18a].

Par la suite, nous avons aussi appliqué différents détecteurs de l'état de l'art du Deep Learning comme the Faster R-CNN, R-FCN et SSD sur le jeu de données de partitions musicales manuscrites MUSCIMA++ [Hajič 17] basé sur le jeu de données MUSCIMA [Fornés 12] pour démontrer la capacité de ces modèles à détecter des symboles musicaux dans des régions plus grandes et avec un ensemble plus grand de symboles musicaux. Nous démontrons aussi que les détecteurs de l'état de l'art du Deep Learning peuvent aussi être appliqués à des données manuscrites. Notre travail, qui a été publié dans l'article PACHA et al. [Pacha 18b], montre que nous achevons un mAP de plus de 80% sur un large ensemble de symboles musicaux. Nous appliquons ces détecteurs sur des régions suivant les lignes de portées des partitions car la page entière serait trop grande pour être directement traitée par les détecteurs. Le principal problème que nous avons rencontré durant ces travaux est lié au déséquilibre des différentes classes de symboles dû à la notation musicale où des symboles comme les têtes de notes sont extrêmement fréquents tandis que d'autres symboles comme le double dièse sont très rares et utilisés uniquement dans des situations très précises.

Détection de symboles non supervisée avec le Isolating-GAN

L'utilisation de technique de Deep Learning pour la détection de symboles musicaux est un processus demandant une grande quantité de données annotées. Ces annotations sont traditionnellement produites manuellement et sont longues et coûteuses à produire. Lorsqu'un petit jeu de données existe, il est possible d'augmenter artificiellement la taille du jeu de données en utilisant des techniques comme le déplacement aléatoire de la région d'intérêt et c'est ce que nous avons fait dans nos travaux précédents. Mais lorsqu'il n'existe aucunes données préalables, il ne reste que peu de stratégies pour amorcer un processus de reconnaissance automatique. Une stratégie communément utilisée dans le domaine de la reconnaissance de document est l'utilisation de données synthétiques produites automatiquement. Pour des partitions musicales, des logiciels de gravure de partitions musicales peuvent être détournés pour produire une grande quantité d'images de partition contenant des variations de musique, de notation, de police. Cependant, il est difficile d'adapter ce processus de génération à la reconnaissance d'un nouveau corpus de données, car certaines caractéristiques de partitions historiques comme l'utilisation de la notation musicale de l'époque, la gravure manuelle sur plaque de cuivre et la dégradation à cause du temps ne sont pas pris en compte par ces logiciels.

C'est pourquoi, nous avons décidé de rechercher une méthodologie plus simple pour la détection de symboles non supervisée. Au cœur de nos travaux, nous proposons une nouvelle méthode de détection de symboles musicaux non supervisée appelée Isolating-GAN utilisant uniquement des symboles isolés et basée sur une combinaison de méthodes syntaxiques et de modèle génératif de Deep Learning (GAN). Une vue d'ensemble de notre méthode est présentée dans la figure 4.1 et se définit donc en trois étapes :

1. l'identification des régions d'intérêt ;
2. l'isolation des symboles ;
3. la détection des symboles isolés.

L'identification des régions d'intérêt Notre méthode reposant sur un système génératif instable de type GAN, il est vital de réduire la taille des images à générer. Pour cela, nous utilisons la méthode syntaxique DMOS pour isoler des régions d'intérêt en utilisant des indices contextuels. Dans notre tâche de détection d'altérations, nous utilisons les têtes de notes précédemment reconnues pour décider d'une zone carrée de 4×4 interlignes située à gauche de la tête de note. Cela étant, notre méthode est conçu de façon à pouvoir être appliquée à d'autres type de symboles musicaux.

L'isolation des symboles L'idée centrale de notre méthode est de construire un domaine de représentation simplifiée utilisant uniquement des symboles isolés et servant d'intermédiaire entre la représentation complexe des partitions historiques réelles et la tâche de détection. Concrètement, ce domaine de représentation se résume à des symboles isolés de tailles aléatoires, positionnés de manière aléatoire dans une image à fond blanc. Nous proposons de réaliser ce transfert entre la représentation complexe des partitions historiques réelles et le domaine de représentation simplifiée de symboles isolés sur fond blanc en utilisant un réseau génératif antagoniste (GAN) [Goodfellow 14]. Ce modèle de GAN est composé de deux parties : une partie générative de type auto-encoder utilisant une architecture existante appelé U-Net [Ronneberger 15] capable de transformer une image en une autre et une partie discriminante permettant la mise en place d'un apprentissage antagoniste. Ce réseau est entraîné par une fonction d'apprentissage hybride comprenant la fonction d'apprentissage antagoniste originelle du GAN et une fonction d'apprentissage de reconstruction d'image appliqué à la partie générative. Cet apprentissage permet au générateur du GAN de transformer des images de partitions réelles en images contenant uniquement des symboles isolés à détecter sur un fond blanc. Ces symboles isolés sont ensuite détectés par un détecteur préentrainé que nous présentons maintenant.

La détection de symboles isolés En utilisant uniquement les symboles isolés, il est trivial d'entrainer un détecteur pour réaliser une tâche de détection synthétique dans ce domaine de représentation simplifiée de symboles isolés sur fond blanc. Une fois entrainé, nous appliquons ce détecteur sur les images générées de l'étape précédente pour détecter les positions et la classe des symboles dans les images réelles de partitions historiques.

Pour évaluer la robustesse de notre méthode, nous montrons l'application de notre méthode sur deux jeux de données de détection d'altérations dans des partitions historiques : l'un homogène contenant 70 pages de 5 partitions historiques et 2150 symboles annotés et l'autre plus large et plus hétérogène contenant 1812 pages de 58 partitions avec 818 symboles annotés. La méthode de génération étant instable, nous proposons un processus d'arrêt de l'entraînement et une méthode de sélection du meilleur entraînement parmi 10 entraînements identiques en utilisant un petit jeu de validation de 20 exemples par classe manuellement annotés. Nous obtenons un mAP de 94,8% sur le premier, petit jeu de donnés et un mAP de 82,5% sur le second jeu de donnés plus hétérogène. Pour démontrer l'utilité de la partie générative de notre méthode nous comparons les résultats de notre méthode en faisant un test d'ablation de la partie générative et montrons que l'utilisation de la partie générative permet d'améliorer de 30% la précision sur le premier petit jeu de données et de 66% la précision sur le deuxième jeu de données plus hétérogène.

Conclusion

Dans ce travail de recherche sur la détection de symboles musicaux dans le cadre de partitions imprimées historiques denses, complexes et bruitées, nous avons tout d'abord exploré la tâche de détection de symboles musicaux en utilisant des modèles de détecteur de l'état de l'art du Deep Learning et avons aussi proposé une nouvelle architecture de détecteur adaptée pour une tâche de détection de symboles musicaux dans une zone restreinte. Après cette exploration de modèles de détections supervisés, nous avons fait le constat du manque crucial de données annotées permettant l'utilisation direct de modèles supervisés de Deep Learning. C'est pourquoi, nous proposons notre nouvelle méthode appelée Isolating-GAN de détection de symboles musicaux non supervisée utilisant uniquement des symboles isolés. Cette méthode utilise une combinaison d'une méthode syntaxique et de Deep Learning pour graduellement simplifier la tâche de détection en trois étapes. Tout d'abord, la méthode syntaxique est chargée d'identifier des régions d'intérêt en utilisant des éléments contextuels reconnus apriori, puis notre modèle génératif est en charge de simplifier la représentation graphique en isolant les symboles à détecter et en effaçant le fond bruité. Enfin, un détecteur préentrainé en utilisant des symboles isolés préexistants est utilisé pour détecter les symboles présents dans les images générées à l'étape précédente.

Nous appliquons notre méthode sur deux jeux de données, l'un petit et homogène et l'autre plus large et hétérogène et obtenons des résultats de détection de 94,8% de mAP et de 82,5% de mAP respectivement. Cela nous a aussi permis de traiter de manière automatique le jeux de données large et hétérogène en détectant 38908 symboles dans 1774 pages de partitions historiques imprimées. L'ensemble de nos expérimentations a nécessité 830 entrainements différents, ce à quoi nous estimons avoir pris environ 2 mois et demi de temps d'entrainement. Dans de futurs travaux, nous projetons d'appliquer notre méthode à de nouveaux types de symboles musicaux, tout en stabilisant le processus d'apprentissage en utilisant des avancées de l'état de l'art sur les réseaux antagonistes génératifs comme le Wasserstein GAN [Arjovsky 17b]. L'application à d'autres types de documents comme des partitions manuscrites est aussi envisagée.

Nous pensons que notre nouvelle méthode de détection de symboles non supervisée est un premier pas dans la construction de méthode de détection entièrement autonome, pouvant s'adapter à de nouveaux corpus sans effort d'annotations manuelles.

INTRODUCTION

A document can be defined as a human artifact which purpose is to preserve and share knowledge. Music score documents is a specialized type of documents transcribing the act of playing music and aims to capture pitch, rhythm, intensity and many other aspects of music. The specific case of music scores as a document is very interesting by its unending goal of trying to capture in writing something as ephemeral and emotional as music. Music notation, like any documents, has both the purpose of preserving a music piece as good as possible, capturing as many details as possible, while being easy to read in real time by musicians during a performance. Moreover, different musical cultures and traditions expressed different needs for music notations, as well as changing needs through time because of the evolution of music. As a result, we can observe widely different music notations, going from early music notation to the modern music notation in Europe, while Korea, China and India, to only name a few, developed different music notations to transcribe traditional music. In this work, we focus on the modern music notation, which is now the most commonly used music notation throughout the world. However, this music notation has a long history in Europe, starting from the beginning of the 18th century. The printing of the music scores using the modern notations could not be done anymore using movable music types because of the structural complexity of the notation, where the relative placement of symbols could be much less linear than the early music notations. Thus, a new kind of imprinting technique was developed where the music score would first be manually engraved on copper plates, and then be used in printing press to replicate the music score page on sheets of papers. While composers would write music manually, a large amount of music scores were printed and are now precious historical documents that are studied by many musicians and musicologists. Unfortunately, it is often hard to access these documents because the hard copy is often restricted and the digitalization of the music scores only gives images for musicians and musicologists to study.

The modern music notation consists of a collection of staves which are five vertically stacked horizontal lines. These five lines serves as a 2 dimensional axis to describe both the flow of time, by reading from left to right and the pitch of musical notes, by the vertical placement of note heads. Musical notes are laid out on the staves from left to right at different vertical position and transcribes a music piece or melody.

Optical Music Recognition (OMR) is the automatic process of turning images of music scores into a machine-readable format, allowing easier use and study of those historical printed scores. This subfield of document recognition research has been studied since the 1960s, starting from the work of Pruslin [Pruslin 66] and is still actively researched. Well-known OMR literature review like Fornés et al. [Fornés 14] and Rebelo et al. [Rebelo 12] segments the OMR workflow as a succession tasks. This workflow often starts with various low-level graphical processing that are often common to a lot of document recognition workflow like binarization and noise removal. Then staff lines are recognized and optionally removed, symbols are detected However, the task of recognizing a historical printed score is very challenging because of the combination of a very complex music notation with high symbol density and degradations due to the printing method and age of the document. While classical OMR systems always struggled to recognize historical printed music score documents, the OMR research has seen a renewed interest recently due to novel computer vision architecture based on Deep Learning models. The early stages of OMR such as music symbol detections being inherently a computer vision task, we demonstrate in this work how Deep Learning models can produce highly accurate and precise music symbol detections. One of the downside of classical fully supervised Deep Learning models and methods is the needs for large amount of annotated data to train the models. This need for large amount of annotated data is problematic, especially in OMR, where no such datasets exists for historical printed music scores outside of datasets produced during the course of this work. Therefore, the heart of this work is to propose a new unsupervised method called Isolating-GAN able to produce highly accurate symbol detection without using any manual annotations for the training of Deep Learning models and only use a simple set of isolated music symbols.

To summarize, in chapter 1, we expose the difficulties of correctly recognizing a historical printed music scores because of the printing techniques and degradations of old documents. Then we present an overview of OMR and its different processing steps, with a focus on the step of music symbol detection. Finally, we present a review of relevant Deep Learning methods that we will use throughout this work, starting from fully/weakly supervised detectors to generative models like GANs.

Then, we proceed in chapter 2 onto using fully supervised Deep Learning music symbol detector to accurately detect music symbol for an OMR pipeline. We present results on two different datasets and tasks: first, a small and constrained accidental detection task on a newly constituted dataset using 4 different detectors to give a sense of speed versus accuracy of the task of detecting music symbols. Secondly, we apply fully supervised state-of-the-art detectors on the handwritten music score dataset called MUSCIMA++ and present results on a large set of music symbol classes.

In chapter 3, we discuss the various possibility of using Deep Learning models without manual annotations. We first discuss the possibilities of using synthetic data as commonly done in the document recognition field. In our case, we chose to use a combination of synthetic data based only on isolated symbols and generative methods such as GANs so that we can train a music symbol detector in an unsupervised fashion while not relying on a complex synthetic generation method such as music score typesetting software. The generative capability of our method also implies that our method will be able to adapt seamlessly to new corpus of documents.

This leads to chapter 4 where we expose the design of our new Isolating-GAN method for unsupervised music symbol detection. We propose a three steps method with the general idea of gradually reducing the complexity of the detection task. In the first step, we reduce the spatial search space using the DMOS syntactical method. In the second step, we simplify the graphical representation using an image-to-image translation generative model. This simplification consists of isolating symbols to detect in a white background and this target representation is synthesized using only a preexisting dataset of isolated music symbols and injected using a hybrid training objective. This simplification consists of isolating all symbols to detect while erasing other symbols and noises to produce a white background. This target representation is synthesized using only a preexisting dataset of isolated music symbols and injected using a hybrid training objective. The third and final step is to detect the previously isolated symbols by using a detector pretrained using a synthetic isolated symbol detection dataset.

Finally, we proceed to show an extensive set of experiments in chapter 5. Because we have the very challenging task of training an unstable generative model such as a GAN with no manually annotated ground truth, we started out our experiments with a simplified task and dataset to tune the common hyperparameters of our method. Then, we gradually raise the difficulty of the task by detecting different types of symbols, as well as rarefy the amount of symbols to detect. Then we validate our evaluation methodology when no ground truth data is present, which is the real use case of our method. Finally, we evaluate our method using a much larger, more heterogeneous dataset, where symbols to detect are even less frequent. We demonstrate the effectiveness of our generative method by comparing the results with an ablated non-generative version of our method and shows a massive improvement in precision.

In the end of this manuscript, we discuss in chapter 6 the future work and possible improvements of our method. We discuss possible strategies to stabilize and improve the training of our method. We also believe that we can improve our method by taking larger images as input, a more diverse set of symbols to detect and could even be applied on handwritten music scores or other structured documents with significant segmentation problems such as electrical circuit design documents. Finally, we believe that the work presented in this manuscript could be the first step towards entirely autonomous structured document recognition systems, where documents could be recognized in stages, with each stage processed entirely in an unsupervised manner using only isolated symbols and gradually building a syntactical structure of the document with each stage reusing and building on the symbols recognized on previous stages.

CHAPTER 1

STATE-OF-THE-ART

1.1 Introduction

Optical Music Recognition as a subfield of Document and Image Recognition domain is a very active area of research and has lately seen a significant increase in interest. This renewed interest is mainly caused by new advances in machine learning techniques like Deep Learning model that has been shown to produce state-of-the-art results on OMR tasks like staff lines recognition and removal, music symbol detection or music notes recognition in historical printed or handwritten music scores.

OMR is a very broad subject and can be studied under many angles depending on the end user needs. However, we can distinguish two broad kinds of tasks in OMR. We have OMR end user focused tasks like MIDI transcription (music note extraction) of a music score or even the full transcription of a music score into a machine format such as the MusicXML format that can be used in music typesetting software. Such end user tasks are often very difficult to accomplish in a single step and that is why OMR researchers has often decomposed the OMR process into smaller subtasks, while not useful to OMR end users, are useful to construct an entire modular OMR system. Music score preprocessing, staff lines recognition and removal, music symbol detection and music notation reconstruction are commonly identified subtasks in OMR systems.

In this work, we focus on improving the subtask of music symbol detection because it is often the pivotal and most important step of an OMR system. Music symbol are the primary building blocks of a music score and the ability of correctly parsing and identifying music symbols almost guaranty a correct interpretation of a music score. Moreover, by being inherently a computer vision tasks, music symbol detection has the most to gain from recent advances in computer vision tasks like supervised object detection. However, one of the consequences of using fully supervised object detection model is the need for large amount of manually produced ground truth, which is very impractical in the OMR domain since only a few music symbol detection datasets currently exists for training such model and none exists for our actual task of detecting music symbols in historical printed music scores. That is why we propose to build on recent advances on generative Deep Learning models like the GAN architecture to reduce our dependency on manually produced ground truth.

We now present the state-of-the-art in both the OMR and Deep Learning domains in regard to music symbol detection. In section 1.2, we present enough of the music notation and history to understand the problem domain of music symbol detection in historical music scores. We then move on to present in section 1.3 an overview of the different OMR tasks and state-of-the-art techniques. We also present in more details the DMOS syntactical method in section 1.4 often used throughout our work to ease the application of Deep Learning models to OMR tasks. Finally, we review the Deep Learning existing work we build on for supervised object detection in section 1.5 and semantic segmentation in section 1.5.2, as well as generative models like GANs in section 1.6 for reducing the need for manual annotations.

1.2 Music Notation

Music notation is the written visual transcription of music as an audio signal and is used to capture, save and share how a piece of music should be performed. In Europe, music notation was mainly developed by religious monks to transcribe sacred voiced music, starting with the early music notation. During 17th century, the music notation was transformed to better represent instrumental music like the piano as shown in fig. 1.2b and produced what is now known as the modern music notation or Common Western Music Notation (CWMN).

The modern notation is structured around a staff which is a collection of five long horizontal lines going across the whole page of a music score. A staff is horizontally divided into bars of regular duration specified by the time signature placed at beginning of the staff after the clef. These horizontal lines are stacked vertically with a regular spacing and forms a two-dimensional system where the horizontal axis represents the flow of time and the vertical axis represents the pitch of a musical note. A musical note is defined by its duration and pitch and is usually composed of a note head (oval blob full or empty) and an optional stem, flag, dot or accidental.

For determining the duration of a musical note, first a metronome mark $♩ = 120$ is used to specify the absolute duration of a quarter note ♩ in beat per minute, then different symbols are used to specify the relative duration of a note either by multiplying the duration by two (half note 𝅗𝅥), four (whole note 𝅝) and more or dividing the duration by two (eight note ♪), four (sixteenth note 𝅘𝅥𝅯) and more. A dot can be placed at the right of the note head ♩. to lengthen the duration by one-half and additional dots can be added to further lengthen the note. Multiple notes with a duration inferior to the quarter note can be merged to form beamed notes ♫ to compact and facilitate the reading. The same principle is used to represent silences with whole rest 𝄻, half rest 𝄼, quarter rest 𝄽, eight rest 𝄾, sixteenth rest 𝄿 and more.

The pitch of a note is determined by first specifying a clef like the G clef 𝄞 which sets the absolute pitch of one of the line of the staff (second line from the bottom has the pitch G for the G clef). Modern music notation uses five staff lines to represent the diatonic scale where the alternating lines and spaces between two consecutive lines (also called an interline) represent shifts from one pitch to the next. The difference in pitch between two consecutive lines is called a step while the difference in pitch between a line and the touching interline is called a half-step. The vertical position of the note head is then used to specify the pitch which will depends on the clef that was specified at the beginning of the staff. The pitch of a note can further be modified by using accidental symbols located at the left of the note head to either increase the pitch by half a step using a sharp ♯, decrease the pitch by half a step using a flat ♭ or cancel a previous accidental by using a natural ♮. However, this change is only temporary and applied only for the current bar. For a durable change, a key signature is used by adding one or multiple sharps or flats at the right of the clef on the beginning of the staff.

Articulation marks can be placed above or under a note to specify how the musician should perform the note. For example, a Fermata sign 𝄐 can be used to indicate that a note can be played for a longer duration than the written duration. A list of all the commonly used symbols are shown in fig. 1.1.

In the modern music notation, a voice is a sequence of following notes that should be played together. Polyphonic instruments have the particularity of being able to play multiple voices at the same time which will be transcribed by stacking the notes vertically to represent the fact that these notes should be played simultaneously. Polyphonic instruments can have one or multiple staves with different keys that tries to better cover the range of pitches possibly played by the instrument, and a voice can sometime go from one staff to another.

1. `https://en.wikipedia.org/w/index.php?title=List_of_musical_symbols`

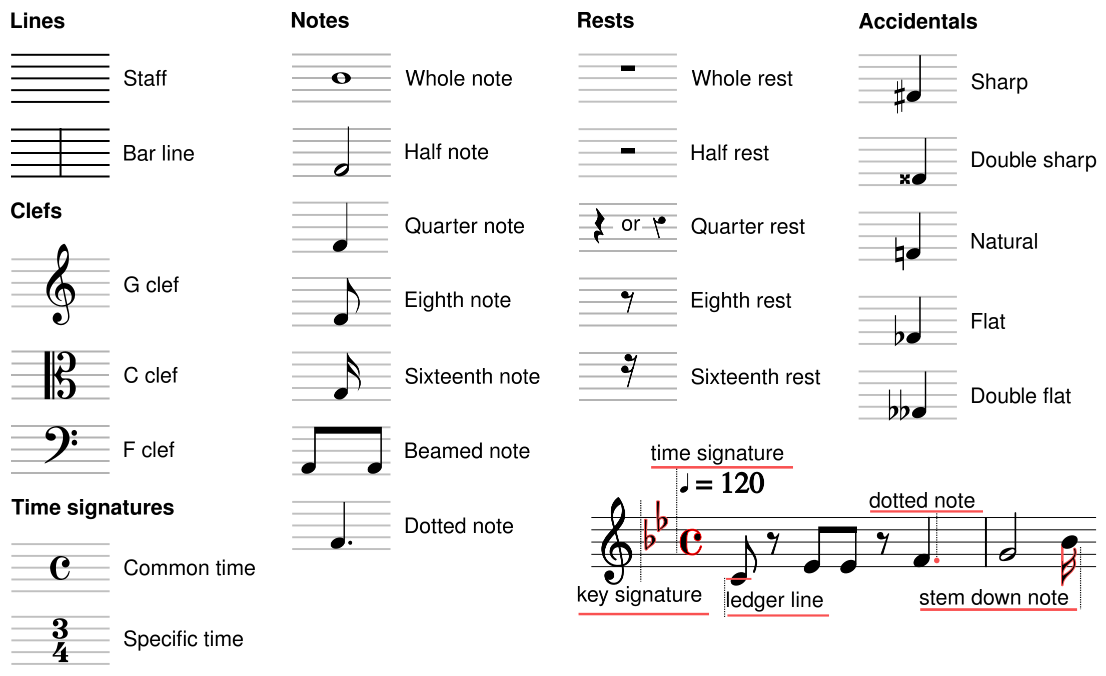


Figure 1.1 – Common elements of music notation. Images provided by Wikipedia [1].

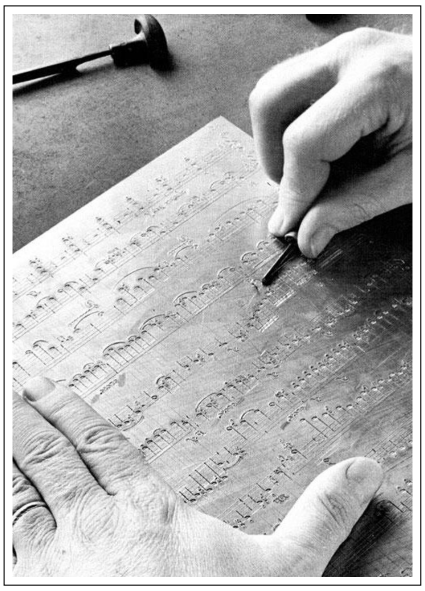
While most of the music notation uses simple shapes to graphically represent how a musical piece should be played, it is the complex spatial organization of the symbols and their relative relationship that is difficult to process by an OMR system. Moreover, in this work, we concentrate on the recognition of complex printed historical music scores like very dense piano music scores as shown in fig. 1.2b or orchestral music scores that push the boundaries of what the music notation can represent. The complexity of these scores stems from the use of polyphonic instruments like the piano able to simultaneously play multiple voices at the same time and, for orchestral scores, the transcription of multiple instruments all synchronized in time. This density and notation complexity coupled with the variability resulting from the manual engraving process we now present makes the recognition of such scores yet an unresolved challenge.

1.2.1 Music Score Engraving

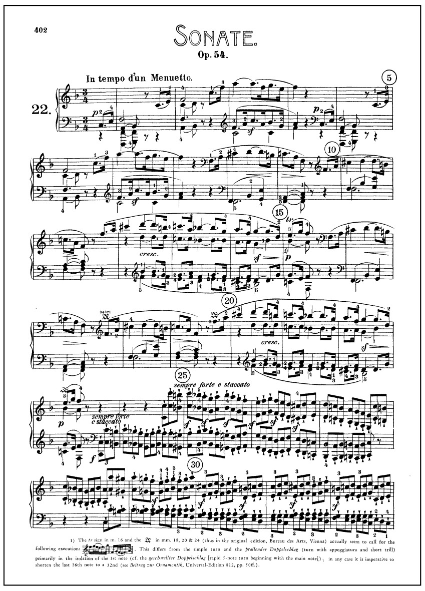
Music engraving started in a manuscript form by monks replicating sacred liturgical music. With the invention of the printing press, music types were then developed to produce music scores at a faster pace. In order to replicate a music score, the music had to be assembled in reverse with many small music types representing all the music symbols, notes and lyrics like a puzzle. Scores printed with music types have the characteristics to produce a very clean and readable music score although it was still very time consuming and error prone to produce and the assembling of tiny music types produced wavy staff lines. With the coming of modern classical music in the 18th century, new techniques like plate engraving emerged. Plates engraving is a technique where copper plates is first engraved using hand tools as shown in fig. 1.2a. While this step is very time-consuming, often taking multiple hours for a single page of music scores, music scores could then be printed at a much faster pace for a much longer time. Manual copper plate engraving of music scores is generally seen as a form of art, where the engraver needs very high level of technical skills to be able to replicate in reverse a music score with maximum readability. However, this manual process eventually leads to variation in how the music score is transcribe where the engraver compromise between music notation rules, readability and typesetting complexity. These variations often lead to an increased difficulty for OMR system to correctly parse music scores, either in the music symbol detection process with incorrect overlapping and broken symbols or in the music notation reconstruction process with incorrect or uncommon music typesetting. At the same time, since copper plate engraving is a technique used during the 18th, 19th, 20th century, music scores imprinted using copper plate are often old, noisy and damaged with adds to the difficulties of correctly parsing the score.

Nowadays, music scores can be produced at very high quality using music typesetting software like Sibelius, Finale or MuseScore. Even music composers now uses such software to compose music, often using MIDI keyboard to directly send musical notes to the typesetting software. The typesetting software will take care of automatically arranging music symbols in order to produce a highly readable music score with accurate spacing. While all the different typesetting software each have their own file format to represent a music score, two standards have emerged: the Music Encoding Initiative (MEI) format and the MusicXML format. Moreover, open sourced typesetting software like MuseScore can also be modified and used as a music score dataset generator, potentially being able to produce an infinite amount of synthetic data for any of the OMR tasks. However, the generation of synthetic music scores only solves one part of the problem and does not take into account artifacts presents in real historical music scores originating from the engraving process or document degradations.

2. Courtesy of `https://musicprintinghistory.org/about-music-engraving/`



(a) Copper plate engraving of music scores [2].



(b) First page of the Piano Sonata No.22, Op.54 composed by Ludwig van Beethoven and edited by Universal Edition in 1918–21.

Figure 1.2 – Examples of music scores production and imprint using copper plate engraving as typically done in the 18th, 19th and 20th century.

1.3 Optical Music Recognition

Optical Music Recognition is the task of converting a music score document into a machine-readable format. This task is a well-known problem in the Document Image Analysis and Recognition domain and has been researched since the 1960s with Pruslin [Pruslin 66]. We base this state-of-the-art review of OMR on Fornés et al. [Fornés 14] and Rebelo et al. [Rebelo 12] which are the two most well-known and complete literature review of the OMR domain. In this section we review the field of OMR with a focus on symbol detection and classification. We present in section 1.3.1 an overview of the different components of an OMR workflow. We also present an overview of existing OMR systems in section 1.3.2 and existing OMR datasets in section 1.3.3 that can be used for the tasks of classification and detection of music symbols.

1.3.1 Optical Music Recognition Stages

In this work, we concentrate on the recognition of historical printed scores using the Common Western Music Notation. As explained before, these scores present multiple difficulties for OMR systems at different stages of the recognition process. First of all, modern music notation is a very complex two-dimensional system with music notation rules changing through time and editors. Secondly, music scores imprinted with manually engraved copper plate sometimes presents typesetting artifacts due to its manual process which leads to segmentation problems. Finally, historical music scores presents noises and defects caused by the age of the document.

Because of the complexity of the task, OMR studies by Fornés et al. [Fornés 14] or Rebelo et al. [Rebelo 12], illustrated in fig. 1.3, typically present the OMR workflow as multiple consecutive stages: image preprocessing, staff detection with possible removal, music symbol segmentation/classification and finally music notation reconstruction. However, many works reorganize, merge or remove some of these stages as we will see in section 1.3.1.5.

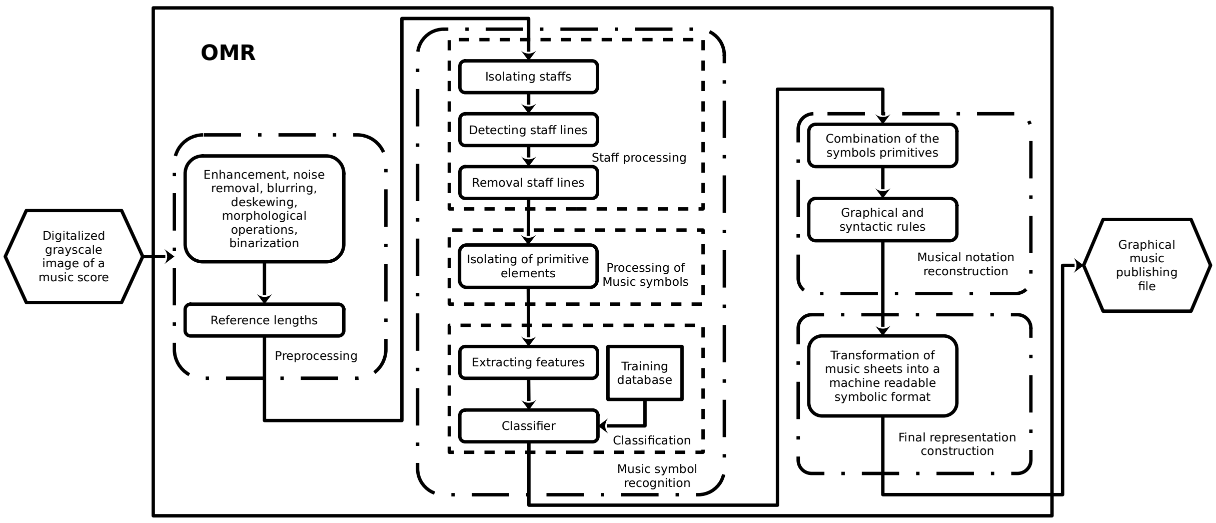

OMR
Digitalized grayscale image of a music score
Enhancement, noise removal, blurring, deskewing, morphological operations, binarization
Reference lengths
Preprocessing
Isolating staffs
Detecting staff lines
Removal staff lines
Staff processing
Isolating of primitive elements
Processing of Music symbols
Extracting features
Training database
Classifier
Classification
Music symbol recognition
Combination of the symbols primitives
Graphical and syntactic rules
Musical notation reconstruction
Transformation of music sheets into a machine readable symbolic format
Final representation construction
Graphical music publishing file

Figure 1.3 – Typical architecture of an OMR processing system as shown by Rebelo et al. [Rebelo 12].

1.3.1.1 Preprocessing

Existing work in OMR tends to use common document preprocessing operations. Binarization is often applied to isolate connected components from the background which can later be used for the music symbol detection and classification step explained in section 1.3.1.3. However, OMR pipelines often works on multiple representation of the image, gray-scale or binarized image, depending on the need of following stages. For example, the DMOS system [Coüasnon 01] uses a gray-scale image for segment detection and a binarized image for connected components extraction. Recent work like [vdWel 17] omits this step completely to avoid introducing additional defects that can be caused by a binarization process and directly feed a gray-scale image into an end-to-end trainable music note recognition model.

Historical music scores often presents defects either caused by time degradation or a poor scanning process. Time degradation induced noise can sometimes be removed using noise removal filters but it is often hard to consistently apply such filters manually. On the other hand, poor scanning process often present the same kind of wave-like distortion that can be corrected using skew-correcting techniques. This skew-correction step is often very important in OMR because of the presence of long staff lines going through the whole width of the page, the recognition of which is critical to correctly deduce the pitch of musical notes.

1.3.1.2 Staff Lines Detection and Removal

Staff lines detection is a very important step of an OMR system for a multiple of reasons. First of all, the vertical stacking of staff lines is used to specify the pitch of music notes while their horizontal direction specify the flow of time. From a graphical point of view, the vertical distance between two staff lines, also called the interline, is used as a normalized unit distance that parameterize the size of most other graphical objects in the score. Therefore, the correct recognition of this size will have a significant impact on later OMR steps like music symbol recognition presented in section 1.3.1.3 and music notation reconstruction presented in section 1.3.1.4. Another characteristic of staff lines is the fact that it is a very long horizontal graphical line that join most of the music symbols in a staff. That is why it is quite common that staff lines detection algorithms are coupled with staff line removal algorithm.

The OMR literature as explained by Fornés et al. [Fornés 14] groups staff lines recognition methods into five categories:

Projections, Histograms, Run lengths These methods [Pruslin 66; Fujinaga 04; Kato 92] are based on the distinctive characteristics of staff lines being thin horizontal lines and are very fast to compute. However, the simplicity of the methods often does not account for skewed lines or overlapping symbols.

Candidate Assemblage and Contour Tracking These methods [Prearu 70; Roach 88; Fornés 06] first try to isolate candidates to be considered for being a staff lines, for example by skeletonizing the image and isolating thin lines, then joins the candidates to construct whole staff lines. These methods are known to have a good balanced between speed and accuracy.

Graph Path Search These methods [Carter 92; Cardoso 09] tries to construct a graph either based on lines primitives or the image itself. Then, the graph is analyzed to construct the best interpretation possible of staff lines. These methods are known to produce the most accurate results by being able to cope with skewed or gaps in staff lines together with overlapping symbols but are also slow because of the expensive graph analysis step. In this work, we use a graph-based Kalman filtering method [dAndecy 94] to detect and remove staves from the original gray-scale image. It's ability to process broken or curved lines accurately is very important given that we want to process dense, noisy and damaged historical printed music scores.

Convolutional Neural Network Recent work like [Calvo-Zaragoza 17] has used Convolutional Neural Networks (CNNs) to do pixel-wise classification to locate staff lines. These methods produce very good results and are shown to outperform other state-of-the-art classic methods for staff lines detection and removal. Another advantage of this method is that no preprocessing like binarization is required for the method to work. However, these methods require pixel-wise ground truth information which are very costly to produce and expensive hardware to train the CNN model.

No Staff Removal Once staff lines are correctly removed from a binary image of a music score, later OMR segmentation and recognition steps are much easier to perform. However, this removal also frequently damages other symbols in the music score because of the real difficulty of the task, with a lot of overlapping symbols and gaps and distortions in the staff lines. Damages done during the staff line removal process often impacts negatively successive OMR recognition steps, which is why some OMR systems like [Pugin 06] do not remove staff lines before further processing of the image. This allows for faster processing of the image or a more accurate detection of music symbols since no degradations can be introduced by the staff removal steps. However, following steps like music symbol detection which we now presents have to be adapted to work with the presence of staff lines.

1.3.1.3 Music Symbol Detection

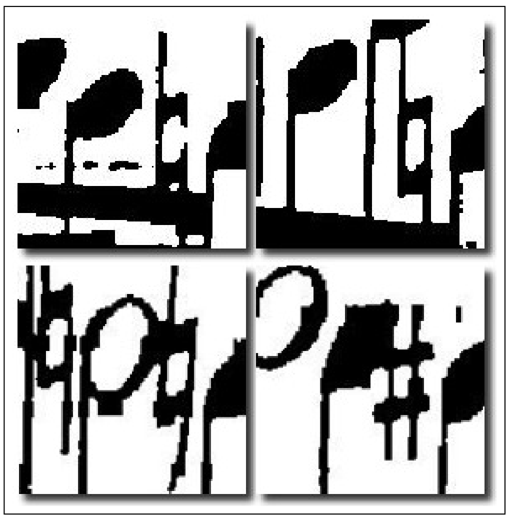
Music scores are constructed using a lot of relatively simple shapes like lines and blobs in a complex bi-dimensional structure. This fact has pushed OMR systems to use simple extraction algorithm like graphical primitive detection or connected components, and then use complex ad hoc rules to merge or over-segment primitives [Fornés 14]. The classification of music symbols can be done using a variety of techniques like simple filters, template matching or classifiers like HMM, neural network, K-NN and SVM as presented by Rebelo et al. [Rebelo 09].

More recently, convolutional-based neural network detectors [Pacha 18b] that merge the segmentation and classification steps have been applied to a variety of dataset like the newly annotated handwritten dataset of modern music, the MUSCIMA++ dataset [Hajič 17] or on mensural music scores by Pacha et al. [Pacha 18a]. Fully convolutional neural networks have also been used by Hajic et al. [Hajic 18] and Tuggener et al. [Tuggener 18b] which allows for pixel wise segmentation of music symbols.

1.3.1.4 Music Notation Reconstruction

Finally, the last step of the recognition process is to reconstruct the music notation and validate the structure produced. Rebelo et al. [Rebelo 12] shows that because of the strong structure and graphical rules of music notation, it makes sense to model this organization using a grammar. Most of these methods are used at the end of the OMR pipeline, to either check the validity and construct the music structure from previously detected primitives. For example, the work of Modayur et al. [Modayur 93] uses a constraint based system driven by music notation rules to check the validity of the previously produced recognition. Graph grammars can also be used to model music notation as shown by the work of Fahmy et al. [Fahmy 93] and Baumann [Baumann 95] where graph grammar rules are used like a declarative knowledge base that is used to construct a symbolic representation of the music scores using previously detected primitives. However, these types of grammars never reconsider the segmentation done at a previous stage of the OMR processing pipeline. Throughout our work, we use the DMOS method described in section 1.4 to model and construct the musical structure of a music score. The interesting characteristics of this method compared to the other grammar method is its ability to drive both the recognition of graphical primitives and construct the musical structure of a document. This dual characteristics allow us to use this method both at the beginning of the OMR recognition stage for some automatic data generation process or at the end of the OMR process to validate the produced musical structure.

We believe that low-level symbol segmentation problems caused by the density, noise or preprocessing of a music score (see fig. 1.4) should not be part of the grammar for an OMR system, as it is too complex to be modeled explicitly. We wish to devise a method that delegates the segmentation task to a statistical model, in our case a Convolutional Neural Network (CNN) designed to do both segmentation and classification. Furthermore, our method was developed with the goal that it could be applied to any kind of structured document.


(a) Touching Symbols

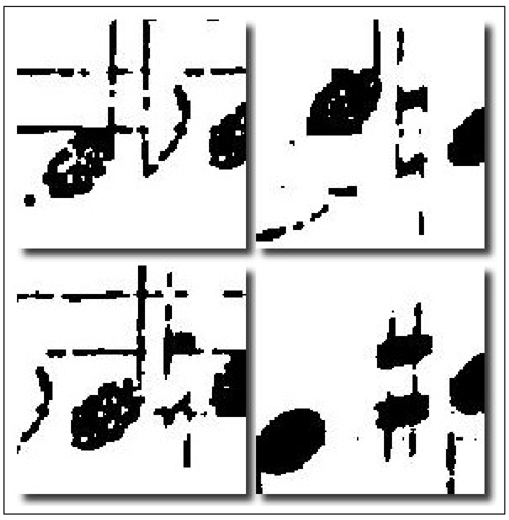

(b) Broken Symbols

Figure 1.4 – Hard segmentation problems for accidental detection task.

1.3.1.5 Full-Pipeline OMR

The disadvantages of segmenting the OMR process into small modular tasks is the introduction of errors, especially in the earlier stages, that can prevent later stages to work correctly. Modern approaches often choose more integrated approaches where a trainable model will try to learn a task end-to-end. These kinds of approaches have the advantages of using trainable models able to adapt to many distortion and noises without modifying the training method.

The work of Hajic et al. [Hajic 18] uses a U-Net based model that we also present in section 1.5.2 to train a model that will be able to directly detect music notes in handwritten music scores as well as predict the duration and pitch of the musical notes. This approach is enabled by the use of the MUSCIMA++ dataset that we present in section 1.3.3, which contains graphical annotation of music notes such as shapes and bounding boxes as well as music structural annotations like music note duration and pitch. Pacha et al. [Pacha 18a] also take this end-to-end approach using a region-based convolutional neural network applied to handwritten music scores with mensural notation. The authors of [vdWel 17] uses a convolutional sequence-to-sequence model to directly predict the sequence of notes in synthetically generated music scores augmented with artificial degradation techniques. One of the current limitation of end-to-end approaches is the difficulty to correctly represent complex musical structure at the end of the recognition process. For now, most of the end-to-end method like the recent work from Rıos-Vila et al. [Rıos-Vila 20] can only recognize monophonic scores or mono-voiced scores that often concentrates on only recognizing the musical notes useful for music playback of the scores but not sufficient for a complete re-imprinting task.

While there are obvious advantages of end-to-end approaches, like the ability of the same model to adapt to different kind of data and tasks, the main disadvantages is the need for large amount of ground truth often annotated manually. This is the main reason we propose to study weakly-supervised detection method in section 1.6.2 and in a more general manner generative method in section 1.6. Moreover, in the second part of this work chapter 4, we propose a novel approach to symbol detection, using a method able to detect music symbols without using any detection ground truth. Although our approach is not end-to-end, we believe that creating an OMR process using less manually annotated data should be the core focus of modern approaches.

We now show present briefly exiting OMR system that tries to integrate all this step into a single software package so that the task of automatically processing music scores could be done by users other than OMR experts.

1.3.2 Software

While the end goal of OMR varies with users, a few fully integrated OMR software exists, with the goal of transcribing an image of a music score into its equivalent in MEI or MusicXML format in order to be opened, read and adapted using music typesetting software. At the time of writing of this manuscript, here is the list of known still developed OMR software:

— Audiveris: `https://audiveris.github.io/audiveris/`

— SmartScore: `https://www.musitek.com/`

— SharpEye: `http://www.visiv.co.uk/`

— PhotoScore: `https://www.neuratron.com/photoscore.htm`

— ScanScore: `https://scan-score.com/en/`

— capella-scan: `https://www.capella-software.com/us/index.cfm/products/capella-scan/info-capella-scan/`

— OMeR: `https://www.myriad-online.com/en/products/omer.htm`

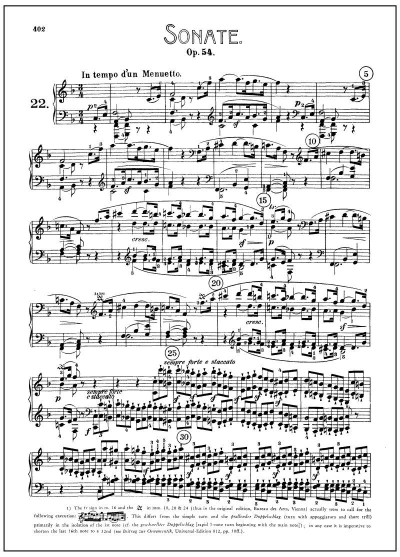
It is, however, difficult to compare the results of such software because of the lack of a common agreement on how the OMR task should be evaluated and the fact that most of these software programs are black box commercial software that are expensive to acquire and difficult to automate. Nonetheless, ongoing work like [Byrd 15] tries to define the music notation complexity of a music score in order to propose a set of performance metrics based on a standard dataset of music scores of varying difficulties. In the scope of our work, we only study the specific subtask of music symbol detection, for which there are commonly used metrics such as the mean Average Precision metric generally used in object detection [Everingham 10].

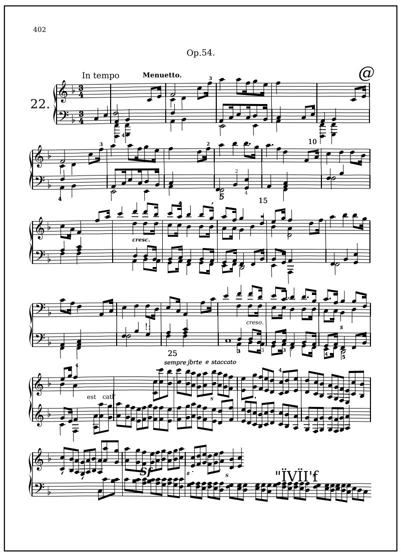
Although we only quickly tested with the open-source Audiveris OMR software as shown in fig. 1.5, we agree with the general statement of Fornés et al. [Fornés 14] that OMR software are able to correctly recognize synthetically cleanly generated music scores but the recognition of handwritten or historical printed music scores or even synthetic music scores containing scanning or camera artifacts are still a too difficult task for these systems.

Now that we have presented the available OMR systems, we now present available OMR datasets used by OMR researchers to further the state-of-the-art in OMR.

1.3.3 Datasets

Multiple datasets exist for different OMR tasks. First of all, the most well-known handwritten music score dataset is the MUSCIMA dataset made by Fornés et al. [Fornés 12]. While the original goal of the dataset was to evaluate the staff lines recognition


(a) Historical printed score


(b) Audiveris pdf output

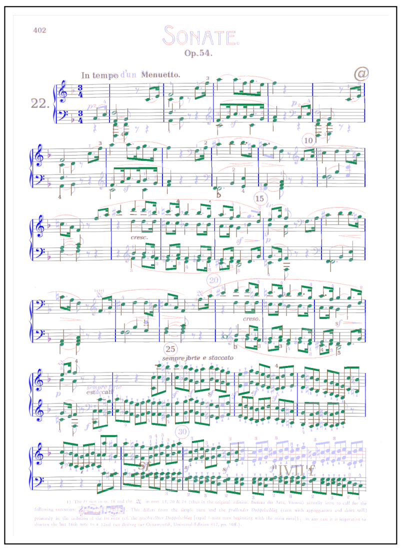

(c) Audiveris visualization

Figure 1.5 – Example of Audiveris OMR software version 5.1.1 recognizing the first page of the Piano Sonata No.22, Op.54 shown in fig. 1.2b. Staff lines and measure bars recognition errors leads to the impossibility to correctly reconstruct the rhythms information of the score. Lots of symbols are also missing such as accidental symbols. The manual correction of audiveris recognition should be a very long process akin to manually typesetting the original music score.

and removal tasks, it was later augmented by Hajič et al. [Hajič 17] to also include symbol level annotations in order to evaluate symbol detection tasks and also some musical relationship annotations for a subset of music notation reconstruction task. The dataset is fairly large with 1,000 sheets written but the dataset is very homogeneous since it is the same 20 music pages manually transcribed 50 times by different musician writers and only present monophonic music scores. The dataset is fully annotated with every musical primitives and symbols described at the pixel level. This dataset is widely recognized as one of the most complete dataset for handwritten scores using the modern music notation and can be used for a wide variety of tasks, from staff line recognition and removal to symbol detection and classification and also music notation reconstruction.

Other datasets exist, going from symbol classification dataset such as the Rebelo dataset [Rebelo 09] with 15,000 printed symbols or the Fornes dataset [Fornés 08] with 4,100 handwritten symbols to synthetic dataset such as the DeepScore dataset [Tuggener 18a] with 300,000 annotated images for symbol detection.

Unfortunately, no datasets currently exists for historical printed music score using the modern music notation, apart from isolated symbol datasets. We believe the lack of such dataset is the results of the very complex nature of such music scores and this is why we propose to work on such type of music scores. Moreover, we also propose a new dataset in the context of this work, a symbol detection dataset[3] containing bounding boxes and label annotations for three different music symbol classes in historical printed music scores that we reuse throughout this work.

1.3.4 Conclusion

We have now presented an overview of the state-of-the-art in OMR, starting by the presentation of the modern music notation that constitute a music score (section 1.2) and the different problematics that we will face in this work. We have shown that the recognition of historical printed music scores is a very difficult challenge because of the complexity of its music notation, manual engraving methods and time degradations which complicates the segmentation task and music notation reconstruction task. From this assessment, we have presented the state-of-the-art for the different OMR steps (section 1.3) from preprocessing, staff line recognition, symbol detection to music notation reconstruction showing that the crucial step of music symbol detection is

3. `https://www-intuidoc.irisa.fr/en/choi_accidentals/`

still one of the most challenging step of the OMR pipeline especially in historical printed music scores. Finally, after enumerating existing fully integrated OMR system (section 1.3.2) and existing OMR datasets (section 1.3.3), we have shown the evident lack of OMR research on historical printed music scores. To complete our OMR state-of-the-art review, we now review the DMOS syntactical method used throughout this work to drive our OMR system and construct our hybrid approach to detecting symbols in historical printed music scores.

1.4 The DMOS Syntactical Method

The DMOS syntactical method was introduced by Coüasnon [Coüasnon 01], and is a general off-line method for recognizing structured documents. DMOS uses attributed two-dimensional grammars to define the symbolic and graphical representation of documents, producing constituent parse trees. The contextual information produced by the grammar can also be used to restrict the search space of our detector, as explained in section 2.2.1. Musical scores was the first application of the system by specifying a grammar to describe the modern musical notation. The grammar could describe fully polyphonic scores with different voices on a single staff, allowing syntactical checking of the number of beats in a bar and the vertical alignment coherence of synchronized notes.

The hierarchical graphical structure produced, for example a simple music note as illustrated in fig. 1.6, is described by a set of rules that can search through the use of backtracking and check the coherence of different note elements. An implementation of this rule using the EPF syntax is shown in listing 1.1. This ability to pinpoint inconsistencies can be used to efficiently produce semi-annotated data by reducing the amount of manual verification. Although the grammar is tailored to deal with complex polyphonic orchestral scores, segmentation had to be addressed using dedicated rules, which are difficult to produce and maintain. This detection of music symbol is the task we are proposing to resolve using Convolutional Neural Network-based detectors that we now present in our next section.

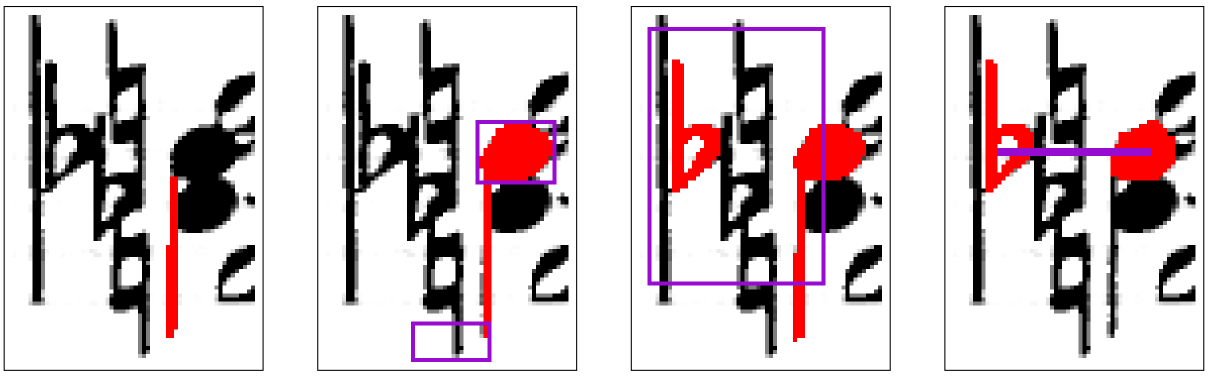


Figure 1.6 – Grammar workflow. Recognized elements are red. Violet squares are zones where recognizable elements are searched for. The construction of a musical note starts (a) find a potential stem, (b) two possible locations (top-right and bottom-left) are searched for a note head, (c) a potential accidental is searched at the left of the note head, (d) an alignment check is done between the note head and accidental.

1.5 Supervised Detection

The detection of symbols is an essential step of an OMR workflow. In the document recognition field in general and especially in OMR, we need a very precise localization of music symbols in order to correctly guess the relative relationship between the symbols and derive the structure of the music notation.

We therefore propose a summary of state-of-the-art fully supervised techniques for symbol localization. Since the localization of an object or symbol in an image can be done at different level of precision, we first present a coarser approach of detecting object using bounding-boxes based method in section 1.5.1. We then present in section 1.5.2 a finer but more heavy approach of detecting objects at the pixel level using semantic segmentation techniques.

1.5.1 Supervised Object Detection

We now review state-of-the-art bounding-boxes based object detectors we use or build on in the course of this work.

Spatial Transformer Network In the work of Jaderberg et al. [Jaderberg 15], the authors present a new Spatial Transformer Network (STN) architecture in order to

Listing 1.1 – EPF grammar example graphically describing a musical note

```
noteStemDownWithAccidental Note ::=
    stem StemSeg &&
    AT(aboveRight StemSeg) && notehead NoteheadCC &&
    AT(left in|NoteheadCC) && accidental NoteheadCC AccidentalCC &&
    constructNote StemSeg NoteheadCC AccidentalCC Note.

stem StemSeg ::=
    TERM_CMP isStem StemSeg.

notehead NoteheadCC ::=
    TERM_CMP isNotehead NoteheadCC.

accidental NoteheadCC AccidentalCC ::=
    TERM_CMP isAccidental AccidentalCC &&
    isAligned NoteheadCC AccidentalCC.
```

enhance the classification accuracy of a simple Convolutional Neural Network (CNN) model. The ST network as shown in fig. 1.7 is composed of two stacked CNN: a localization network and a classification network. The localization network has the task to output a 2D affine transformation for a given input image effectively isolating the symbol to recognize from its background. A Spatial Transformer Layer (STL) applies this transformation to the input image, that will be then fed to the classification network. Because the detection mechanism is implicit through the use of a 2D affine transformation, this architecture has a lot of flexibility to define a detection task, either unsupervised or in a supervised manner as we proposed in section 2.2.2.

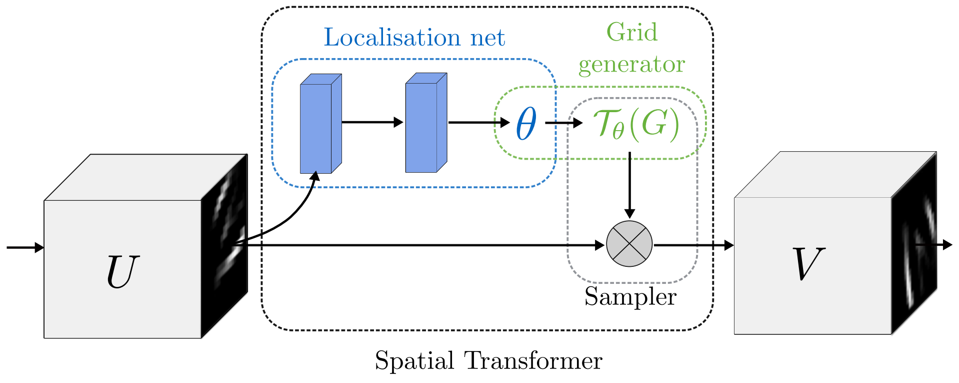


Figure 1.7 – Spatial Transformer architecture as proposed by Jaderberg et al. [Jaderberg 15].

Faster R-CNN The Faster R-CNN [Ren 15] is one of the pioneer object detection architectures in Deep Learning and is now widely used in very diverse tasks. The detection process happens in two steps. First, in a Region Proposal Network (RPN) stage, a feature extractor (VGG-16 or resnet 101) is used to process input images. Then, at some intermediate layer of the feature extractor, anchor boxes are used in a sliding window manner to predict class agnostic box proposals. This RPN is trained using a multi-loss function taking into account both localization and objectness score produced by the RPN. Secondly, some of these proposals (usually 300) are cropped from the feature layer used to predict them, and the rest of the feature extractor is processed. Unlike the RPN stage, the second stage outputs class-specific bounding boxes refinement for each of the proposals. Finally, a similar multitask loss is used to optimize the second stage detection. The Faster R-CNN has presented state-of-the-art results when first introduced in 2015 on both the Pascal VOC 2007 dataset with 73.2% of mean Average Precision (mAP) and the Pascal VOC 2012 dataset with 70.4% mAP.

R-FCN The R-FCN detector proposed by Dai et al. [Dai 16] is an adaptation of the Faster R-CNN architecture designed for even faster detection. While the Faster R-CNN avoids a lot of computation by sharing a single network for both RPN and full detection stages, it still needs to process each region proposal until the end of the feature extractor. That is why the R-FCN architecture proposes to extract region proposals only at the last layer of the feature extractor and therefore reduces the amount of computation for each proposal. They also propose a position-sensitive cropping mechanism using position-sensitive score maps in order to retain the localization information for each proposed region. R-FCN is much faster than the Faster R-CNN, while maintaining comparable accuracy.

Single Shot Detector The third object detector we propose to use is the Single Shot Detector (SSD) [Liu 16]. Unlike the Faster R-CNN and R-FCN that use two stage predictions, the SSD architecture predicts directly class and bounding boxes of objects from a single pass of the feature extractor as shown in fig. 1.8. This model is typically significantly faster than two stage detectors like Faster R-CNN and R-FCN.

Overview Since CNN bounding boxes-based detectors have shown to produce state-of-the-art results for object detection tasks like the Pascal Visual Object Challenge

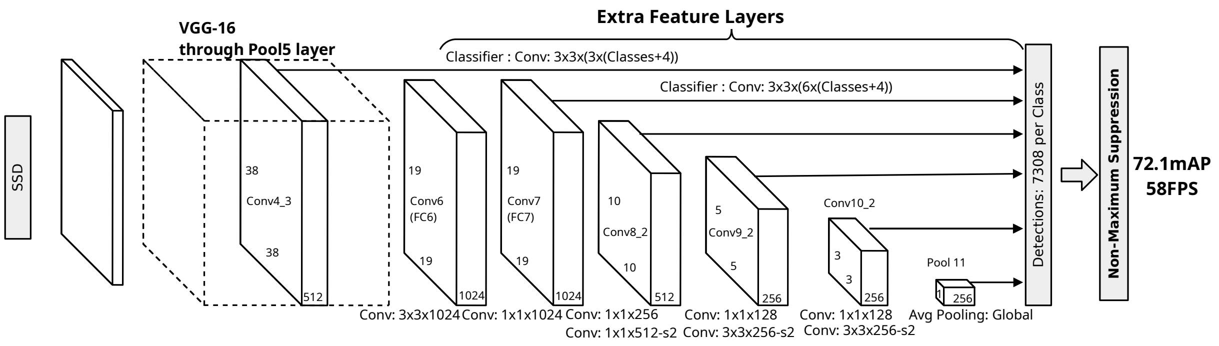


Figure 1.8 – Single Shot Detector as presented by Liu et al. [Liu 16].

(VOC) [Everingham 10], we propose to use such detectors for localizing music symbols in dense, noisy historical printed music scores. We believe such detectors can be trained to accurately music symbols despite the noise and density characteristics of historical music scores. Different detector models have a different speed and accuracy trade off where the Faster R-CNN and R-FCN have a very high accuracy and slower processing speed and the SSD model a faster processing speed but a generally lower detection accuracy.

When more fine-grained detection is needed for the localization of objects, Semantic Segmentation models can be used to accurately localize an object with its contour using a pixel-wise classification approach. We now review state-of-the-art Supervised Semantic Segmentation models.

1.5.2 Supervised Semantic Segmentation

Semantic segmentation is the task of classifying every pixel of an image. After clustering neighboring pixels of the same classes, we can deduce a very precise location of objects or symbols in the image. Most of the state-of-the-art approach builds on the Fully Convolutional Network (FCN) architecture.

Fully Convolutional Network Fully Convolutional Network introduced by Long et al. [Long 15] proposes to adapt existing convolutional network like AlexNet [Krizhevsky 17], VGG net [Simonyan 15] and GoogLeNet [Szegedy 15] to output a dense prediction map of the size of the input image in order to classify every pixel of the input image. Their final proposal is the Fully Convolutional Network (FCN) architecture that contains skip connections between the convolutional layer in order to combine the feature hierarchy of different layers and refines the spatial precision of the output.

U-Net The U-Net model proposed by Ronneberger et al. [Ronneberger 15] is designed for task of segmenting biomedical images by labeling each pixel of an image. Starting from a Fully Convolutional Network (FCN), the U-Net adds a symmetric expanding path which is able to up-sample each of the feature maps produced by the corresponding level of the FCN. This additional path forming a U shape is shown in fig. 1.9. The contracting path (FCN) is therefore able to capture contextual information and build the relevant feature extractor for the task while the symmetric expanding path is able to produce a precise localization by directly reusing the FCN extracted features. This work also emphasizes the use of data augmentation in order to efficiently train a U-Net model with very few annotated examples.

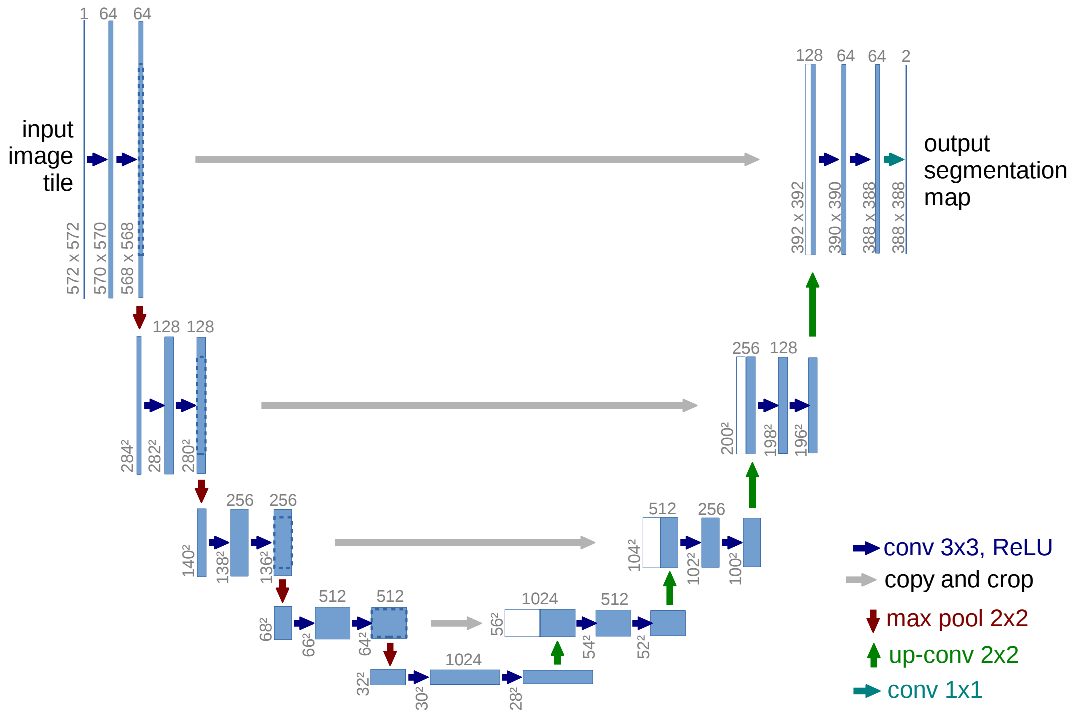


Figure 1.9 – U-Net architecture as shown by Ronneberger et al. [Ronneberger 15].

The U-Net architecture has also been successfully applied to document processing tasks [Ares Oliveira 18] like page extraction, baseline extraction, layout analysis, illustration and photograph extraction. This work emphasizes the use of a generic CNN-based architecture for different document processing tasks followed by tasks specific post-processing like thresholding, morphological operations, connected components analysis and shape vectorization. Although pixel level annotations are hard to produce, the authors notes that the generic U-Net model requires a small amount of manual annotations to be successfully trained thanks to the pretraining of the U-Net architecture on the ImageNet dataset [Deng 09]. Building on this successful application of the U-Net architecture on document processing tasks, we also propose to reuse this architecture in our unsupervised detection method presented in section 4.5.2.

1.5.3 Conclusion

The literature on supervised object detection methods or supervised semantic segmentation methods have shown the capacity of these methods to be applied on a wide range of applications, going from natural object recognition, from street signs, cars and many types of objects in natural scenes but also to the segmentation of medical images or segmentation of historical documents. We believe such detection models can be applied for music symbol recognition, significantly boosting the recognition capacity of any OMR pipeline. That is why we study such application of object detectors for music symbol recognition in chapter 2. However, one significant downside of supervised methods is their need for annotated data, often with a costly and slow manual process. Although many possible strategies can be used to reduce the need for manually annotated data, the combination of supervised and generative models have shown to be a very promising path to reduce the use of manual annotations. We review such generative methods in our next section.

1.6 Generative Methods for Reducing Manual Annotations

The recent advances in supervised computer vision models like CNN have also driven a significant increase in the demand of annotated data. In our case, the application of supervised music symbol detectors first needs for music symbol detection datasets to exist. To find a solution to this lack of annotated data, domains like weakly-supervised/unsupervised learning or cross-domain adaptation learning has researched ways to reduce the need for annotated data while training supervised models. One of the recent advances of these domains is the combination of generative models like Generative Adversarial Networks (GAN) together with fully supervised models that can successfully reduce the amount of annotations needed and even improve the accuracy of the supervised model. Therefore, we first review the principle of the original GAN model in section 1.6.1 and then shows the application of the GAN model for cross-domain adaption learning in section 1.6.2 and for data generation tasks for weakly-supervised learning in section 1.6.3. Finally, we review the main caveat of these approaches which is the instability of training GAN models in section 1.6.4.

1.6.1 Generative Adversarial Network

Generative Adversarial Network (GAN) introduced by Goodfellow et al. [Goodfellow 14] and illustrated in fig. 1.10 is a type of generative model which can be trained using an adversarial training objective.

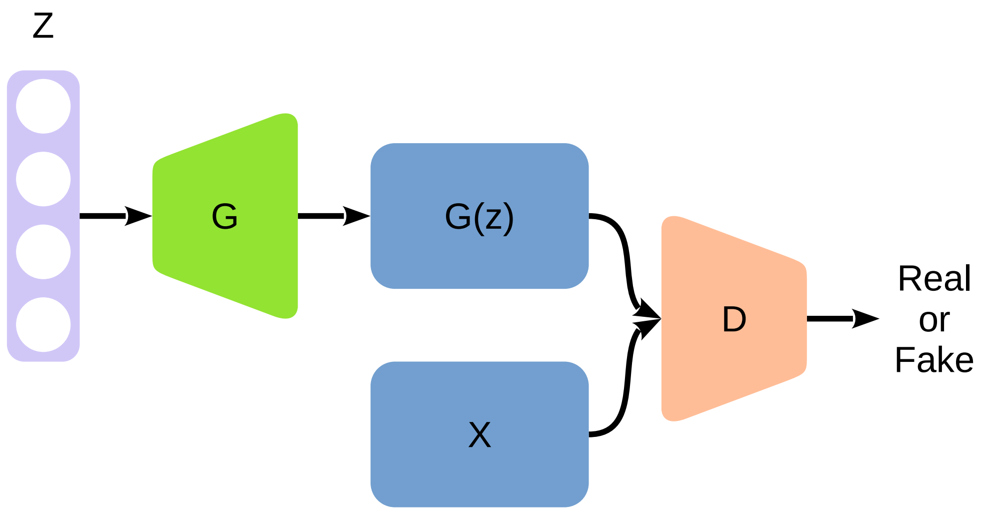
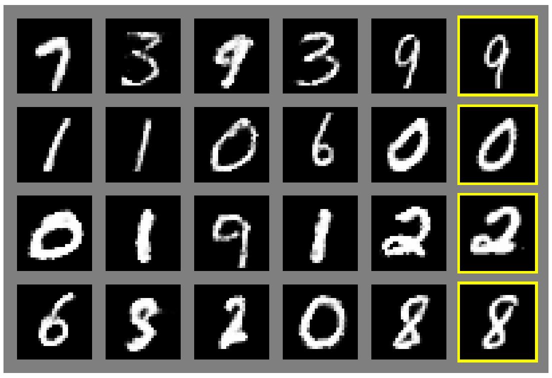
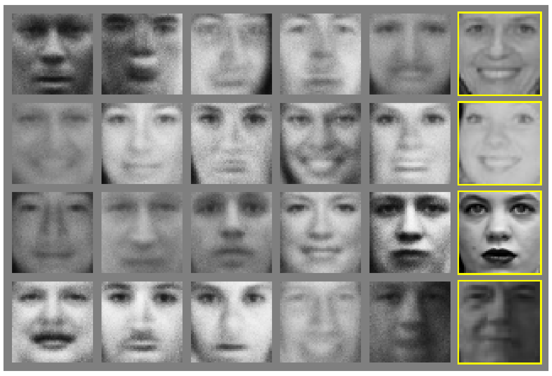


Figure 1.10 – Schema of a Generative Adversarial Network (GAN) as presented by Pan et al. [Pan 19]. A GAN is composed of a Generator (G) and a Discriminator (D). The Generator transforms a random noise vector z into $G(z)$ to fool the Discriminator. The Discriminator has to differentiate between real X samples and generated samples $G(z)$.

This adversarial training puts in competition a generator network and a discriminator network where the generator needs to fool the discriminator and the discriminator needs to identify data coming from the generator. This mechanism can then be used to generate images that are indistinguishable, in principle, from real images. An example of generated digit images, faces and natural images is shown in fig. 1.11. The input of the generator is called a latent representation of the generated output as it parameterize the generation process. In the original proposal of the GAN model, the values for the input vector is randomized in order to explore the latent representation space. Although the original work managed to replicate small images of digits, faces and animals, this type of adversarial training is known to be very unstable and further modification of the loss objectives and architecture has been proposed to use GAN for more complex tasks.


(a)


(b)

(c)

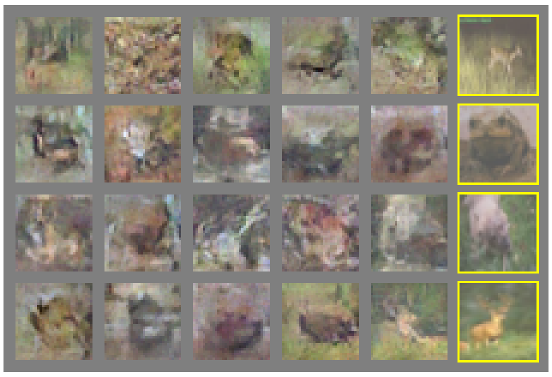

(d)

Figure 1.11 – (a) Generated digit, (b) faces and (c)-(d) natural images produced by the original GAN architecture as shown by Goodfellow et al. [Goodfellow 14].

1.6.2 Cross-Domain Adaptation using GAN

One of the most interesting application of the GAN model is the use of its latent representation as a bridge between two domains of representation. This allows to use the generator as a translator between two domains, which in turn allows a transfer of any other prediction task trained on one representation to be applied on another representation.

Cycle-GAN The work by Zhu et al. [Zhu 17] proposes to apply a modified version of the original GAN model for the task of Image-to-Image translation. Traditionally, the task of Image-to-Image translation needed datasets of aligned pairs of images where each image of the source domain had a known corresponding image in the target domain. The use of a GAN model allowed here to realize this task without having aligned pairs of images. However, instead of having only one generator and discriminator network, the Cycle-GAN is using two generators and discriminators network where each generator can produce images of the source and target domain and each discriminator can identify images of the source and target domain. Additionally, two cycle consistency losses are used to minimize the difference between an image and its translation to and from the target domain. This training mechanism allowed to train a generative model able to transform an image from a source domain to an image resembling images of the target domain while retaining properties that are common to both the source and target domain.

Weakly-Supervised Detection Although large scale detection dataset already exists for popular application domain like object detection in photographic images, the high cost of creating such large datasets for less popular domain is driving the research for weakly-supervised detection models. Formally, weakly-supervised object detection is a task where only images and image-level class annotations are provided for training.

The work of Inoue et al. [Inoue 18] proposes an improved weakly-supervised detection model by using cross-domain adaption of a pretrained detection model. In this task, we consider both a source domain and a target domain, where the source domain, for example photographic images, is fully annotated with classes and bounding boxes annotation at the instance-level. However, the target domain, for example cartoons or drawings, only contains image-level annotations and its class set is a subset of the source domain class set. The proposed model is a fully supervised detector model, pretrained on the source domain, which is then successively fine-tuned using two synthetically generated samples. The first synthetic samples are images of the source domain which are transferred to the target domain using a Cycle-GAN model. Therefore, the detector is fine-tuned using synthetically generated images and annotations from the source domain. In a second time, the detector is fine-tuned by using a pseudo-labeling of the target domain images where synthetic bounding boxes are generated by using a combination of the detector output and image-level annotations. The results using the Pascal VOC20017 and VOC2012 source dataset (with instance level annotations) and the Clipart1k target dataset (using only image level annotations) shows that the proposed method of Domain Transfer + Pseudo-Labeling (DT+PL) is outperforming other weakly supervised detection approaches and unsupervised domain adaptation approaches with 46% of mAP against 27.4% of mAP at best for other approaches.

In summary, weakly supervised detection tasks implies that image-level class annotations of the target domain are used to train a weakly supervised model and the use of a GAN model to cross the boundaries between images of different domains does not require any annotations.

Unsupervised Word Recognition This cross-domain translation can even be used in a total unsupervised fashion using synthetic data as shown by the work of Kang et al. [Kang 20]. The authors show how a classic word recognition architecture such as a Recurrent Convolutional Neural Network (RCNN) can be trained using a hybrid loss. The word recognition task is trained using synthetic words where the ground truth is automatically generated while another adversarial task is learned by using an additional discriminator so that the convolutional network of the RCNN produces an identical embedding for synthetic word images and real word images. This approach has the very interesting advantage of never needing any manually annotated ground truth for training a word recognition model that can be applied on real images. However, in order to train such model, a synthetic data generation method has to be manually crafted such that the convolutional network will be able to learn a common embedding process for synthetic data and real data. On the IAM dataset, this method produces a Character Error Rate (CER) of 14.05% and is compared to the upper bound of 6.88% CER when training directly with real annotations and the lower bound of 26.44% when training with synthetic annotations only. In general, after experimenting with 5 different datasets, the method always shows a improvement over only using synthetic dataset for word recognition.

1.6.3 Data Generation with Localization using GAN

A very practical use of the GAN architecture is to use them jointly with another fully supervised method to reduce the need for manually produced annotations. Such approaches, also called weakly supervised approaches, have been successfully applied to task like object instance segmentation with the work of Remez et al. [Remez 18]. This work uses as input information produced by a pretrained Faster R-CNN detection model in order to train a GAN generator network to learn to segment a single object previously detected. An adversarial training strategy is proposed based on a cut and paste approach (named Cut&Paste). The generator produces a mask of a previously detected object by using features extracted by the Faster R-CNN. This mask is then used to extract the object from its original image and pasted into a background image. Then, the discriminator has to differentiate between the original images and images produced using the generator. In this instance, the use of a GAN architecture train with an adversarial loss allowed to train a segmentation model without using any manually annotated segmentation data. Instead, the model uses an indirect source of information which is the output of a previously trained detection model and managed to refine the localization of detected objects without using additional ground truth information. After experimenting on the COCO dataset, this Cut&Paste method shows to outperform a similar weakly supervised instance segmentation approach Simple Does It by 2% on the mAP metric and approach the upper bound of 70% of mAP produced by a fully supervised model version of the model trained using the ground truth mask of the COCO dataset. This method shows an original way of using a GAN-based method to adapt image of object from one scene to another, but it still needs a fully-supervised detector as prerequisite of the method.

The work of Yang et al. [Yang 17] is another examples of data generation using a GAN model. This work proposes a modified version of a GAN model called the Layered Recursive GAN (LR-GAN) that is able to iteratively generate images of individual object and finally compose them using a modified version of the Spatial Transformer Layer previously presented at section 1.5.1. Although, the goal of this model is to generate realistic images, the use of an explicit localization information could be interesting for other detection tasks. Unfortunately, the training stability of such model is very difficult to maintain. This is why we now review the study of GAN models under the lens of stability.

1.6.4 Generative Adversarial Network Instability

The main difficulty of using a GAN model identified by its original author Goodfellow et al. [Goodfellow 14] stems from having no explicit representation of generator output and having to synchronize the training between the discriminator and generator. When badly synchronized, the training can lead to a collapse of the generator to a single output. This instability has been since studied in works like [Arjovsky 17a] which identifies that one of the probable source of instability of GAN models stems from the characteristics of the loss function used to train a GAN model. Indeed, the use of a binary cross-entropy loss can lead to a gradient vanishing issue, for example in situation where the discriminator accept or reject all images coming from the generator. The authors further expand its idea by Arjovsky et al. [Arjovsky 17b] where they propose to improve the stability of the GAN training framework by mainly changing the loss function by using the Wasserstein distance (also called Earth Mover distance). This alternative loss function have the property to continue to provide a useful gradient (as opposed to a null gradient for the original binary cross-entropy loss) even when the discriminator is trained optimally. This reduces the impact of the training balance between the discriminator and the generator.

Unfortunately, we did not have the time to integrate this work into our own GAN model architecture but we believe that using a better loss function such as the Wasserstein distance could improve the stability of our approach. We discuss in section 6.2 how we could take advantage of such improvements in future works.

1.6.5 Conclusion

To conclude, we have shown the working principle behind the original GAN model (section 1.6.1) together with their possible application for reducing the need for manual annotations by combining them with supervised models. By doing cross-domain adaption learning (section 1.6.2), GAN models are able to bridge different representation space, allowing the application of a fully supervised model trained on one domain to another domain without retraining the supervised model. Weakly-supervised models (section 1.6.3) aims to improve the accuracy and reduce the need for manual annotation of existing supervised models by feeding artificial samples generated by a GAN model. Finally, we have presented the main difficulties of these methods which is the instability of GAN training methodology (section 1.6.4).

1.7 Conclusion

Following the hybrid nature of our work, we now have presented the state-of-the-art for both OMR and Deep Learning based object detection methods.

We have shown that OMR is still an active area of research with a great diversity of tasks and applications such as re-imprinting, music extraction and playback or enhanced search of music scores. For the particular case of historical printed music scores using the modern music notation, we have shown that the recognition of such music scores is a very difficult task, combining a very complex music notation, a manual typesetting process introducing defaults and unpredictable variations and finally lots of noises and degradations caused by old documents or a bad scanning process. This kind of music scores have often been neglected and considered an easy recognition task by the OMR community, often because of the apparent similarity with modern software produced music scores. This is shown by the absence of existing historical printed music scores suitable for machine learning tasks like symbol detection prior to this work.

Since very few works exists on the recognition of historical printed music scores, we believe that focusing on the problem of music symbol detection is the most effective way to bootstrap a modular OMR recognition system. For this symbol detection task, we have presented state-of-the-art Deep Learning detection method that can be seamlessly applied to music scores. Since annotated dataset are needed for training such supervised model and that none such datasets exists for historical printed music scores, we explore the combination of detection method with generative methods such as GAN able to reduce the need for manually annotated dataset.

Starting with chapter 2, we present the first step of this work: applying supervised Deep Learning-based detector to music symbols in noisy and dense historical printed music scores.

CHAPTER 2

SUPERVISED MUSIC SYMBOL DETECTION

2.1 Introduction

In the domain of Optical Music Recognition, the music symbol detection step is one of the central steps of the full OMR pipeline, following the preprocessing and staff lines recognition steps and preceding the music notation steps. While not every OMR approaches perform an explicit music symbol detection step and favor more end-to-end approaches (section 1.3.1.5), we believe that a self-contained music symbol detection step is essential for a better modular and flexible OMR system.

With the rise of highly precise and efficient supervised Deep Learning detector, large improvements can be made to existing OMR system by simply integrating existing Deep Learning detector for music symbol detection. In this chapter, we show our novel use of Deep Learning-based detector for music symbol detection, first in a constrained accidental symbol detection task on historical printed music score in section 2.2. In this first approach, we integrate this detection task into an existing syntactical method describing the music notation, making this a hybrid approach with the DMOS syntactical method driving our Deep Learning detector with a limited amount of annotated data. We propose a comparative study between multiple state-of-the-art detection model as well as a new architecture based on the Spatial Transformer exploring an original approach of the detection task where the localization could be trained implicitly with no explicit ground truth. With this new detector, we show the effectiveness of data augmentation techniques as well as the use of contextual information. In the next section 2.3, we also apply Deep Learning detectors on a more challenging detection task of detecting a wide variety of music symbols in modern handwritten scores and show the effectiveness of Deep Learning-based detectors on the task of handwritten music symbol detection.

2.2 Hybrid Approach to Accidental Detection

The recognition of mid-18th to mid-20th century dense and damaged piano scores presents unique segmentation problems of touching and broken music symbols as shown in fig. 1.4 due to their imprinting techniques and time degradation. Segmentation and classification of music symbols is an early task of the pipeline and should be highly precise and reliable because a segmentation or classification error could propagate and ruin the latest stages like music notation reconstruction. The segmentation of music symbols is the most challenging task because of the lack of previous work/datasets to test and compare approaches on historical printed music scores. With the introduction of recent deep learning architectures for object detection, we can now apply an end-to-end approach to segmentation and classification of music symbols. However, deep learning architectures generally need a lot of annotated training data, which we do not have for old printed scores.

In this chapter, we consider the problem of detecting a single accidental symbol with the a priori knowledge of the position of the associated note head (see fig. 2.1). We bootstrap this task by introducing a new detection dataset of accidentals and address this detection challenge using a new Spatial Transformer-based detector that is both small and fast, and compare this with three state-of-the-art object detectors. Our objective is to train these detectors with a limited number of annotated samples and design methods general enough to be easily adapted for other symbols.

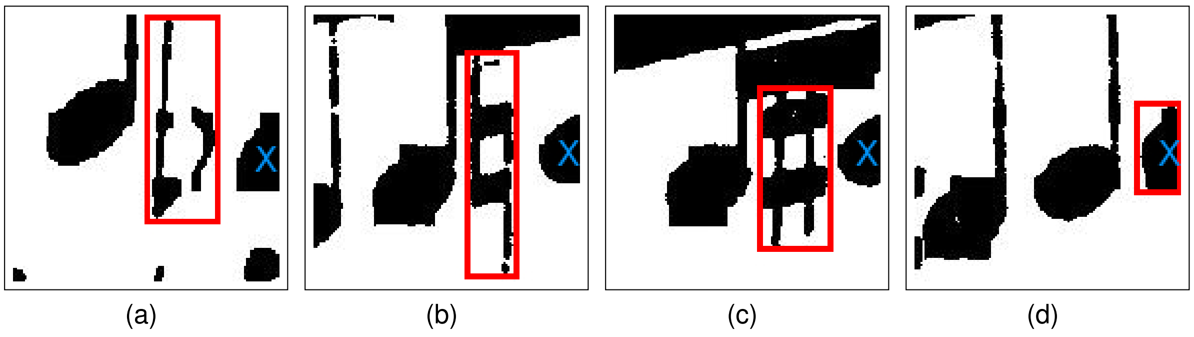


Figure 2.1 – Task definition: detector should predict the position (red box) and class of an accidental (flat, natural, sharp or no accidental (rejection)) using raw image pixels and the apriori knowledge of the centroid of the note head (blue cross).

In summary, we propose to resolve this detection task by semi-automatically creating a new detection dataset in section 2.2.1 using a grammatical description of the music notation that contextualize our detection. We then propose to resolve this detection task using either a contextual bootstrapping approach with a Spatial Transformer-based detector in section 2.2.2 or a multiple RoI object detector approach in section 2.2.3. Finally, we validate and compare the results of our two detection approaches in section 2.2.5.

2.2.1 Dataset Construction

In order to evaluate the performance of Deep Learning detectors for our music symbol detection task, we first propose a new small dataset for single accidental detection in dense and noisy piano scores that is freely available online for the research community [1].

We used three different scores edited in the 19th century from the composers Friedrich Kuhlau, Felix Mendelssohn and Richard Wagner. The constitution of this dataset was semi-automated by using the DMOS syntactical system shown in section 1.4 to analyze the layout of the score. Using the recognized structure, we were able to extract potential locations of accidental by looking to the left of note heads. Connected components in the location were then classified using a simple CNN based classifier trained on isolated printed music symbols. This automated process only allowed to correctly detect symbols that were already isolated and constituted of a single connected components. Therefore, we manually verified every potential accidental symbol and manually annotated incorrect detections, obtaining 2,955 examples containing three accidental classes and a reject class (for when a note has no accidental), see table 2.1. In fig. 2.1, we show how we position our detection window, with the target note head on the right side, using four times the size of the space between staff lines as estimated by DMOS (the *interline* distance) as the window side length.

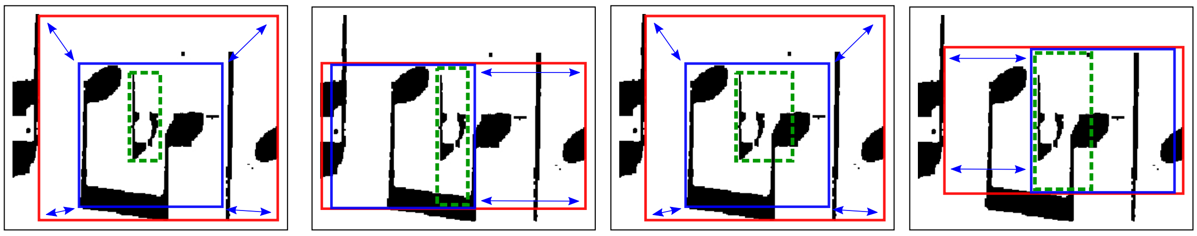
Table 2.1 – Dataset produced by the DMOS pipeline driving a simple connected component based segmentation, a simple music symbol classifier and a manual check.

Label	**Quantity**
No accidental (Reject)	968
Natural	968
Sharp	777
Flat	242
Total	**2,955**

1. `https://www-intuidoc.irisa.fr/en/choi_accidentals/`

Given the omnipresence of the target note head, we require that the detector localize the note head if there is no accidental associated with it. This allows us to define rejection using a concrete symbol detection goal, rather than trying to detect missing accidentals in background noise.

Bootstrapping Strategies Having a small initial number of training examples, we use a translation-based data augmentation method, randomly moving the window framing the accidental (or note head for rejection). We propose four different variations shown in fig. 2.2. The figure shows how we set boundaries on possible positions for randomly located windows. The *unconstrained* model in fig. 2.2a requires the accidental, or note head for rejection, to always be entirely in the sampling window (blue square in the figure). We avoid introducing the vertical displacements not present in the original data using the *novertical* model in fig. 2.2b, where the window must be vertically centered around the centroid of the note head. We still allow a small range of 10 pixels of vertical variation. The note head being a strong visual cue linked to the accidental, we propose a third generation model called *notehead* (see fig. 2.2c) where at least half of the current note head should always be inside the sampling window. Finally, we combined the *novertical* and *notehead* model constraints (the *novertical_notehead* model in fig. 2.2d). For each of these bootstrapping strategies, we augment the dataset in different quantities: 25k, 50k, 100k, 200k, 400k.



(a) *unconstrained* (b) *novertical* (c) *notehead* (d) *novertical_notehead*

Figure 2.2 – Four randomized sample bootstrapping techniques. The red square shows possible areas where the blue square, which is the sampling window, can be positioned. The green zone is always inside the blue sampling zone.

We also use this data augmentation opportunity to balance our dataset and over-sample less frequent classes like the flat and natural. Our previous use of the centroid position of the current note head can now be used to distinguish between multiple accidentals, and help the network to pick the right accidental to localize anywhere in the image.

2.2.2 Spatial Transformer-based Detector

We propose a new accidental detector based on the Spatial Transformer (ST) Network previously presented in section 1.5.1. In this architecture, a localization network learns to localize a region in the input image. The regions is then cropped from input image and fed to a classification network for a classification task. The original appeal of this architecture was the fact that an explicit localization was learned implicitly, driven by the classification task. We first believed that we could take advantage of this original architecture to avoid using bounding box annotations and train a detector only using classification label information. However, it turned out that there is a discrepancy between the classification network, needing a larger than the symbol region to produce a high quality classification and the localization task defined by a very tight box around the symbol to detect. Nonetheless, we manage to transform this architecture into a fully supervised detector using both classification labels and bounding box annotations to produce high quality music symbol detections. We present a schema of this new architecture we propose in fig. 2.3. We use only four parameters for affine transformations instead of the original six by Jaderberg et al. [Jaderberg 15] by zeroing out the two shearing parameters to produce axis-aligned bounding boxes as is common for detection tasks. The main modification of the ST architecture is the forwarding of the affine transformation produced by the initial localization network to the new multi-task network that produces both classification and a localization correction for a symbol. This localization correction is added to the initial localization to produce the final detection. This two steps localization allows both the classification network to see a large region produced by the first localization and the second localization to produce a very tight box around the symbol to detect for a high precision detection. Localization and classification is learned jointly using a weighted multi-task loss eq. (2.1) composed of a mean squared loss for the localization L_{reg} and categorical cross-entropy for classification L_{cls}. To normalize the localization loss with respect to the classification loss, the localization loss is scaled up using a weighting coefficient λ.

$$L(t, t_{corr}, p) = L_{cls}(p, p^*) + \lambda \cdot L_{reg}(t + t_{corr}, t^*) \tag{2.1}$$

Here, p and p^* are respectively predicted class and ground truth class. t, t_{corr} and

t^* are respectively the initial transformation produced by the localization network, the transformation correction produced by the multi-task network and the ground truth transformation.

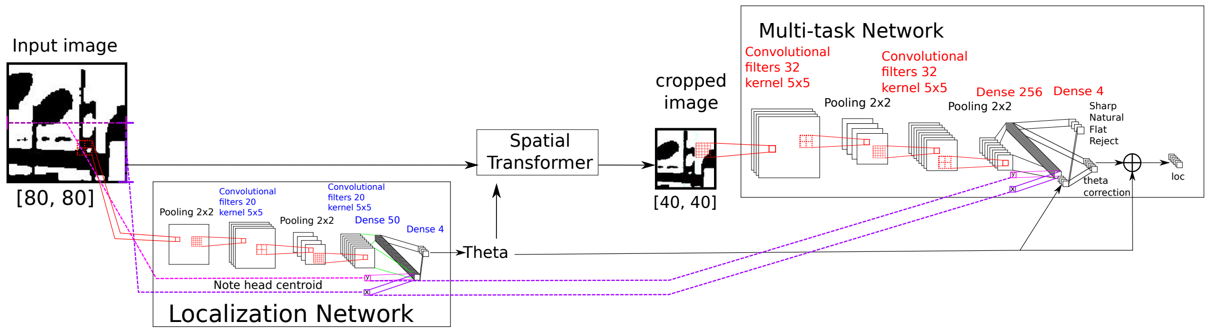


Figure 2.3 – Accidental detection using a Spatial Transformer (ST). The localization network takes an 80×80 image as input, and produces an axis-aligned bounding box represented by an affine transformation in 4 outputs nodes. The sub-image in the bounding box, resized to 40×40 pixels, is then classified by the Multi-task network into four classes (Sharp, Natural, Flat, or Rejection). The Multi-task network also refines the localization of the Localization network by producing offset for the affine transformation theta. We train the whole architecture end-to-end using a weighted multi-task loss composed of a classification loss and localization loss.

Use Of Contextual Information To this localization and multi-task network, we propose an improvement in order to use more contextual knowledge available during a typical OMR workflow. Knowing that the position of the note head is strongly correlated to the position of the accidental, we provide this coordinate as an input feature using two additional neurons in the first feed-forward layer of the localization network and multi-task network. In the same manner, the affine transformation produced by the localization network is forwarded to the first feed-forward layer of the multi-task network to provide more contextual information.

We chose the origin to be the upper left corner of the window and normalize the coordinates using the size of the window, that is the bottom right corner has a coordinate of (1, 1). In the original training samples, every note head centroid will have the same coordinate (1, 0.5), because the window is positioned relative to the note head. However, if we use this contextual approach together with our bootstrapping strategies presented in section 2.2.1, the position of the note head centroid will move randomly relatively to the upper left corner of the window. With this contextual information, we hypothesize that the network will be able to correlate the note head centroid information with the position of the accidental to detect, maybe even discriminate the correct accidental to detect when multiple accidentals are present in the image.

2.2.3 Multiple RoIs Object Detector Approach

Instead of specializing our detection model to the data we want to process, we show in this section the use of more complex and general CNN based detectors like the Faster R-CNN, R-FCN and SSD previously presented in section 1.5.1. These detectors typically extract multiple Region of Interests (RoIs) from the input image and make either two steps detection (Faster R-CNN and R-FCN) or single step detection (SSD). This strategy has the advantage of dramatically improving detection precision at the expense of adding a lot of computation, as discussed in section 2.2.5.3. We use the official implementation of [Huang 17] that can be used to train and compare Faster R-CNN, R-FCN and SSD models. For the Faster R-CNN and R-FCN models, input images are typically scaled to 600 pixels on the shorter edge while keeping a maximum width or height of 1,024 pixels. The SSD models only takes fixed size input images of 300×300 pixels. We propose to reuse our cropping strategy presented in section 2.2.1 for the input of the network. This strategy produces relatively small images of around 130×130 pixels. However, we note that the up-sampling to the normal input size of the different object detector models does not deteriorate the image like a down-sampling strategy and it also allows us to use an unmodified Resnet 101 and MobileNet v1 feature extractors, without having to change ratio and scales of anchor boxes. Further work could be done to better adapt these architecture by reducing the expected input size of the Faster R-CNN and other models, which would in turn reduce the processing time and the need to pretrained weights.

Faster R-CNN Our objective is to build the most precise music symbol detector possible in order to minimize errors early in the OMR pipeline. That is why we use the Resnet 101 feature extractor that produces excellent accuracy while providing pretrained weights on the Common Objects in Context (COCO) dataset for both Faster R-CNN and R-FCN models. The COCO dataset is a large object detection dataset containing around 200K images with 1.5 million object instances and is currently one of the major dataset used to train and evaluate object detection models. The ability of using pretrained weights is essential because our dataset of 2,955 examples is far too small to train these complex architectures from scratch. By using transfer learning, we can reduce over-fitting and benefit from the start of powerful feature extractors learned on the COCO dataset. It is therefore a way for us to reduce the amount of training data needed to produce an accurate detector.

Using the Faster R-CNN, we also propose to combine this complex object detector architecture with our previous propositions of bootstrapping and concatenation of contextual information (section 2.2.1). We experiment using the best performing bootstrapping method, which is *novertical* as shown in table 2.5, in order to augment the number of training samples. In combination with the bootstrapping method, which could lead to a confusion for the object detector of the correct accidental to localize, we concatenate the (x, y) coordinate information of the center of the note head to the first fully-connected layer after the crop and resize operation of the selected RoIs. Originally, the coordinates of the center of the note head are relative to the top left corner of the original image and scaled relatively. Because of the crop operation of RoIs, we duplicate the note head centroid position of one dataset example for each RoI and translate and scale the coordinate relatively to the cropped area. The Faster R-CNN, as well as the R-FCN and SSD, are trained using the classical multi-loss function, combining a classification loss L_{cls} (Softmax) and localization loss L_{reg} (Smooth L1):

$$L(a_i, I) = L_{cls}(p_i, p_i^*) + \lambda \cdot [a_i \; is \; positive] \cdot L_{reg}(t_i, t_i^*) \tag{2.2}$$

For each anchor a_i of image I, we search for the best matching predicted box t_i. If such a box exist, a_i is assigned to be *positive* and enable the localization loss L_{reg}. p_i and p_i^* are respectively the predicted class and the ground truth class, t_i^* is the ground truth bounding box associated with the anchor a_i. Here, $\lambda = 2$ meaning that the localization loss has twice as much weight as the classification loss. All other parameters are left to their default values.

R-FCN R-FCN is very similar to the Faster R-CNN except for optimization in how the RoIs are computed and extracted from the feature extractor. This led to a significant speed-up as shown by Huang et al. [Huang 17] and our own results in section 2.2.5. The loss function is the same as the Faster R-CNN, see eq. (2.2).

Single Shot Detector By doing detection in single step fashion, the Single Shot Detector is able to produce multiple detection with much faster speed than the Faster

R-CNN and the R-FCN. We also use a different, more lightweight, feature extractor known as MobileNet v1. We resize all input images at 300×300 pixels as the model does not accept variable size input. No bootstrapping and contextual information was used for the R-FCN and SSD detectors. We use the same multi-task loss function as the Faster R-CNN, but use a weighted sigmoid function for the classification loss. All parameters are left by default and use $\alpha = 1$ meaning that both classification and localization loss has the same weight.

2.2.4 Training Protocol

Contextual Bootstrapping Approach The training of our Spatial Transformer architecture, shown in fig. 2.3, used in our contextual bootstrapping approach is done in a single end-to-end approach using a multi-task loss function composed of a categorical cross-entropy loss for classification and mean-squared error loss for localization. The normalization of the two losses is done by multiplying the localization loss with a weight coefficient. After a quick grid search for this weight parameter in {1,5,10,15,20}, the best results were obtained using a value of 20. The network is trained using the Adam backpropagation algorithm with a learning rate of 0.0001 and a batch size of 50.

Multiple RoIs Object Detector Approach For training the Faster R-CNN, R-FCN and SSD models, we mainly reuse the recommended parameters by Huang et al. [Huang 17]. We chose to use pretrained weights on the COCO dataset for all feature extractors used: Resnet 101 and MobileNet v1. We train the Faster R-CNN and R-FCN with the Stochastic Gradient Descent (SGD) optimizer configured with a learning rate of respectively 0.0001 and 0.0003. For the SSD, we use the RMSProp optimizer with a learning rate of 0.004 and a batch size of 24.

Cross-Validation Our dataset consists only of 2,955 original images with very imbalanced classes. We do a 5 fold cross-validation in order to test our different approaches to produce reliable results. This strategy was implemented by splitting the original dataset of 2,955 images into 5 folds of $\sim$593 examples. We iterate 5 times and each time we choose a different fold to be the testing fold and use the remaining 4 folds for training. In the context of bootstrapping as seen in section 2.2.1, we make sure that the data augmentation only operates on the training folds and happens only after

Table 2.2 – Architecture and Data Usage for Accidental Detectors. Detector type are Spatial Transformer (ST), Faster R-CNN, R-FCN, and SSD. The table shows whether detectors use transfer learning, can detect multiple objects, make use of the associated note head location, or use bootstrapped samples. Version labels (v1, v2, ...) are used to differentiate different experimental conditions for the same detector and are reused in table 2.3.

Detector	Transfer Learn.	Mult. Objs	Note Head	Bootstrap
CNN + ST v1				
CNN + ST v2			✔	
CNN + ST v3				✔
CNN + ST v4			✔	✔
Faster R-CNN v1	✔	✔		
Faster R-CNN v2	✔	✔	✔	
Faster R-CNN v3	✔	✔		✔
R-FCN	✔	✔		
SSD	✔	✔		

the cross-validation splitting is done. That way, there are no possibilities that different bootstrapped images coming from the same original image are present in both training and test set. Using this cross-validation method, we therefore propose both the mean and standard deviation for every results presented in the next section.

An overview of all the combination of detectors and training parameters is shown in table 2.2. We now present the results of the different detection experiments we have done throughout this work.

2.2.5 Results

Using the cross-validation protocol described in the previous section, we evaluate our detectors using the mean Average Precision (mAP) metric proposed by the PASCAL VOC Challenge by Everingham et al. [Everingham 10]. This metric allows us to jointly evaluate classification and localization accuracy and compare the impact of bootstrapping of our ST-based detector in table 2.5. We also compare with state-of-the-art detectors in table 2.3. However, we make two small modifications to this metric. The mAP metric uses an IoU threshold in order to decide if a detection is a True Positive (TP) or a False Positive (FP). It is common to use 0.5 as the IoU threshold for object detection. In our context of precise music symbol detection, we propose to add a second threshold of 0.75, which is much stricter and more representative of the level of precision we want to obtain. Also, rejection (i.e., the absence of any detection target) is not considered in the original mAP metric. That is why we propose to ignore the localization if the model correctly predict the input image to be a rejection (no accidental). Although, we define our rejection task to localize the note head, our objective is to give the network something stable to localize in order to simplify the rejection.

2.2.5.1 Detector Comparison

Using the mAP metric, we show in table 2.3 the performance of our different approaches. We can clearly see that results are very good with an mAP of $\sim$99% with an IoU threshold of 0.5 except for our ST-based detector which only performs at $\sim$97%. Using an IoU threshold of 0.75 more clearly distinguish the detectors and place first the R-FCN with a mAP of 98.7%, then Faster R-CNN followed by the SSD and finally the ST-based detector. These results show the clear superiority of using multiple RoIs generated from different part of the images instead of a single RoI from the whole image.

Table 2.3 – Results comparing the best Spatial Transformer (ST) based detector, Faster R-CNN, R-FCN and SSD. Results shown are mAP (in %) with an IoU threshold of either 0.5 or 0.75. See table 2.2 for an overview of each detectors.

Detectors	mAP with IoU > 0.5		mAP with IoU > 0.75	
	$\mu(\%)$	$\sigma(\%)$	$\mu(\%)$	$\sigma(\%)$
ST v4	97.25	1.68	94.81	2.99
Faster R-CNN v1	98.73	0.94	98.34	0.73
Faster R-CNN v2	98.85	0.67	98.65	0.59
Faster R-CNN v3	86.91	3.79	84.80	3.86
R-FCN	**99.17**	**0.30**	**98.73**	**0.40**
SSD	98.93	0.67	97.81	0.92

However, more complex detectors come with additional overhead, as shown in table 2.4. The Faster R-CNN is about 90 times slower than the ST-based detector, and takes about 18 times more memory. Given that this detector will be used extremely frequently by the OMR system (more than a thousand of calls by page of music score), using a Faster R-CNN will provoke a significant slow-down of the recognition process. The slow processing time of state-of-the-art detectors are largely explained by the fact

Table 2.4 – Speed and memory consumption of the ST-based detector, Faster R-CNN, R-FCN and SSD. Measures were taken on a Nvidia GPU K80.

Detectors	Speed (ms/image)	Memory (Mb)
ST	2	260
SSD	14	300
R-FCN	80	4,800
Faster R-CNN	180	4,800

that they were designed to process much larger images. Further work could be done to reduce the size of these architectures in order to reduce the processing time but we did not have the time to do this very difficult process of downsizing all three Deep Learning detection networks and tuning again all the hyperparameters of the networks.

2.2.5.2 Impact Of Bootstrapping Techniques And Contextual Information

In the case of the ST-based detector, we found that the augmentation of training data almost always leads to better localization as shown in table 2.5. Another interesting observation is that different sampling strategies led to different results. The unconditional inclusion of the note head in the sampled image does not lead into an improvement, which seems to correlate with the property of translation invariance found in classical CNN architecture based on convolution and pooling operation. We also found that reducing the vertical displacement of the sampled images relatively to the vertical position of the note head lead to better results than allowing an unconstrained positional sampling. This seems to confirm our starting hypothesis that introducing variation in a very stable characteristic of our data does not help the ST-based detector to converge.

In case of the use of the Faster R-CNN, we found that bootstrapping techniques actually hurt the precision of the detection. To better analyze this result, we divided the dataset into four categories: *single accidental* where only one accidental is visible in the image, *multi accidental* where multiple accidentals are visible, *reject without accidental* where no accidental are visible and finally *reject with accidental* where an accidental is visible in the image but should not be detected as it is not associated with the correct note head. We found that when using bootstrapping the results for both *multi accidental* and *reject with accidental* decreased significantly: 11% decreased for mAP with IoU $>$ 0.5 and 17% decreased for mAP with IoU $>$ 0.75. Our conclusion is that the architecture of the Faster R-CNN, designed for multi-object detection with strong translation invariance using the anchor boxes mechanism, is not suitable to be used in combination with bootstrapping for our particular task of contextual detection.

Again, for the ST-based detector, we found that the use of contextual information like the centroid position of the current note head always helps the detector improve the detection results. In contrast, this additional information for the Faster R-CNN did not change anything to the results.

Finally, for the ST-based detector, combining bootstrapping techniques and contextual information lead to an improvement of 9.3% mAP for an IoU threshold of 0.5 and 30.8% mAP for an IoU threshold of 0.75 (line 1 and 6 comparison in table 2.5). In the contrary, for more complex detectors like the Faster R-CNN, the use of contextual information or bootstrapping techniques did not improve the already very good results.

Table 2.5 – Mean Average Precision results (in %) with IoU > 0.5 or > 0.75 (5-fold cross-valid., 2,955 samples) for the Spatial Transformer based detector, fig. 2.3. We explore the use of: *nh* which means the note head centroid is provided to the input of the network, different data *quantity*, different *bootstrapping methods* and different localization *loss weight*.

Spatial Transformer-based Detector Conditions				mAP with IoU 0.5		mAP with IoU 0.75	
nh	quantity	bootstrapping method	loss weight	$\mu(\%)$	$\sigma(\%)$	$\mu(\%)$	$\sigma(\%)$
✔	400k	novertical	20	**97.3**	1.7	**94.8**	3.0
✔	200k	novertical	20	96.0	2.4	92.0	2.8
✔	100k	novertical	20	94.6	3.5	86.1	6.3
✔	25k	novertical	20	92.9	3.1	68.0	1.5
✔	original		20	90.0	4.0	62.5	4.3
	original		20	88.0	3.6	64.0	3.7
	400k	novertical	20	94.3	3.5	86.2	5.9
✔	400k	novertical_notehead	20	96.0	**1.6**	92.2	**1.1**
✔	400k	unconstrained	20	94.4	2.1	92.4	2.3
✔	400k	notehead	20	95.7	1.9	88.4	2.5
✔	400k	novertical	15	96.8	1.6	93.6	2.7
✔	400k	novertical	10	96.6	1.9	93.8	2.2
✔	400k	novertical	5	96.4	1.7	91.3	4.1
✔	400k	novertical	1	95.6	1.9	89.7	3.5

2.2.5.3 Discussion

The results of our experiments, which we have now published in Choi et al. [Choi 17; Choi 19], show the clear superiority of the Faster R-CNN and R-FCN for the task of detecting an accidental. However, we also propose less powerful models like the

SSD and ST-based detector for having more efficient and faster inference time for less accuracy. Also, if we consider using semi-supervised or unsupervised architecture in order to resolve the detection of music symbols, the Spatial Transformer should be simpler to integrate as it was designed as an attention model and seamlessly integrates in any kind of neural network architectures.

Regarding the full OMR task, we only show here how to resolve a small subset of the pipeline. Future works will be oriented towards two main points: extend detection to other symbols, further reduce the number needed of training samples and propose a new corpus of dense and complex orchestral printed scores.

We plan to extend the detection of music symbols in a bottom up fashion, first adding multiple note heads detection and then gracefully integrate accidentals, followed by articulation marks, ornaments. . .

More investigation is needed in order to characterize the relation between the accuracy of the detectors relative to the number of training samples and the size of the window in which we want to detect a symbol. Our focus on reducing the number of needed training samples is based on the observation of the fact that manually annotating data is very slow and costly. Moreover, it has to be done again each time we work in a new corpus/type of documents.

Even though the situation in OMR recently improved by the introduction of the MUSCIMA++ dataset which contains localization, class and relationship information of music symbols in handwritten scores, the dataset is still extremely homogeneous because the corpus was originally designed for a writer identification task where the same 20 music score pages were transcribed by 50 different musicians. That's why we intend to propose a new corpus of dense and complex orchestral and piano printed scores. We feel that the OMR community is neglecting printed scores because of recent software printed music scores with very good impression quality. We feel that the OMR community is neglecting printed scores because of the association with recent software printed music scores where readily available fully integrated OMR software systems (see section 1.3.2) can be used with great results. However, there is still a huge quantity of music scores printed or engraved from the 18th to the early 20th century. These scores present a lot of challenges because of their printing techniques, time degradation, bad scanning qualities and complexity of the baroque, classic or romantic music style.

2.2.6 Conclusion

A renewed interest is shown toward OMR in the computer vision and pattern recognition research communities, because many interesting challenges remain to be overcome. In this work, we concentrated on designing a method that produces an accurate segmentation and classification of the accidental associated with a note head, without a priori rules concerning segmentation problems. We propose four different detectors: a Spatial Transformer-based detector, SSD, R-FCN and Faster R-CNN. We show the trade-off in speed over accuracy in different detectors with the best detector having 98.73% mAP for an IoU threshold of 0.75. The fastest ST-based detector shows very bad out-of-the-box performance. However, by using contextual information like the position of the note head and bootstrapping techniques, we improve significantly the accuracy of the detection by 9.3% mAP for an IoU threshold of 0.5 and by 30.8% mAP for an IoU threshold of 0.75.

We have now shown the effectiveness of using small and large Deep Learning detectors on a simple constrained task of a single accidental symbol detection in a small patch of historical printed music score image. On the other hand, the object detection literature has shown that large Deep Learning detector architecture can scale to much larger input images and larger class set of object to detect. This is why we now propose to apply large Deep Learning detectors such as a Faster R-CNN to a much larger music symbol detection task using the MUSCIMA++ dataset.

2.3 Large Scale Supervised Music Symbol Detection

Existing state-of-the-art object detectors such as Faster R-CNN or R-FCN were designed to detect objects in natural scenes and have been shown to work well on challenging datasets such as COCO [Lin 14] or ImageNet [Deng 09]. In this work, we propose to apply such detector on the challenging task of detecting a large class set of music symbols in handwritten scores. But applying them out-of-the-box on sheets of music can lead to a suboptimal performance, due to the dense nature of music scores with many tiny objects. Therefore, we now present the MUSCIMA++ dataset and various preprocessing operations to tailor the detectors used to our specific music symbol detection task.

2.3.1 MUSCIMA++ Dataset for Handwritten Symbol Detection

For training a music object detector, we use the MUSCIMA++ dataset [Hajič 17] based on the CVC-MUSCIMA dataset [Fornés 12]. The CVC-MUSCIMA dataset is constituted of 1,000 music sheets and was produced by 50 different musicians writing the same 20 music pages in order to capture a large variability in handwriting. Hajič et al. [Hajič 17] later augmented this dataset with over 90,000 symbol-level annotations, made by human annotators across 105 different classes of music symbols. The images have a high resolution of about $3,500 \times 2,000$ pixels, are binarized and optionally come with staff lines removed. For consistency, all white-on-black images are first inverted and then converted to RGB, as the evaluated implementations take colored images as input. Note that the overhead created by this conversion is only minimal, as the duplicated information gets merged again in the first layer of the CNN.

To efficiently train an object detector on such images, the image size has to be reduced in order for the entire process to fit in the GPU virtual memory. We propose to crop the images contextually, by cutting images first vertically and then horizontally, such that each image contains exactly one staff and has a width-to-height-ratio of no more than 2:1, with about 15% horizontal overlap to adjacent slices (see fig. 2.4). Basically, each horizontal slice extends from the bottom of the staff above to the top of the staff below. This cropping can also be done by automatically detecting staffs and then applying the same slicing rules leading to image crops that partially overlap both horizontally and vertically. For splitting the cropped images into a train and test set, we follow the recommendations from Hajič et al. [Hajič 17] to ensure that the test set contains scores of all complexities and that there is no overlap of writers between the training and the test set. We furthermore used 10% of the remaining training set for validation during the training. In total, we obtained 6,181 samples, that were divided into a training, validation and test set, containing 4,794, 533 and 854 images respectively.

One limitation of this cropping approach is, that all objects significantly exceeding the size of cropped regions along the staff lines like big slurs or big system braces, will not appear in the training and test data, as only annotations that have an intersection-over-area (IoA) of 0.8 or higher between the object and the cropped region are considered part of the ground truth.

As music objects, we consider the full vocabulary of all 105 classes contained in the MUSCIMA++ dataset, containing both primitives such as note heads as well as compound objects such as key-signatures that consist of one or multiple accidentals.

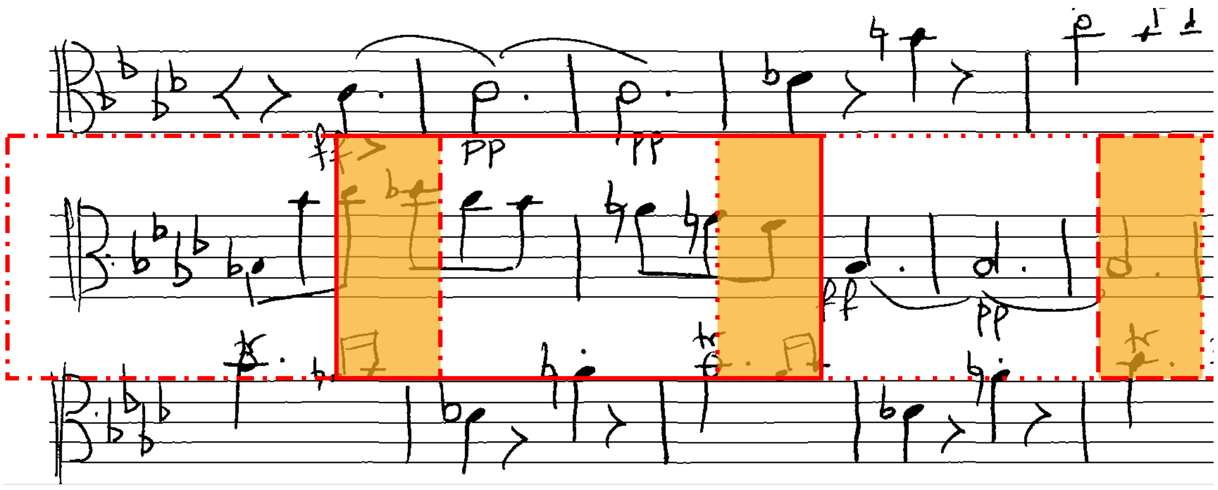


Figure 2.4 – Illustration of the sliding window approach, used to crop music scores into meaningful sub-images (red) with horizontally overlapping areas (orange) between adjacent crops.

As said before, big symbols and symbols on the edge of the region are automatically discarded based on their IoA.

2.3.2 Experimental Design

This experiment was done in collaboration with Alexander Pacha from the Vienna University of Technology. For evaluating our suggested approach, we conducted several experiments to study the performance of various object detectors and feature extractors, as well as the effects of staff line removal, transfer-learning and removing classes with rare symbols. Using the deep learning library TensorFlow [2], we adapted the work from [Huang 17] to detect music objects by training on the data described in section 2.3.1. The entire source code, including training protocols and detailed instructions to reproduce our results, can be found at `https://github.com/apacha/MusicObjectDetector-TF`. We considered:

— The three meta-architectures Faster R-CNN, R-FCN, and SSD as object detectors. Faster R-CNN and R-FCN are both two-stage detectors with a region proposal network and a region classifier. The difference is that Faster R-CNN uses a sliding window for classification, whereas R-FCN uses position sensitive score maps and per-RoI pooling, which is more efficient at the cost of a slightly reduced precision.

2. `https://www.tensorflow.org/`

SSD is a generalized region proposal network for one stage detection on multiple feature maps

- ResNet50, Inception-ResNet-v2, MobileNet-v1 and Inception-v2 as feature extractors, explicitly excluding custom-made networks that cannot benefit from transfer-learning
- Images with and without staff lines (based on the images provided along the CVC-MUSCIMA dataset)
- The full vocabulary of all 105 classes included in the MUSCIMA++ dataset, as well as a reduced set of only 71 classes, removing 34 classes that appear less than 50 times in the ground truth and are only of minor importance such as uncommon numerals and letters. Exceptions were only made for the classes double sharp and the numerals 5, 6, 7 and 8: although they appear less than 50 times in the dataset, we consider them essential to recover music semantics such as pitch and time signature.

All the above-mentioned object detectors have a certain set of hyperparameters that need to be fine-tuned for the particular dataset. For example, Redmon et al. [Redmon 16] shows that using statistical analysis to obtain a sensible number of anchor boxes, anchor box sizes, and anchor box ratios can improve the results significantly compared to handpicked priors. When running similar analysis on the cropped images, we obtain the following characteristics: for a typical input image of 600 pixels width and 300 pixels height (see fig. 2.5), we found the average square box size is about 37 pixels with a standard deviation of 48 pixels. Note, that the dataset also contains extreme cases of small objects like dots with only a few pixels and large objects that spans hundreds of pixels. The mean ratio from width to height of boxes is 0.7 which means that the majority of boxes are higher than they are wide. Furthermore, cropped images that are to be fed to the detector contain 19 symbols on average, with a standard deviation of 11. Concluding the analysis, we decided to use a grid of 32×32 pixels with a stride of 8 pixels and aspect ratios of 0.06, 0.29, 0.48, and 2.2 with the scales 0.25, 0.5, 0.75, 1.0, 1.75, and 4.0 to reflect the wide range of object shapes in the dataset.

2.3.3 Results

Following the evaluation protocols of the Pascal VOC challenge [Everingham 10], we report the mean average precision (mAP) for each completed training in table 2.6

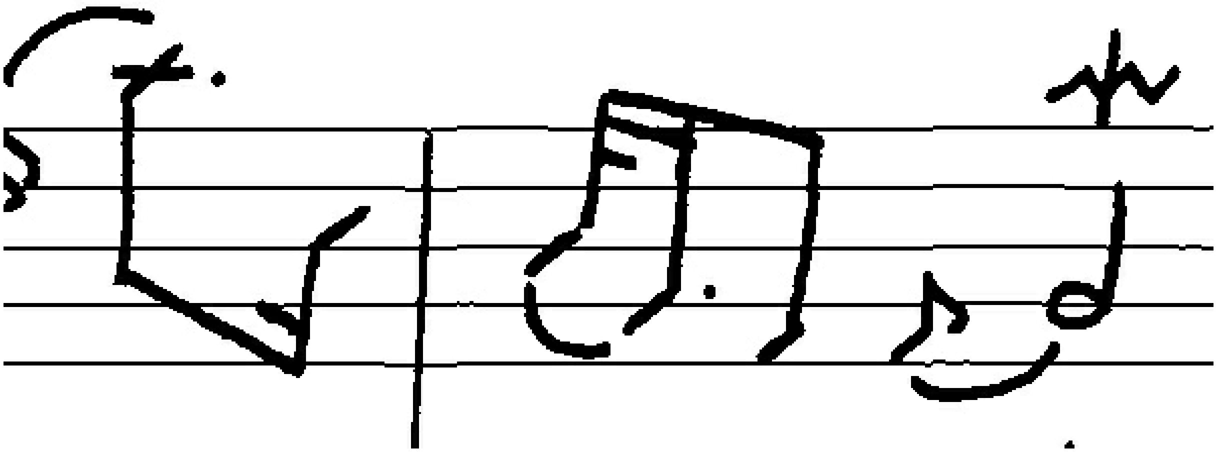


Figure 2.5 – Typical sample of a cropped image that serves as input for the music object detector.

and the detailed average precision per class for the combination that yielded the best results in table 2.7. Figure 2.6 shows a typical detection within a single image.

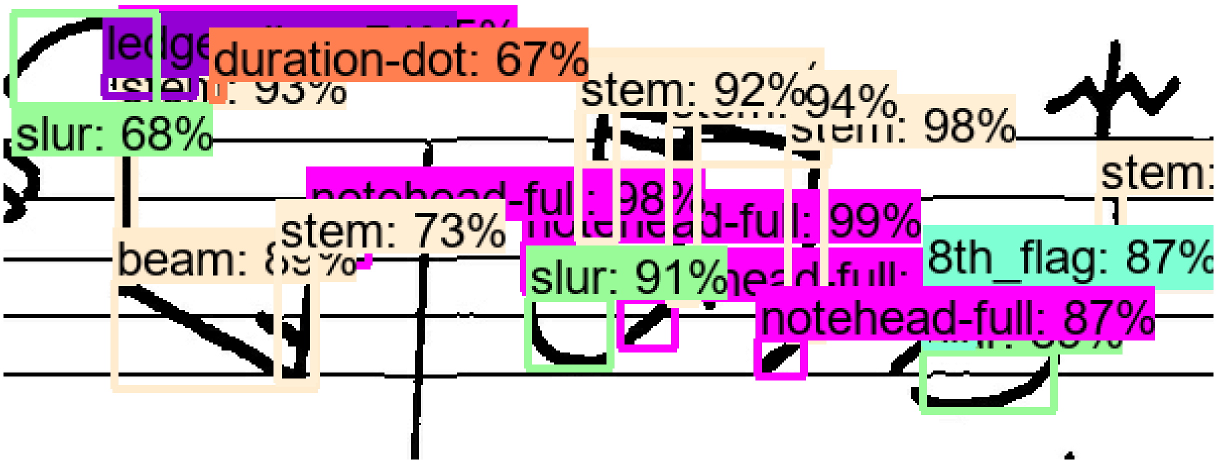


Figure 2.6 – Typical detection results with most symbols recognized correctly.

Table 2.6 – Detailed results for various hyperparameter combinations of the music object detector. We present the mean Average Precision (mAP) and Weighted mean Average Precision (WmAP) on the test set of MUSCIMA++ to show the influence of class imbalance.

Meta-Architecture	Feature Extractor	Nb of classes	Staff lines	mAP (%)	WmAP (%)
Faster R-CNN	Inception-ResNet-v2	105	✔	81.56	94.22
Faster R-CNN	Inception-ResNet-v2	105		81.23	94.56
Faster R-CNN	Inception-ResNet-v2	71	✔	85.12†	94.68
Faster R-CNN	Inception-ResNet-v2	71		87.80‡	95.05
Faster R-CNN	ResNet50	105	✔	76.39	93.07
Faster R-CNN	ResNet50	105		78.45	93.10
Faster R-CNN	ResNet50	71	✔	82.30	93.47
Faster R-CNN	ResNet50	71		84.85	93.63
R-FCN	Inception-ResNet-v2	105	✔	69.75	89.12
R-FCN	Inception-ResNet-v2	105		70.88	89.42
R-FCN	ResNet50	105	✔	75.53	92.59
R-FCN	ResNet50	105		74.29	92.33
SSD	Inception-v2	105	✔	71.52	82.44
SSD	Inception-v2	105		70.40	81.75
SSD	MobileNet-v1	105	✔	62.30	74.97
SSD	MobileNet-v1	105		61.56	76.74

Table 2.7 – Detailed precision results per class for the best obtained music object detector on the reduced set of classes (see table 2.6, line 3† and 4‡).

Class name	Total number of instances	Average precision on the test set (%)	
		with staff lines†	without staff lines‡
notehead-full	31,084	99.85	99.64
stem	27,108	98.82	98.71
ledger_line	14,500	97.89	97.40
beam	8,677	93.86	94.57
slur	3,859	90.34	88.54
duration-dot	3,195	95.12	94.21
thin_barline	3,071	99.49	99.64
8th_flag	2,744	93.46	93.37
measure_separator	2,649	43.64	52.09
staccato-dot	2,507	94.23	94.97
sharp	2,420	99.42	99.46
notehead-empty	2,385	99.31	99.11
flat	1,467	96.97	97.98
natural	1,427	96.90	97.61

Continued on next page

Table 2.7 — *Continued from previous page*

Class name	Total number of instances	Average precision on the test set (%)	
		with staff lines†	without staff lines‡
dynamics_text	1,374	85.25	87.12
8th_rest	1,339	98.86	99.36
tie	1,085	82.39	81.85
quarter_rest	1,060	96.05	96.78
letter_p	1,038	89.70	89.84
letter_f	1,035	93.10	92.77
letter_e	926	82.12	85.29
letter_r	750	51.64	62.25
key_signature	697	79.31	77.80
letter_o	655	94.47	93.82
16th_flag	652	36.62	40.19
letter_s	649	71.89	74.30
grace-notehead-full	576	85.75	85.37
numeral_3	548	98.73	98.04
16th_rest	531	96.17	99.93
letter_t	513	92.10	94.42
other_text	508	83.99	89.30
letter_c	469	89.82	88.57
tuple	459	30.41	77.11
accent	421	99.08	95.75
g-clef	403	100.00	100.00
other-dot	402	94.40	95.19
repeat-dot	359	99.75	100.00
trill	315	100.00	99.74
letter_d	313	93.49	89.36
letter_m	293	74.19	74.43
f-clef	285	100.00	98.21
half_rest	241	95.53	91.16
time_signature	221	96.33	95.02
tenuto	216	88.45	74.79
letter_l	192	78.75	86.00
c-clef	190	97.68	98.68
whole_rest	183	90.73	84.66
letter_P	177	45.83	45.80
tempo_text	174	69.40	78.32
letter_i	171	66.48	81.08

Continued on next page

Table 2.7 — *Continued from previous page*

Class name	Total number of instances	Average precision on the test set (%)	
		with staff lines†	without staff lines‡
letter_n	164	79.51	80.26
numeral_4	155	99.60	99.47
letter_a	134	90.36	83.81
multiple-note_tremolo	126	81.01	82.42
ornaments	123	85.22	83.90
letter_M	115	65.83	71.47
grace_strikethrough	110	98.14	97.96
letter_u	106	65.98	62.69
repeat	73	84.42	88.87
double_sharp	44	100.00	100.00
numeral_2	40	100.00	92.50
numeral_6	36	100.00	100.00
numeral_8	36	100.00	91.67
numeral_7	24	28.32	62.59
numeral_5	11	26.67	100.00

We find that the best performing detector with regard to precision is the Faster R-CNN using the Inception-Resnet V2 feature extractor, pretrained on the COCO dataset. This model produces a mAP of over 80%. The training on a GeForce GTX 1080 Ti takes approximately one day per configuration before results become stable. Validating ~ 500 images takes about 2–4 minutes, so inference should take less than half a second per (cropped) image. When comparing the results of training on images with and without staff lines, the impact is no longer significant, supporting the claim of Pugin [Pugin 06], that staff line removal might no longer be necessary. However, readers should also note that the staff lines in the CVC-MUSCIMA dataset are synthetic and do not experience the usual distortions that apply to scans or pictures of real music scores.

Other detectors like the R-FCN or SSD produce good results as well, with a mAP of 75% and 71% respectively. Our results, therefore, comply with the findings of Huang et al. [Huang 17], where in particular the SSD model trades smaller accuracy for higher processing speed. Using pretrained weights, instead of random initialization and the RMSProp optimizer as opposed to Stochastic Gradient Descent, improved the results significantly, sped up convergence and was therefore used throughout the experiments. Modifying the set of classes by removing underrepresented classes as described in section 2.3.2, boosted the mAP by up to 6% in some cases. Note, that table 2.7

is missing six classes, that did not have any instances in the test set because they exceeded the size of the image crops and were thus discarded during the preprocessing.

2.3.4 Conclusion

In this work that we have published in Pacha, Choi, Coüasnon, Ricquebourg, Zanibbi, and Eidenberger [Pacha 18b], we show that state-of-the-art deep learning detectors like Faster R-CNN, R-FCN and SSD can produce accurate detection results on a wide range of music symbols. After optimizing different hyperparameters, we achieve a mAP of over 80%, which is a solid baseline.

However, there are still a couple of open issues, that need to be addressed in future work, like how to process a whole page of a score. In this work, we used a simple overlapping sliding window approach. This method, although simple to use, has many well-known downsides like the poor performance of processing empty images or cutting up large symbols as well as a non-trivial merging step that has to fuse information from multiple overlapping sections.

Another problem, specific to OMR, is the inherent imbalance of symbol classes: some symbols like note heads are extremely frequent whereas others like double sharps are rare and often tied to a specific type of score. Having experimented with state-of-the-art deep learning object detectors, we found that classes do not interact with each other: simplifying the task by removing line-shaped classes did not improve the overall precision. There also seems to be a minimum threshold of about 20 samples per class, in order to be meaningful during the training. Currently, there is no guarantee, that the model does not overfit, but with recently published work like the RetinaNet and its focus loss [Lin 17] the effects of this class-imbalance could be mitigated to improve the training, especially on hard to detect classes.

Although we used the test set, proposed by the MUSCIMA++ authors, where writers in the test set do not appear in the training set, we are still not certain whether this system is truly writer independent or not. One way to confirm this would be to perform a cross-validation, where each writer in the dataset is evaluated independently.

Finally, we have shown that removing staff lines can be omitted for music object detection, when using CNNs. Future experiments that apply data-augmentation using noise models and deformed images, as proposed for the staff removal challenge [Fornés 13], can give even more insights into the robustness of our approach.

2.4 Conclusion

In this chapter, we have shown that fully supervised Deep Learning-based detector can be successfully applied to the task of detecting music symbols. We propose in section 2.2 to use our new, very small and fast Spatial Transformer-based detector for detecting a single accidental symbol in a small image patch of dense and noisy historical printed music score. We constrain this detection task using a hybrid approach of first preprocessing the score reusing the DMOS syntactical method and its music grammar. While the standard use of the music grammar is the reconstruction of the music notation structure once all the low-level graphical recognition tasks are done, it can also be used as a preprocessing engine. By using already recognized contextual information, it can identify RoIs (Regions of Interests) that can be use to train new detectors on new class of symbols. We use such mechanism to train the second part of our hybrid approach and apply our Deep Learning detector for localizing the subset of symbols we are interested in.

With the success of this first approach and an almost perfect recognition rate confirmed by state-of-the-art detectors like the Faster R-CNN, we then propose to apply state-of-the-art detectors in a much more challenging detection task of general handwritten music symbol detection task. In section 2.3, we propose a baseline for handwritten music symbol detection using the MUSCIMA++ dataset. Although, we could not directly apply state-of-the-art detectors to a whole music score page, we show that competitive results can be obtained by preprocessing the score into small sub-regions.

While the task of detecting music symbols for OMR is far from being a solved problem, we now have shown that state-of-the-art detectors can be applied to obtain detection with a sufficient accuracy for the overall OMR process. However, fully supervised detector models used in this chapter have the need for large fully annotated detection dataset in order to be trained. Although we have tried in section 2.2.1 to minimize the size of the dataset used by using bootstrapping method, detection datasets are generally very slow to produce, where the human annotator has to localize a symbol by drawing a bounding box and identify the symbol by attaching a label to the symbol. As an example, the creators of the MUSCIMA++ dataset reported that it took 400 hours to manually annotate the whole dataset.

We believe that the need for such music symbol detection dataset can not be annotated manually in the long run because of the lack of time and funds of OMR researchers. Therefore, it is of the utmost importance to discover new symbol detection approaches able to deal with non-annotated corpus of documents. In the coming chapter, we now review and discuss the existing approaches to unsupervised music symbol detection such as synthetic data generation as an answer to the lack of annotated data for supervised music symbol detection.

CHAPTER 3

LEARNING SYMBOL DETECTION WITHOUT MANUAL ANNOTATIONS

3.1 Introduction

In the previous chapter, we have demonstrated the application of Deep Learning-based detectors for the task of music symbol detection. We trained small (ST detector, SSD) and large (Faster R-CNN, R-FCN) detector models on a constrained detection task (single accidental in a small image patch) as well as a more general detection task (large set of music symbol classes in medium size images) and produced highly accurate detection suitable for an OMR system. While much more work still is needed for improving the detection of music symbols in larger images (whole page detection) and different music scores types, we believe we have shown solid baseline results that demonstrate the interest of using Deep Learning-based detectors that are able to seamlessly adapt to a new corpus of music scores documents.

However, during our work on accidental detection in dense and noisy historical printed music scores presented in section 2.2, we faced the major problem of not having an already available music symbol detection dataset suitable for our task. With our hybrid approach of using a syntactical method driving our Deep Learning-based detector and using data augmentation techniques like bootstrapping explained in section 2.2.1, we managed to reduce the amount of manual annotation to a minimum. Nonetheless, this approach required the manual annotation of $\sim 10,000$ small image patches by drawing a bounding box around accidental symbols and affecting a label to each bounding boxes.

For the document recognition community, this lack of annotated dataset is quite a common challenge faced for each new type of documents we want to process. The general solution to this problem has often been to use synthetically generated data to train Deep Learning model by using software such as DocCreator [labri 18]. Then the recognition process is bootstrapped by applying these pretrained models to the real documents.

In this chapter, we review how synthetic data can be used to bootstrap the recognition of historical printed scores with no previously existing dataset. We first consider how synthetic music scores can be used to train Deep Learning models in section 3.2 and discuss the advantages and shortcomings of using whole page synthetic scores. In section 3.3, we argue for a simpler approach for synthetic data generation that can be coupled with a generative method in order to do unsupervised music symbol detection.

3.2 Train With Synthetic Data

As mention previously, the lack of annotated dataset is quite a common challenge for the document recognition community. Existing Deep Learning models are often reused for a wide variety of task, going from document structure recognition to Handwritten Text Recognition (HTR). With the growing need for annotated data, the document recognition community has already constituted several large annotated dataset such as the IAM database [Marti 02], the RIMES dataset [Grosicki 08] or the READ dataset [Sánchez 16] that can be used mainly for HTR. However, it is unrealizable to manually constitute datasets of such magnitude for every type of documents and every kind of document recognition tasks.

One of the particularity of the document recognition domain is the fact that the heart of our work is to process automatically human artifacts. Indeed, documents are physical objects that are produced by humans and therefore often follows some notations or logic in their creation. For example, a letter will often follow the same structure, handwritten text will follow the writing convention of its language, mechanical drawing will use a certain set of standardized graphical components. Paradoxically, rules that define how a document should be formatted will be broken because of human unintentional or intentional mistakes or by natural degradation artifacts often present in historical documents.

Therefore, a lot of research efforts of the document recognition community has been directed towards understanding the production rules of document we want to recognize and capturing the possible variability of such rules. With the expert knowledge of how a document was produced and transformed, it is then possible to automatically produce synthetic documents that present the same format, content and variability as the original documents. The advantages of synthetically produced documents is the automatic production of associated annotations useful for training Deep Learning models for task like HTR or document structure recognition. Once a recognition model is trained on synthetic data, the model can then be applied to the original documents we wanted to process in the first place. An illustration of the whole process is given in fig. 3.1.

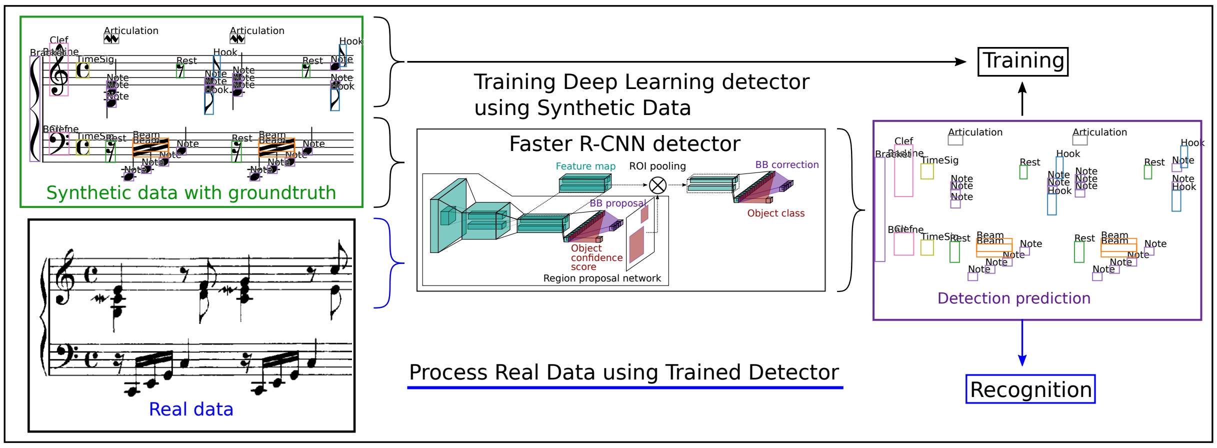


Figure 3.1 – Training Deep Learning detector with synthetic data and process real data using the final trained detector.

Following the need for synthetic data generation, we explore in the next section how synthetic data could be generated in the context of music score documents.

3.2.1 Synthetic Music Score Generation

In the context of printed music score creation, the production of such documents has been digitized since the advent of personal computers in the 1980s. Multiple high-quality music typesetting software exists such as Finale, Sibelius or MuseScore as explained in section 1.2.1. These computer programs propose a graphical user interface with which the user is able to interact and input music notes and symbols, either using a computer keyboard and mouse or by using a MIDI keyboard for a more intuitive process. To produce a digitized music score page, the music typesetting software uses a combination of drawing primitives like lines and curves to draw staff lines and slurs and glyphs from music fonts that follows the Standard Music Font Layout (SMuFL) standard [SMuFL 13]. The software is also able to automatically format the document so that it can maximize the readability of the score while respecting the complex set of rules of the music notation.

With the objective of producing synthetic printed music scores for OMR, the use of music typesetting software is a self-evident idea to produce potentially an infinite amount of synthetic data and we explored this idea in Choi et al. [Choi 18b]. However, given that the focus of such music typesetting software is to produce readable and beautiful music score document, it is not their explicit goal to imitate existing historical music score document. This leads to various limitation when trying to use such music typesetting software for OMR research.

The modern music notation is a loosely defined set of rules, giving lots of freedom to the engraver for the placement of music symbols. This leads to subtle biases in the position of music symbols either in historical printed music scores or in digitally produced music scores. Depending on the typesetting software used, it is possible to alter the parameters regulating the position of most symbols and therefore potentially adjust positional biases to match historical printed score. However, this process has never been studied and is very difficult to carry out without the use of an already existing real historical printed music score dataset annotated with the position of every symbol. We believe it is in our interest to propose an unsupervised music symbol detection method that is robust to such positional biases.

Moreover, music typesetting software produces document to be read by humans and often produces music score document in the PDF or MusicXML format. For tasks like music symbol detection or music notation reconstruction, such file formats do not contain sufficient information such as the label and position of music symbols, or the relations between symbols. However, the music typesetting software itself is able to produce such detail information about the music score and the open source MuseScore typesetting software [Schweer 18] has already proposed some ways of exporting detailed internal information for use by OMR software and researcher.

Finally, historical printed music scores present artifacts and degradation introduced by a manual engraving errors, bad preservation through time and scanning process. All of these transformations impact the final image to process and is often the biggest source of errors in the OMR process. The document recognition community has studied extensively artifacts and degradation present in historical document and software such as DocCreator [labri 18] exists for the purpose of artificially introducing noises and degradation artifacts in document images. However, it is still a difficult and manual process of choosing and adapting the correct noise models and degradation artifacts types that will correspond to the noises and degradations present in historical music scores. We also believe it is in our interest to propose an unsupervised symbol detection method robust to any kind of noises and artifacts present in historical printed music scores.

3.2.2 Problematic

The second half of the 18th century is often considered as the golden age of classical music and has produced a vast amount of printed music scores from the most famous compositors like Mozart, Haydn and Beethoven. We believe that the automatic processing of such historical scores is very important to further the conservation and utility of such document for musicologist and music enthusiasts alike. However, the recognition process of such historical printed scores is a very difficult challenge for OMR due to the complexity of the modern music notation, their manual production method (manual engraving) and degradation inflicted by time and scanning process. Moreover, the absence of existing annotated dataset of dense and noisy historical printed scores prevents the use of state-of-the-art Deep Learning recognition method for tasks like music symbol detection.

While the use of music typesetting software is an enticing path for synthetic music score generation, we have shown that there is a discrepancy between digital music scores and historical printed scores in the music notation, symbol shape and position and the document general appearance influenced by the degradation state of the historical document image. However, the use of synthetic data is not limited to the previously presented workflow in fig. 3.1. Synthetic data can be used in a weakly-supervised or unsupervised fashion as previously presented in section 1.6 with a data augmentation strategy or domain transfer strategy coupled with generative methods.

Following the general focus of our work, we believe that a new unsupervised symbol detection methodology is needed that can either replace fully supervised method or at least replace the need for manually annotated data. This method should be able to automatically adapt to a new corpus of documents and use a simple synthetic generation strategy to minimize the possible discrepancies between real data and synthetic data.

In the next section of this work, we propose an answer to this very difficult problem by presenting our new Isolating-GAN method.

3.3 Isolating-GAN for Unsupervised Symbol Detection

In this work, we propose a new unsupervised music symbol detection method called Isolating-GAN that is able to learn the task of music symbol detection in historical printed scores without using any manual annotations. In contrast to using a whole page of synthetic music score and complex synthetic generation procedure, our method uses a simpler synthetic data generation procedure needing only a small isolated music symbols dataset. Our strategy is to gradually simplify the complexity of the detection task by using the following iterative three steps method that we also illustrate graphically in fig. 4.1:

1. Identify Region of Interests (RoIs) possibly containing music symbols
2. Simplify the graphical representation by isolating music symbols to detect
3. Detect isolated music symbols

Under the hood, two main ideas form the basis of our method: Section 3.3.1 present the use of isolated music symbol to form a simple graphical representation for symbol detection while section 3.3.2 explains how to transform complex, dense and noisy images of historical printed music scores into a simple graphical representation.

3.3.1 Simple Graphical Representation for Symbol Detection

The fundamental idea behind the design of our method is to create an intermediate graphical representation as a platform for a trivial detection task. A symbol detection task is defined by the objective of predicting a bounding box and class label for each symbol present in an image. Therefore, the simplest detection task that we can artificially create is to synthetically generate an image only containing the symbols to recognize in a white empty background. Such images encode all and only the information needed for detection such as the shape and position of symbols to detect.

Such images can be automatically crafted using only isolated symbols randomly positioned and scaled in a blank image. However, the remaining problem is to know if we are able to adapt real images of historical printed scores containing background, noise and degradations signals to this simple graphical representation while preserving the shape and positional information required for accurate symbol detection predictions. In the next section, we present how we tackle this adaptation using a GAN-based generative method.

3.3.2 Using GAN for Isolated Symbols Domain Transfer

The use of a generative method is inspired by the previous cross-domain adaptation work like the Cycle-GAN previously presented in section 1.6.2 that presented a way of translating images of one representation domain into another representation domain while keeping essential characteristics common to the two graphical domains.

In our case, we propose to use a GAN-based generative method capable of adapting images of historical printed music scores into a simpler graphical representation containing only the isolated symbols to recognize and removing the noise and background information. To simplify, our generative process can be seen as a graphical filter isolating important music symbols we want to detect. However, this adaptation has to be done while keeping identical the size and position of the symbols of interests.

Knowing the relatively low limit in the image size that generative method such as GAN have, we also propose to limit the size of the image patches seen by the generative model using the DMOS syntactical method previously used in section 2.2.1 and identify small Regions of Interest (RoIs) with higher probability of containing the symbols we want to detect.

Once a small image patch of a real historical printed music score has been transformed into our simpler graphical representation containing only isolated music symbols in a white background, it is then trivial to detect such symbols using an isolated music symbol detector trained using synthetic data automatically produced using an already existing isolated music symbol dataset.

3.4 Conclusion

To summarize, the recognition of dense and noisy historical printed music scores is a very complex task lacking the required manually annotated detection dataset for applying existing fully-supervised state-of-the-art detector models. While the use of synthetic music scores as a replacement for manually annotated dataset is tempting, we believe a simpler synthetic generation method using only isolated music symbols coupled with a GAN-based generative method can better adapt to new unseen corpus of music score documents. We believe that this work is the first OMR work focusing on unsupervised music symbol detection and therefore we preferred simplifying our synthetic generation strategy to a maximum. We also believe that synthetic music scores generated typesetting music software could in the future be used together with our approach to improve and extend the reach of our work.

At the end of this chapter, we only have introduced the main ideas and concept underlying our method. In the next chapter, we present in details our new three steps iterative Isolating-GAN method and in chapter 5 evaluate in various experiments the detection accuracy and robustness of our method.

CHAPTER 4

UNSUPERVISED MUSIC SYMBOL DETECTION USING ISOLATING-GAN

4.1 Introduction

We propose now our novel approach to unsupervised music symbol detection of historical music scores by only using an isolated music symbol dataset as presented in section 3.3.

As we have previously discussed in section 3.2, the task of detecting music symbols on an entirely new corpus of music scores document images is very difficult to bootstrap when no music symbol detection dataset exists. This is the case for printed historical music scores where a handwritten detection dataset like MUSCIMA++ can not be used because of the large difference between handwritten symbols and printed symbols. Moreover, historical printed music scores are also very different from software generated music scores because of the manual engraving process of copper plates, see section 1.2.1, and artifacts and noises produced by time degradation as shown in fig. 1.4. Also, synthetic generation procedures of music scores can be really complex, has no guaranties of working and are difficult to generalize to a new dataset. In order for the detection model to correctly work on printed historical music scores, the training data should match the real documents structures, music notation and document degradation as closely as possible. Therefore, we need a new symbol detection method able to adapt to new corpus of unannotated music scores using as little as possible manual annotations such as an existing isolated symbol dataset.

As a result, we design a new three steps iterative method called Isolating-GAN for unsupervised music symbol detection illustrated in fig. 4.1:

1. Identifying regions of interests
2. Isolating music symbols

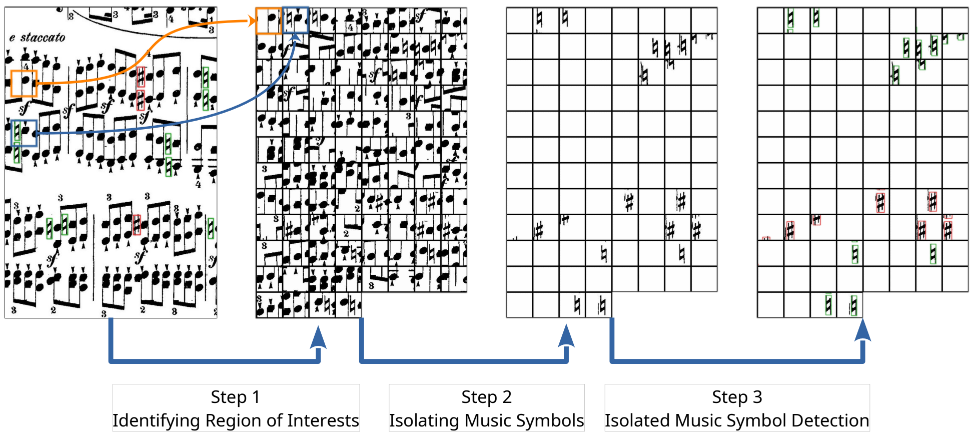

Step 1
Identifying Region of Interests
Step 2
Isolating Music Symbols
Step 3
Isolated Music Symbol Detection

Figure 4.1 – Our three steps Isolating-GAN method graphically illustrated. The complexity of the detection tasks is gradually reduced by 1. identifying RoIs possibly containing symbols to detect, 2. isolating music symbols by removing unwanted background information and finally 3. detecting symbols using a pretrained detector on existing isolated symbol dataset.

3. Detecting isolated music symbols

The first strength of our method is its iterative nature which gives us a lot of flexibility on how we can design our detection task. Using the first two steps, we can reduce the complexity of the detection task both spatially (section 4.2.2) and graphically (section 4.2.2). The other strength and main novelty of our approach is the generative nature of our second step, which allows us to adapt our detection model to any new corpus of documents as long as only the shape of the symbol we want to detect is known. Overall, the usefulness of our method is found in its sole and original use of isolated music symbols to train a precise music symbol detector without using any detection ground truth. We now introduce each of our three steps in the next section.

4.2 Objectives

The core proposition of our method is its original use of isolated music symbols to train a music symbol detector capable of locating symbols in real images of historical printed music scores. However, before applying our method, the corpus of document on which we will apply our method needs to be carefully studied so that we can correctly adapt our method to the data. In section 4.2.1, we present the characteristics of the type of documents we will be processing so that it can drive the application of our method.

Then, we introduce in section 4.2.2 how we can use isolated symbols as an intermediary step for the detection of music symbols in real images of music scores.

4.2.1 Real Data to Annotate

As explained before in section 2.2, we concentrate on the detection of music symbols in historical printed music scores of the mid-18th to mid-20th century. These documents present unique segmentation problems because of their music notation density typically present in orchestral or piano scores. The imprinting techniques of the era, previously presented in section 1.2.1, and time degradation also creates hard segmentation problems of touching and broken music symbols as shown in fig. 1.4. The lack of manually annotated datasets for symbol detection existing prior to this work makes this kind of document an ideal test bed to focus on unsupervised music symbol detection.

Another characteristics of these documents are the voluminous size of the scanned images. The causes are mainly the combination of fairly large original documents (A4 or larger) and the need for high resolution images that can capture enough details for an accurate detection. A good heuristic is that the minimum resolution required for a scanned image of a music score is to have a minimum height of 20 pixels between two lines of a staff which often translate approximately to a DPI of 300. Combined with a document containing 10 or more staves, this will produce scanned images of music scores with height and width in the magnitude of multiple thousands of pixels.

Finally, the music notation allows the use of a very large set of music symbols in order to better express the complexity of describing musical sounds. For example, the Standard Music Font Layout (SMuFL) [SMuFL 13] specify the use of at least 2,600 different glyphs that covers very diverse music notation systems, although each glyph does not necessarily translate into a musical symbol as some music symbols are constructed from smaller primitives. However, the use of the different music symbols are not equally distributed in a music score, where symbols like note heads are overwhelmingly frequent while others like the double sharp will be very rarely used. To illustrate this unbalanced distribution, the MUSCIMA++ dataset [Hajič 17] annotated 91,255 music symbols distributed in 105 different classes where 23,352 symbols are actually note heads, i.e. $\sim$25% of annotated symbols are note heads.

All these characteristics make the detection of music symbols in historical printed scores a very difficult task, especially when no annotated dataset exists to directly train a supervised detection model such as a Faster R-CNN. In contrast, we now present an iterative method where each step is able to reduce the complexity of the detection task step by step in order to accurately produce detection annotations without any ground truth data.

4.2.2 Overview of Unsupervised Detection of Symbols: Isolating-GAN Method

Our novel approach named Isolating-GAN to unsupervised music symbol detection can be broken down into three specific steps as shown in fig. 4.2.

Step 1: Identifying Region of Interests As explained in section 4.2.1, images of music score document can be very large when full pages are scanned at the appropriate resolutions.

Intuitively, a detection task complexity is directly related to the size of the area we search into. The larger the search area is, the more difficult it will be to fit the correct bounding box to the corresponding symbol. This problem is all the more relevant for generative models, which we use in the second step of our method, where generating high resolution images has been a notably difficult task as shown for example by Brock et al. [Brock 18] who only managed to generate 128×128 pixels high fidelity natural images. The size of the search area will also directly impact the virtual memory usage of the computing device, which will often be limited on GPU devices. Deep Learning models will typically adapt to larger input images by having deeper networks, which also increases training complexity, training times and the model memory usage.

Therefore, we propose in section 4.3 to reuse of the DMOS-PI syntactical method and its music grammar that we previously presented in section 1.4, to identify Region of Interests (RoIs) and reduce the amount of search areas for music symbols as illustrated in the step 1 of figs. 4.1 and 4.2. Using this method, we can also progressively build our detection ground truth for the corpus of music scores document by iteratively focusing on different set of music symbols and reusing previously detected symbols to improve the detection of a new set of symbols. For example, if we have already detected note head symbols, it is easy to know that regions at the left of note head symbols are likely to present accidental symbols.

Step 2: Isolating Music Symbols Using Isolating-GAN Once we identified region likely to contain symbols to detect, we propose to use our new Isolating-GAN model as an image-to-image transfer mechanism as explained in section 4.5. We train a U-Net generator of the GAN model to isolate existing music symbols in small input images and removing unwanted background noise or symbols as illustrated in the step 2 of figs. 4.1 and 4.2. However, this training is very unstable and this is why we show multiple ways of stabilizing the training process in section 4.5.2.1. The training strategy is designed by using an existing isolated music symbol dataset that will be used to generate image patches with isolated symbols on a white background. These generated images serves two purposes: one is to be the target representation that the GAN is trying to imitate. The other is to allow for a trivial detection task to be learned by a small and fast detector model like the SSD model which we now present.

Step 3: Isolated Music Symbol Detection After using the U-Net generator of the GAN model to transform and isolate existing symbols in real image patches of music scores, the final step of our method is to detect such isolated symbols using a small and fast detector such as the SSD model as illustrated in the step 3 of figs. 4.1 and 4.2. Our choice for this small and fast detector stems from the fact that the detection task is trivial and done on small image patches. We present how we pretrain the SSD model in section 4.4 together with the generation methodology using the isolated music symbol dataset.

The core of our proposition is the use of the Isolating-GAN model as a transfer function for symbols in real images of music scores into a graphical representation only containing isolated music symbols and a white background. In order to correctly evaluate this task in relation to our end-goal task of detecting music symbols in real music scores documents, we first present our detection method using the SSD model in section 4.4 which we will later reuse to evaluate the GAN model in section 4.5.3.1.

4.3 Step 1: Identifying Region of Interests

Due to the need to reduce the size of images that the GAN will take as input, we process the music scores document images with the DMOS syntactical method [Coüasnon 01] and present now the first step of our method. This method, previously

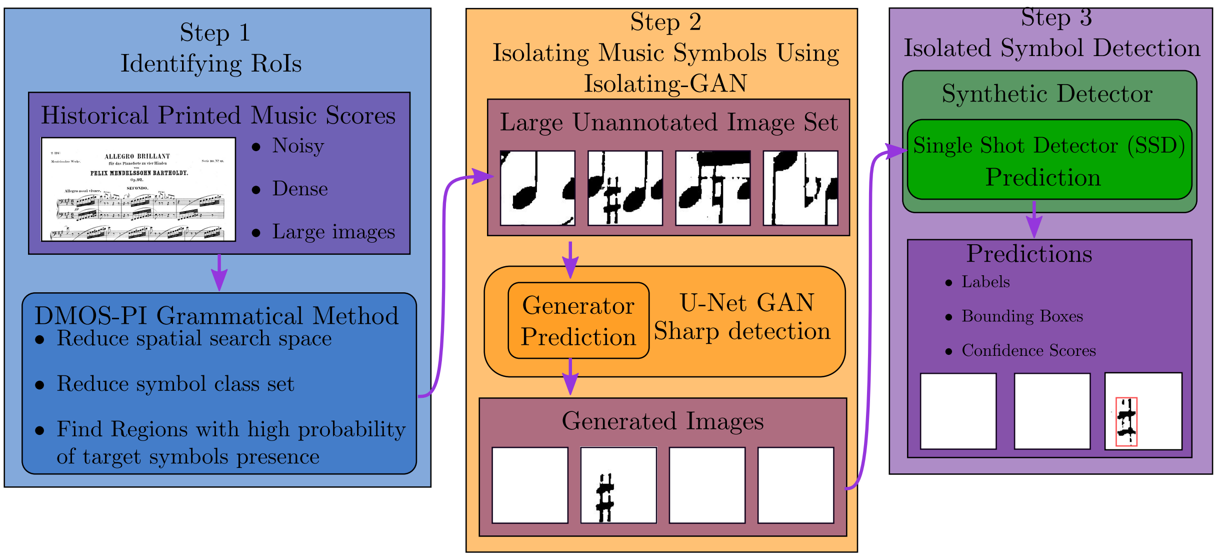

Step 1
Identifying RoIs
Historical Printed Music Scores
• Noisy
• Dense
• Large images
DMOS-PI Grammatical Method
• Reduce spatial search space
• Reduce symbol class set
• Find Regions with high probability of target symbols presence
Step 2
Isolating Music Symbols Using Isolating-GAN
Large Unannotated Image Set
Generator Prediction
U-Net GAN Sharp detection
Generated Images
Step 3
Isolated Symbol Detection
Synthetic Detector
Single Shot Detector (SSD) Prediction
Predictions
• Labels
• Bounding Boxes
• Confidence Scores

Figure 4.2 – Detailed overview of our unsupervised Isolating-GAN method for music symbol detection presented with three cascading steps: 1. using the DMOS grammar for identifying RoIs, 2. transferring small image patches of real music scores into a simpler graphical representation using a U-Net GAN, 3. Detection of isolated music symbol using pretrained SSD model.

introduced in section 1.4, can be used to specify a music grammar, able to parse music scores images.

This grammar takes as input graphical terminals such as line-segments and previously detected symbols and is able to reconstruct the music notation structure of a score. One of the strength of this method is the ability of the grammar to be used despite missing terminals with minimal alteration to the grammar itself. When insufficient information is provided to be able to construct the correct music notation structure, the grammar can be made to bypass those regions and continue the processing of the rest of the document. Moreover, zones where not enough information were provided can also be recorded, together with their position and contextual information in the document, so that additional detection process can take place. This process, as shown in fig. 4.3, can be used to reduce both the spatial search space of the music symbol detection task and the class set of symbols to detect.

We use this mechanism to our advantage to bootstrap our detection task for accidental symbols. We first reuse the capability of the grammar to build music notation structures such as systems, staves and bars using only a priori information about staff lines and vertical lines. Once note heads have been detected using a simple black blob

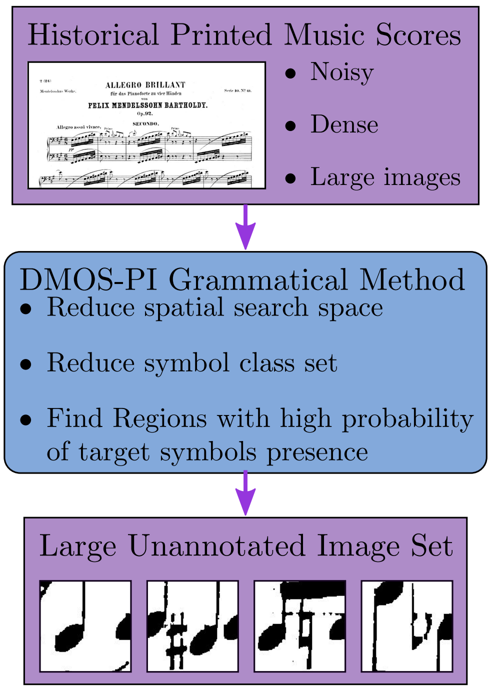

Historical Printed Music Scores
• Noisy
• Dense
• Large images
DMOS-PI Grammatical Method
• Reduce spatial search space
• Reduce symbol class set
• Find Regions with high probability of target symbols presence
Large Unannotated Image Set

Figure 4.3 – Grammatical processing of whole pages of historical music scores to find Regions of Interests with high probability of finding target symbols.

detector, the grammar rule reconstructing music notes can now successfully build a music note from a vertical bar, the stem, and one or multiple note heads. Once a note structure has been built, we can contextually identify RoIs focused at the left of each note heads and search for additional symbols such as accidentals. Moreover, we can artificially augment this dataset by reusing our bootstrapping method previously explained in section 2.2.1. After detecting accidentals in those RoIs, we can apply the grammar a second time using both previously recognized notes and newly detected accidental symbols to build a more complete note structure containing optional accidental symbols. More generally, this procedure could be repeated any number of time to build up RoIs for other kind of music symbols like attack signs, dynamics or ornaments based on the position of previously recognized symbols.

Using the DMOS grammar, we are therefore able to reduce the spatial search space for the music symbol detection task. While the search space is considerably reduced compared to the evaluation of the whole music score page, zones that do not contain symbols that we want to detect will always be present in the selected set of RoIs. Since we do not know how the amount of empty RoIs will impact the stability of our method, we propose a set of experiments in appendix A, section A.1.1 that evaluate the robustness of our method in regard to the ratio of empty RoIs and RoIs containing symbols to detect. With this method, we propose to experiment using a conservative approach on the size of the input image presented to the GAN due to the well-known unstable nature of the GAN architecture. We start with small image patches that guaranties us that the GAN training will manage to converge to competitive results and leave the task of increasing the input image size to future work.

The primary goal of this method is to design a method for resolving the music symbol detection task without using any kind of manual annotations for symbol detection and only using an isolated symbol dataset. While this is true for the training of both the GAN model and detection model, we found that it is necessary to constitute a very small validation detection dataset containing manual annotations of both labels and boxes of a symbol in the context of real historical music scores. The need for this small validation set is due to the unstable nature of the GAN, producing wildly different results when using the same data and model architecture. This small validation set is presented in section 4.5.3.1 where we also explain our strategy to select the best trained model using an early stopping mechanism.

Following this first step in our data processing pipeline, we now present the use of an isolated music symbol dataset in order to design an intermediate graphical representation allowing for trivial symbol detection. We will also present the final step of our pipeline which is isolated symbol detection trained using an existing isolated symbol dataset and synthetically generated images of random size and position of isolated symbols in white background. We first present this detection task because of its simplicity and the fact that we will use the resulting trained detector in our GAN training experiment to evaluate the generation quality of the GAN in regard to the end goal of unsupervised symbol detection.

4.4 Step 3: Isolated Music Symbol Detection

The task of supervised music symbol detection or more generally supervised object detection requires data demonstrating the use of the symbol in context, e.g. the use of music symbols in the context of a music score and manually added annotations such as the bounding boxes and class labels of symbols. To remove the need of such slow and costly manual annotations, such dataset can also be synthesized automatically as explained in section 3.2 by using a lot of expert knowledge of the specific application domain. However, as discussed previously in section 3.2.1, this process is very complex for music scores and the resulting detection model has no guaranties to work on real images of music scores. The basic requirement of any synthetic generation methods of document images is the use of isolated symbol image dataset and it is the process of spatially arranging such symbols in a blank page of a document that we name “Synthetic Generation”. As we explained before in section 3.3, our approach aims to transform this approach to unsupervised symbol detection by automatically translating real images of documents into a simpler representation (isolated symbols on white background) and therefore reducing the need for expert knowledge. We now explain how we design this simpler graphical representation by first presenting the isolated music symbol dataset. Since the use of this simpler graphical representation is the pivotal idea that links our second and third steps of our method, we first present the simpler third step of our method: isolating symbol detection in section 4.4.3 that we will reuse in our second step presented in section 4.5.

4.4.1 Isolated Music Symbol Dataset

An isolated symbol dataset is often used for classification tasks and is constituted of images of symbols together with their corresponding class label. This kind of datasets already gather multiple types of information, such as the shape of a class of symbol and the correspondence between a shape and a label. Multiple examples of the same symbol class are included in order to characterize the variability of a shape corresponding to a single class.

One of the premises of our work is that we believe that isolated music symbol dataset are much more common and easily available than detection dataset. For example, the OMR-Datasets website [Pacha] count at the time of this writing 11 different symbol classification datasets (also called isolated symbol datasets throughout this work) but only three different symbol detection datasets. Although this difference will be reduced as the field mature and more researcher share their manually constituted datasets, the lack of annotated dataset is a recurring problem for any new application domains trying to use supervised detection model.

For this work, we needed an isolated symbol dataset that would strongly resemble symbols from historical music scores of the mid-18th to the mid-20th century. The isolated symbol dataset we use for accidental detection, as shown in fig. 4.4, is an isolated symbol dataset manually constituted by the Intuidoc Team at the IRISA laboratory, where symbols were extracted from real historical printed music scores and classified automatically and verified manually. This dataset, publicly available here [1], contains 541 symbol images with 3 different accidental classes and the following table 4.1 show in details the detailed distribution of symbols.

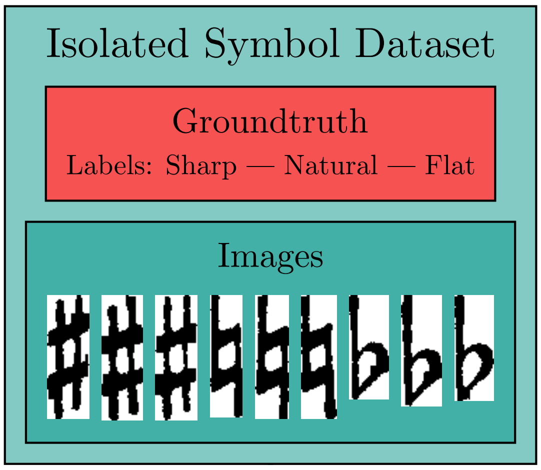


Figure 4.4 – Isolated music symbol dataset, used for music symbol classification task.

Natural	Flat	Sharp	Total
206	144	191	541

Table 4.1 – Isolated Symbol Dataset class distribution.

As explain in our next section, this isolated symbol dataset is used to design a target representation that the GAN model will be trained to generate. Therefore, contrary to the task of classification, the isolated symbol dataset priority is not about the number or diversity of the samples, but samples of high quality and intra-class shape variability that accurately model the intra-class shape variability of the real dataset. That is why we did not consider using the Rebelo dataset [Rebelo 09] because of the low resolution of symbols. Using this isolated music symbol dataset, we are now able to design an intermediate graphical representation to be use a target representation for the GAN model.

4.4.2 Synthetic Detection Dataset Generation

Using this isolated music symbol dataset, we can now construct a detection task in the simplest manner possible. In a blank canvas, we paste isolated music symbols at a random position after randomly resizing the size of the symbols. We illustrate this generation in fig. 4.5. With this method, we can avoid almost all the need for expert

1. https://www-intuidoc.irisa.fr/en/choi_accidentals/

knowledge about music notation which makes this step easier to do and easier to port to a new application field. Although this step seems like a trivial image generation step, it is the pivotal phase of the method that bridges the transformation step done by the Isolating-GAN presented in section 4.5 and the detection step using a SSD that we show next in section 4.4.3. By generating this artificial detection dataset, we design a simpler graphical representation that the Isolating-GAN model can learn to generate while allowing for an accurate detection of the real symbols in generated images.

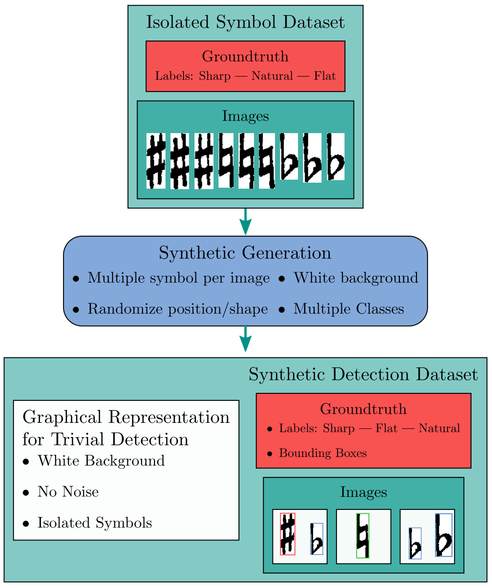

Isolated Symbol Dataset
Groundtruth
Labels: Sharp — Natural — Flat
Images
Synthetic Generation
• Multiple symbol per image • White background
• Randomize position/shape • Multiple Classes
Synthetic Detection Dataset
Graphical Representation for Trivial Detection
• White Background
• No Noise
• Isolated Symbols
Groundtruth
• Labels: Sharp — Flat — Natural
• Bounding Boxes
Images

Figure 4.5 – Synthetic detection dataset generation using only isolated music symbols on white background. Generated images synthesizes a graphical representation that allows trivial detection while keeping interesting characteristics from real images of music scores like the position and shape of symbol to detect.

A number of hyperparameters needs to be defined in order to accurately train this generation method. A set of classes should be chosen as target classes and defines which symbols we want to detect in the original images. The remaining hyperparameters, such as the number of symbols, their size and shape, can only be chosen by using expert knowledge of the specific application domain. In OMR, we can parameterize the size of symbols using the vertical distance between two staff lines, i.e. the interline, as a base distance unit. For example, we can specify that the width of a sharp symbol should approximately be one times the interline and the height 3 times the interline. However, this specific knowledge for a sharp and other symbols can vary in printed scores depending on the typeset used for imprinting.

In section 4.5.1, we explain in more details how this generation step relates to our use of the Isolating-GAN and how we are able to isolate music symbols in real images using this simpler graphical representation. Next, we define how we perform the final detection step using the small SSD model which we will then use to evaluate the Isolating-GAN training.

4.4.3 Isolated Symbols Detection using Single-Shot MultiBox Detector

Since the design of the intermediate representation makes the detection trivial because images present isolated music symbols in a white background, we chose to use a Single-Shot MultiBox Detector model introduced by Liu et al. [Liu 16]. This small and lite detector model is ideal for our use-case where we need a model with a very fast inference time for intensive evaluation as used in section 4.5.3.1. Since our task is so simple, we propose to use both a reduced size of the original SSD model using only the 7 first layers of the original VGG-16 backbone network. We use this reduced detector for our development experiments shown in section 5.2 for fast training evaluation and development iterations. For our final evaluation of our method, we use a regular SSD model using the full VGG-16 backbone network. This bigger detector, although slower than our reduced version, produces much more accurate detections. The training of our two SSD model is illustrated in fig. 4.6 and is done using our synthetic detection dataset previously presented in section 4.4.2.

When we apply our large detector on image patches of real music scores containing both music symbols and background noises, we show in section 5.3.2 that the detector is able to correctly detect existing real music symbols. However, we can not directly apply this trained detector on the real image patches of historical music scores documents because of the large amount of False Positives generated due to the noisy background, which is to be expected given that our data does not model background noises and symbols. To reduce the amount of False Positives, we propose the Isolating-GAN method as a transfer function from real data containing noisy background information to synthetic data with clean isolated symbols on white background.

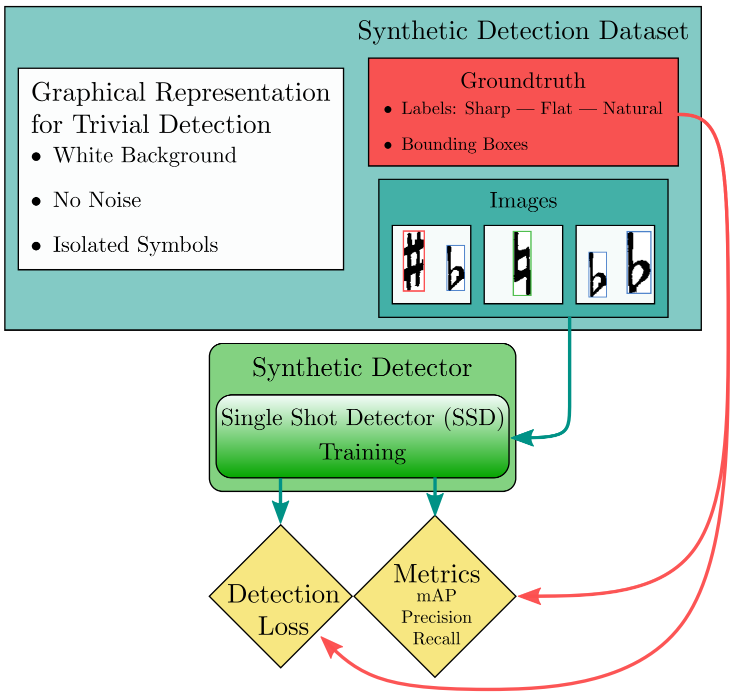

Synthetic Detection Dataset
Graphical Representation for Trivial Detection
• White Background
• No Noise
• Isolated Symbols
Groundtruth
• Labels: Sharp — Flat — Natural
• Bounding Boxes
Images
Synthetic Detector
Single Shot Detector (SSD) Training
Detection Loss
Metrics
mAP
Precision
Recall

Figure 4.6 – Training of a Single-Shot Detection (SSD) model with a synthetic detection dataset of isolated symbols on white background generated as shown previously in fig. 4.5.

4.5 Step 2: Isolating-GAN using Image-to-Image Translation

In section 4.3, we presented the input images extracted from real music scores in which we have to isolate music symbols and in section 4.4.2, we specified the graphical representation on which we want to run the detection task. We now need to design a generative model able to transform our input to our output graphical representation which correspond to the second step of our method illustrated in fig. 4.2. As previously explained in section 3.3.2, the literature has shown that Generative Adversarial Network models have the required characteristics for this difficult problem. We now present in details how we take advantages of the peculiar GAN generative properties in regard to our music symbol isolation tasks.

4.5.1 Learning the Intermediate Representation as Isolated Music Symbol

The introduction of the GAN architecture starting from Goodfellow et al. [Goodfellow 14] made possible the generation of completely artificial data like handwritten digits using an adversarial methodology. One interesting characteristics of this method is that the generation of artificial data is conditioned mainly by another target dataset, the MNIST dataset [LeCun] in this case. A following work [Chu 17] has further shown that a modified GAN architecture called Cycle-GAN can transform a set of images belonging to a specific domain, for example: images of horses, into another representation domain, like images of zebras, and vice-versa. The main novelty of this work is to be able to do **unpaired** image-to-image translation without any paired image information that explicitly link the representation of an object in multiple domains of representation. Specifically, the author of [Chu 17] manage to train a model able to transform a horse into a zebra and vice-versa sharing the same pose and position in the image, without explicitly presenting a paired instance of a horse and a zebra.

The main proposition of this work is to take advantage of this peculiar capacity of GAN models to implicitly learn how to translate an instance of an object from one domain into another. In the case of music symbol detection, our aim is to transform an input image coming from a real historical music scores, containing noise, degradation and complex background symbols and signals into a simple clean output image free of any background signals. However, this output image should keep the shape and exact position and size of symbols to detect so that a detection performed on the output image can be automatically transferred onto the input image. In other words, the task of the GAN is threefold:

1. Keep invariant the position and size of real symbols to detect
2. Modify the shape of real symbols to detect to resemble isolated symbols of the corresponding class
   — Also enhance and repair damaged symbols to resemble isolated symbols
3. Remove background symbols and noise

In order for the GAN model to learn this task, we show in section 4.4.3 how we design an intermediate graphical representation using only isolated music symbols that retain the three main characteristics of the GAN task: music symbol shape information, music symbol position and size information, an empty white background. Using isolated music symbols corresponding to the target symbol class that we want to detect, our objective is for the GAN to learn how to translate real symbols to detect in real music scores into the corresponding isolated music symbol shape. By pasting isolated music symbols into a blank canvas, we introduce the notion of symbol position and size relative to the image patch size.

As we mentioned before in section 4.4.3, symbols present in real images of music scores present an intra-class variability in their size and shapes. This variability can have a huge impact on the performance of the GAN training because the sizes of isolated symbols should be representative of the sizes of symbols in the real images of music scores so that the GAN can learn to transfer symbols to detect with an identical size. This is why we introduce randomized variability into the generated sizes of isolated symbols by using a minimal and maximal height and width parameters. Width and height for each symbol are then sampled using a uniform distribution between these two extrema.

In the case of the classical adversarial training, the GAN is not constrained into respecting the original shape and position of the symbol to detect. This lack of constraint led to very unstable training and poor results. In order to improve the stability of the training, we introduce an additional reconstruction loss using only isolated symbol data described in section 4.5.2.3. By using a white empty background, we also avoid the difficult generation process of trying to synthesize accurately historical music score background images, and we force the generator of the GAN to learn to act as a filter for background symbols and noise of the input real image. We further help the GAN training to model this rejection task by using positive and negative examples of isolated music symbols. While positive examples are symbols we want the GAN to isolate and keep into the generated image, we add negative examples of isolated symbols belonging to symbol classes we do not want to detect. The annotation of symbols being positive or negative examples is actually done in the GAN loss functions as described in section 4.5.2.2.

We now present the architecture of our Isolating-GAN together with the various enhancements to the training methodology in order to further stabilize the process and improve the detection results.

4.5.2 Architecture

The base architecture for our GAN model is the Deep Convolutional Generative Adversarial Network (DCGAN) proposed by Radford et al. [Radford 16]. This architecture shown in fig. 4.7 is an extension to the original GAN architecture of [Goodfellow 14] using convolutional and pooling layer instead of the original dense layers. The use of convolutional layers greatly improve the quality of the features learned during the adversarial training and directly impact the generation and discrimination quality of the model. We replace the generator part of the DCGAN model with the U-Net semantic segmentation model proposed by Ronneberger et al. [Ronneberger 15] to be able to use images as input data to the generator model. The U-Net model architecture is similar to an encoder-decoder architecture, where the encoder is a traditional convolutional/pooling feature extractor. The decoder part is a mirrored version of the encoder where the upscaling is done using transposed 2D convolutions. Features extracted by the encoder are also concatenated to the corresponding level of generated features in order to improve the generation quality. We choose to use the VGG16 model proposed by Simonyan et al. [Simonyan 15] as a backbone network for the U-Net generator which allows us to reuse pretrained VGG16 weights on the ImageNet dataset, see Deng et al. [Deng 09], for the encoder part of the U-Net model. A detailed description of the network generator is shown in table 4.2 and the network discriminator in table 4.3.

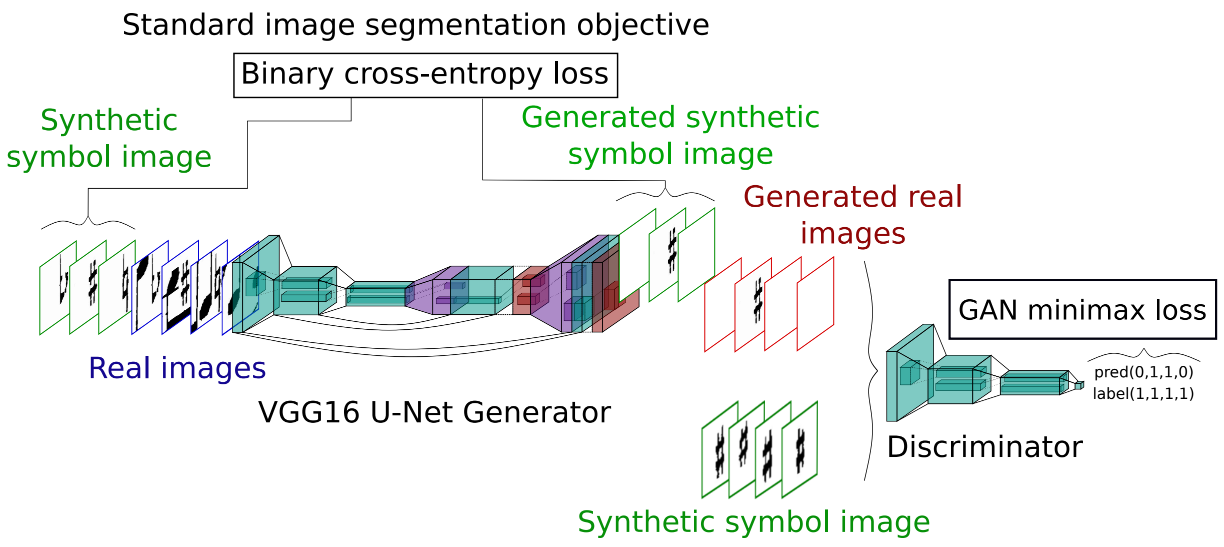


Figure 4.7 – Isolating-GAN architectural overview for unsupervised music symbol detection. Main difference from traditional GAN architecture [Goodfellow 14] (see fig. 1.10) is: 1. the VGG U-Net generator, 2. the additional binary cross-entropy loss.

Table 4.2 – Isolating-GAN **generator** using a VGG16 as a backbone detailed overview with layers and number of parameters per layer. The total number of parameters is 25,854,657. Convolutional layer sizes are given with (Height, Width, Channels).

Layer	Type	Output Shape	Parameters Number
block1 conv1	Conv2D	(128, 128, 64)	1792
block1 conv2	Conv2D	(128, 128, 64)	36928
block2 conv1	Conv2D	(64, 64, 128)	73856
block2 conv2	Conv2D	(64, 64, 128)	147584
block3 conv1	Conv2D	(32, 32, 256)	295168
block3 conv2	Conv2D	(32, 32, 256)	590080
block3 conv3	Conv2D	(32, 32, 256)	590080
block4 conv1	Conv2D	(16, 16, 512)	1180160
block4 conv2	Conv2D	(16, 16, 512)	2359808
block4 conv3	Conv2D	(16, 16, 512)	2359808
block5 conv1	Conv2D	(8, 8, 512)	2359808
block5 conv2	Conv2D	(8, 8, 512)	2359808
block5 conv3	Conv2D	(8, 8, 512)	2359808
conv trans1	Conv2DTranspose	(16, 16, 512)	1049088
conv5	Conv2D	(16, 16, 512)	4719104
conv6	Conv2D	(16, 16, 512)	2359808
conv trans2	Conv2DTranspose	(32, 32, 256)	524544
conv7	Conv2D	(32, 32, 256)	1179904
conv8	Conv2D	(32, 32, 256)	590080
conv trans3	Conv2DTranspose	(64, 64, 128)	131200
conv9	Conv2D	(64, 64, 128)	295040
conv10	Conv2D	(64, 64, 128)	147584
conv trans4	Conv2DTranspose	(128, 128, 64)	32832
conv11	Conv2D	(128, 128, 64)	73792
conv12	Conv2D	(128, 128, 64)	36928
conv13	Conv2D	(128, 128, 1)	65

Table 4.3 – Isolating-GAN **discriminator** using a VGG16 as a backbone detailed overview with layers and number of parameters per layer. The total number of parameters is 1,142,017. Convolutional layer sizes are given with (Height, Width, Channels).

Layer	Type	Output Shape	Parameters Number
conv 1	Conv2D	(64, 64, 32)	832
conv 2	Conv2D	(32, 32, 64)	51264
conv 3	Conv2D	(16, 16, 128)	204928
conv 4	Conv2D	(16, 16, 256)	819456
dense 1	Dense	(1)	65537

4.5.2.1 Isolating-GAN Training Architecture

In order to train the Isolating-GAN architecture, we build on the original training algorithm presented by Goodfellow et al. [Goodfellow 14] using both the discriminator loss and adversarial loss to train respectively the discriminator and the generator.

Discriminator Training with Real Dataset To drive the detection of a specific symbol, we reuse the synthetic detection dataset presented in section 4.4.2 where we generate images containing isolated music symbols on white background. In the particular case of the discriminator loss illustrated in fig. 4.8, we also need to specify for each image generated a positive or a negative label. In this case, the labels marks the origin of the image where a positive labels marks automatically generated images using isolated symbols and negative labels marks images generated by the Isolating-GAN generator. However, this labeling mechanism can also be used to improve the rejection capability of the Isolating-GAN as explained in section 4.5.2.2.

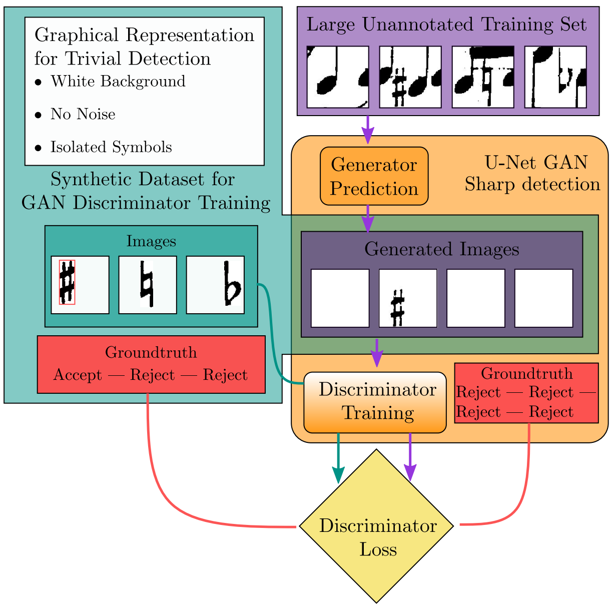

Graphical Representation for Trivial Detection
• White Background
• No Noise
• Isolated Symbols
Synthetic Dataset for GAN Discriminator Training
Images
Groundtruth
Accept — Reject — Reject
Large Unannotated Training Set
Generator Prediction
U-Net GAN Sharp detection
Generated Images
Discriminator Training
Groundtruth
Reject — Reject — Reject — Reject
Discriminator Loss

Figure 4.8 – Isolating-GAN discriminator training using isolated symbols negative examples. Here, the discriminator is trained to differentiate images based on their origin. Synthetic images containing sharps are accepted while real images of music score transformed by the generator are rejected. Additionally, we also add negative synthetic examples of symbol class we do not want to detect such as naturals and flats which we also train the discriminator to reject.

Adversarial Training with Real Dataset The adversarial loss shown in fig. 4.9 is used to train the U-Net generator of the GAN. As usual of the adversarial training algorithm, we only specify positive labels for the adversarial loss in order to push the generator to only keep symbols of the relevant class while removing everything else.

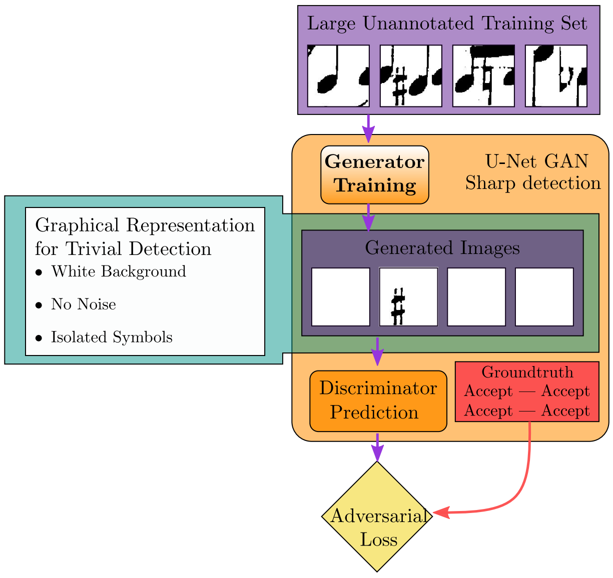


Figure 4.9 – Classical GAN generator training. Here, the generator is trained using only real images of music scores. The goal of the generator is to fool the discriminator to accept the transformed image of a real music music score. Fooling the discriminator is done by generating blank images containing only isolated sharp symbols. The easiest way to generate such images is for the generator to isolate existing sharps and removing everything else.

The adversarial training of the Isolating-GAN is done in two distinct steps in order to optimize in turns the generator and the discriminator. Therefore, we now propose the different improvements over the classical GAN training strategy successively for the discriminator, generator and how to balance the training of the two models.

4.5.2.2 Discriminator Training

The discriminator training illustrated in fig. 4.8 is done by using real images of music scores transformed by the U-Net generator and isolated symbols detection images. Both group of images are processed by the discriminator for binary classification which discriminate the origin of the images, either coming from the U-Net Generator or coming from the isolated symbol detection dataset. The gradients are computed by using a binary cross entropy loss between the output of the discriminator and artificially generated labels that marks the true origins of each image. The gradients are then backpropagated and applied in the discriminator only.

Training Data Composition While the composition of the real input images to the generator is randomized over the whole dataset, we have complete control over of the distribution of the isolated symbol detection dataset. We found in our experiments that the constraints in terms of possible minimum and maximum sizes that a specific symbol class can take is crucial to the performance and stability of the Isolating-GAN training. Unfortunately, these constraints parameters can not be deduced automatically from the real dataset since no bounding boxes ground truth exist, but we can derive approximate parameters using domain specific knowledge as explained previously in section 4.4.2. These parameters can then be fined tune by grid-search exploration using our minimal validation set.

Another important parameters of the isolated symbol detection dataset generation process is the choice of symbol classes to use. Originally, we only used classes of symbols we wanted to our U-Net generator to detect, i.e. positive examples. However, isolated symbol dataset usually contains a large set of symbol classes that covers the entire application domains. For example, our own isolated symbol dataset used in this works contains 26 different classes of music symbols. Instead of trying to use every class of symbol possible, we used a more iterative approach where we first observed the most common mistake the Isolating-GAN was doing. For example, preliminary results for our accidental detection task showed a clear behavior of the U-Net generator to confuse the different accidental classes. From this observation, we made an experiment using other accidental classes as negative examples except the accidental class to detect. We illustrate this in fig. 4.8 where isolated sharps are marked as positive examples and isolated flat and natural symbols are marked as negative examples. This mechanism can be used to explicitly forbid the generator to produce certain shapes and help the generator to avoid confusing symbols with different classes but similar shapes.

Loss Function Definition Formally, we name our source domain X with image samples $\{x^i\}_{i=1}^{M}$ the domain of real music scores image patches. For the discriminator loss, images from the source domain are always labeled as negative examples, i.e. $X_{label} = 0$. In the other hand, we name our target domain $Y = Y_{pos} \cup Y_{neg}$ with image samples $\{y^j_{pos}\}^N_{j=1} \in Y_{pos}$ containing isolated music symbols of classes we want to detect and image samples $\{y^k_{neg}\}^O_{k=1} \in Y_{neg}$ containing isolated music symbols of classes we do not want to detect. We label our target domain images as follows:

$$Y_{label}(\{y^l_{label}\}^P_{l=1}) = \begin{cases} 1, & \text{if } y^l_{label} \in Y_{pos} \\ 0, & \text{if } y^l_{label} \in Y_{neg} \end{cases} \tag{4.1}$$

Our U-Net generator G goal is to map images from the source domain to the target domain while our discriminator D goal is to discriminate images based on their origin: Y_{pos} or $Y_{neg} \cup G(X)$. Using the binary cross-entropy loss, we define the usual discriminator loss:

$$\begin{aligned} \mathcal{L}_D(G, D, X, Y, Y_{label}) = & \mathbb{E}_{(y, y_{label}) \sim (Y, Y_{label})} \overbrace{[y_{label} * \log(D(y)) + (1 - y_{label}) * \log(1 - D(y))]}^{\text{binary cross entropy loss}} \\ & + \mathbb{E}_{x \sim X} \underbrace{[\log(1 - D(G(x)))]}_{\text{simplified binary cross entropy since } x_{label}=0} \end{aligned} \tag{4.2}$$

4.5.2.3 Generator Training

Adversarial Training with Real Dataset The training of the generator illustrated in fig. 4.9 is done using the original adversarial loss, by doing a forward pass using image patches of real music scores through the U-Net generator and the discriminator. Gradients are computed at the end of the discriminator using a binary cross entropy loss between the output of the discriminator and synthetically generated positive labels. Gradients are then backpropagated through the discriminator and the generator but only applied to the generator. By using only positive labels, we train the generator to fool the discriminator by generating what the discriminator consider as positive labels, i.e. isolated music symbol of the class we want to detect.

Reconstruction Training with Isolated Symbols One of the problem with the original GAN training losses, in respect to our detection task, is the lack of constraints to keep the original symbol shape and position in the generated image. As described in the original GAN work, the generator can easily collapse into systematically generating the same symbol at the same position while perfectly fooling the discriminator. However, it is not possible to add additional information about symbol sizes and positions using the generator input image patches of real music scores since this dataset is not annotated.

In another hand, we actually have total control and complete information for our isolated music symbol dataset. We propose to reuse this knowledge to our advantage by adding a supplementary loss called **reconstruction loss** and apply it to the isolated music symbol dataset. The main goal of this new training objective is to explicitly train the generator to correctly process the specific case of isolated music symbols pasted at random position and size in a white canvas. This loss is applied only to the generator in the style of classic auto-encoder network, as shown in fig. 4.10, where input data are images to process and ground truth are respectively images that the generator should produce.

In our case, we use symbols from our isolated symbol dataset and generate both input images and output images using the same method as described in section 4.4.2 with a slight variation. In the case where the input image contains an isolated symbol of the class we want to detect, we use the exact same image as ground truth. For cases where the input image contains an isolated symbol of the class we do not want to detect, we replace the isolated symbol with white pixels. This method allows to both provide an explicit training objective for the ideal input case of the generator (isolated symbol with empty background) for which the generator should learn the identity transform and a few rejection cases using negative examples of symbols that we do not want to detect for which the generator should learn to suppress. This additional information during the training process allows us to stabilize even further the training of the Isolating-GAN while also improving the detection performance.

Formally, we now recall the usual adversarial loss which uses our real music score image patches X as a source domain:

$$\mathcal{L}_{Adv}(G, D, X) = \mathbb{E}_{x \sim X}[\log(1 - D(G(x)))] \tag{4.3}$$

And our additional reconstruction loss which is a binary cross-entropy loss applied between our target domain images Y and target domain ground truth images Y^*:

$$\mathcal{L}_{Recons}(G, Y, Y^*) = \mathbb{E}_{(y,y^*) \sim (Y,Y^*)}[y^* * \log(G(y)) + (1 - y^*) * \log(1 - G(y))] \tag{4.4}$$

Where Y^* is defined as we explained previously, either y the input image for positive

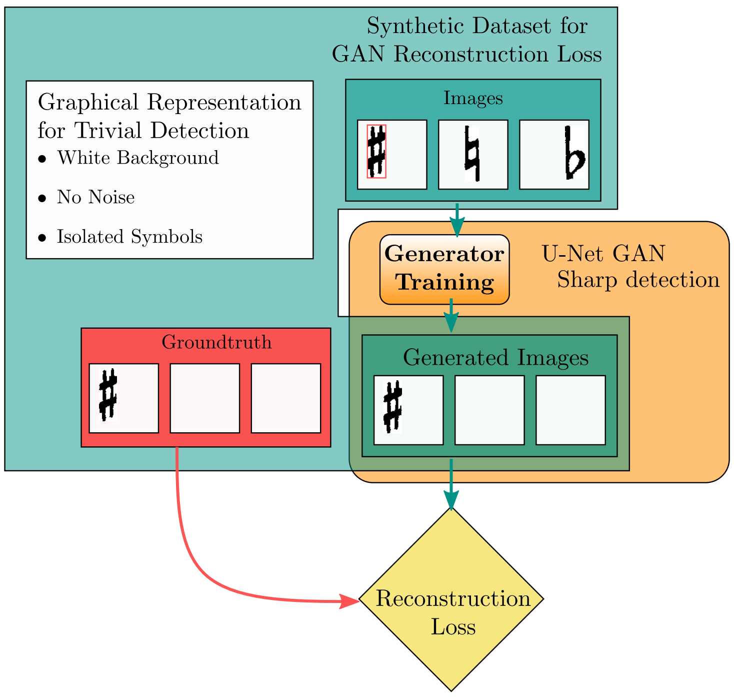


Figure 4.10 – Isolating-GAN reconstruction loss using synthetic data. Here, we additionally train the generator to learn the identity transform of isolated symbols we want to detect and the removal of isolated symbols we do not want to detect. This training can be simply realized using our synthetic detection dataset together with a binary cross-entropy loss as a reconstruction loss for the generator.

examples and y_{blank} an empty white image for negative examples:

$$Y^*(\{y^{*l}\}_{l=1}^{P}) = \begin{cases} y, & \text{if } y^l_{label} \in Y_{pos} \\ y_{blank}, & \text{if } y^l_{label} \in Y_{neg} \end{cases} \tag{4.5}$$

4.5.2.4 Tuning Balance Between Different Losses

We can now define our global training objective by joining our discriminator loss (eq. (4.2)), generator loss (eq. (4.3)) and reconstruction loss (eq. (4.4)) as a two-player minimax game where we aim to solve:

$$\begin{aligned} \min_G \max_D V(D,G) = \; & \mathcal{L}_D + \mathcal{L}_{Adv} + \mathcal{L}_{Recons} \\ = \; & \mathbb{E}_{(y,y_{label})\sim(Y,Y_{label})}[y_{label} * \log(D(y)) + (1-y_{label}) * \log(1-D(y))] \\ & + \mathbb{E}_{x\sim X}[\log(1-D(G(x)))] \\ & + \mathbb{E}_{(y,y^*)\sim(Y,Y^*)}[y^* * \log(G(y)) + (1-y^*) * \log(1-G(y))] \end{aligned} \tag{4.6}$$

The original GAN paper [Goodfellow 14] mention that in order to avoid a collapse of the generator, the discriminator should be synchronized to the generator training, i.e. the generator should not be trained too much. In the original paper, a single parameter *K* is used to balance the adversarial and discriminator loss. This parameter controls the number of batch-wise training iterations the discriminator should do before retraining the generator using the adversarial loss.

In this work, we apply this concept to our training algorithm where we parameterize the number of consecutive batch-wise training step for our three discriminator, adversarial and reconstruction losses, which we name respectively `DiscNumTrain`, `AdvNumTrain` and `ReconsNumTrain`. We also parameterize the number of time a loss can be trained before retraining another loss, such as `GenMaxNumTrain` which is the maximum number of training step allowed for the generator using the adversarial or reconstruction loss before retraining the discriminator. Finally, we limit the use of the reconstruction loss by using an additional parameter `ReconsTrainFreq` which specify the frequency at which the reconstruction loss is used. We present our whole training algorithm given in Algorithm 1 which show how we use our data and parameters for training the Isolating-GAN model.

A simpler implementation of this loss would have been to merge the adversarial and reconstruction loss as a single loss using a weight parameter. The reason for our

Algorithm 1: Minibatch training of Isolating-GAN.

```
GenNumTrainCpt ← $0$
ReconsTrainCpt ← $0$
for number of batches in $X$ do
    Sample minibatch $x$ from $X$
    for AdvNumTrain() steps do
        if GenNumTrainCpt() multiple of GenMaxNumTrain then
DUpdate     Sample minibatch of images $y$ and labels $y_{label}$ from $(Y, Y_{label})$
            for DiscNumTrain() steps do
                Update the discriminator with discriminator loss:
                $$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [y^i_{label} * \log(D(y^i)) + (1 - y^i_{label}) * \log(1 - D(y^i)) + \log(1 - D(G(x^i)))]$$
        GenNumTrainCpt ← GenNumTrainCpt $+ 1$
        Update the generator with adversarial loss:
        $$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(1 - D(G(x^i)))$$
    if ReconsTrainCpt() multiple of ReconsTrainFreq then
        Sample minibatch $(y_{recons}, y^*_{recons})$ from $(Y, Y^*)$
        for ReconsNumTrain() steps do
            if GenNumTrainCpt() multiple of GenMaxNumTrain then
                Update discriminator, see DUpdate.
            GenNumTrainCpt ← GenNumTrainCpt $+ 1$
            Update the generator with reconstruction loss:
            $$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log({y^*_{recons}}^i - D(G({y_{recons}}^i)))$$
        ReconsTrainCpt ← ReconsTrainCpt $+ 1$
```

implementation is the organic growth of the model where we gradually added parameters and improvements in order to improve our results. Simplifying this algorithm is one of the first planned improvement for our future works.

4.5.3 Evaluation Protocol

We now present our evaluation methodology of the Isolating-GAN model in relation to our music symbol detection task.

4.5.3.1 Evaluation architecture: U-Net Generator + SSD

Contrary to more classical model in Deep Learning literature, the evaluation task of a GAN model is not a straightforward task because it is difficult to evaluate directly the generation quality of the GAN generator as explained by Borji [Borji 19]. However, in our case, the images produced by the Isolating-GAN generator are not our end goal but just an intermediary representation for the following detection task.

That is why we propose to evaluate our Isolating-GAN model using a proxy detector model. For that, we reuse our isolated symbol detector model we shown in section 4.4.3. Our evaluation architecture is made by connecting the U-Net generator of the GAN model with the SSD model in order to create an end-to-end detection task going from real image patches of music scores all the way to producing detection predictions with boxes and labels. We illustrate this architecture in fig. 4.11.

4.5.3.2 Training Results Analysis

Next, we need to have annotated detection ground truth in order to evaluate our detection task. However, this need contradicts our approach which is to avoid as much as possible to use manually annotated data since it is costly and slow to produce. Ideally, we could train our Isolating-GAN model without using any ground truth information and directly use it to produce reliable detection prediction. However, we found that it is critical for the Isolating-GAN model to be tuned correctly using the right hyperparameters in order to stabilize the training process and improve the detection accuracy. Instead of trying to optimize the hyperparameters blindly, we propose to use a very small validation dataset of 10 or 20 examples per symbol class which we will manually annotate. This manual annotations are produced by splitting images produced by our DMOS parser into

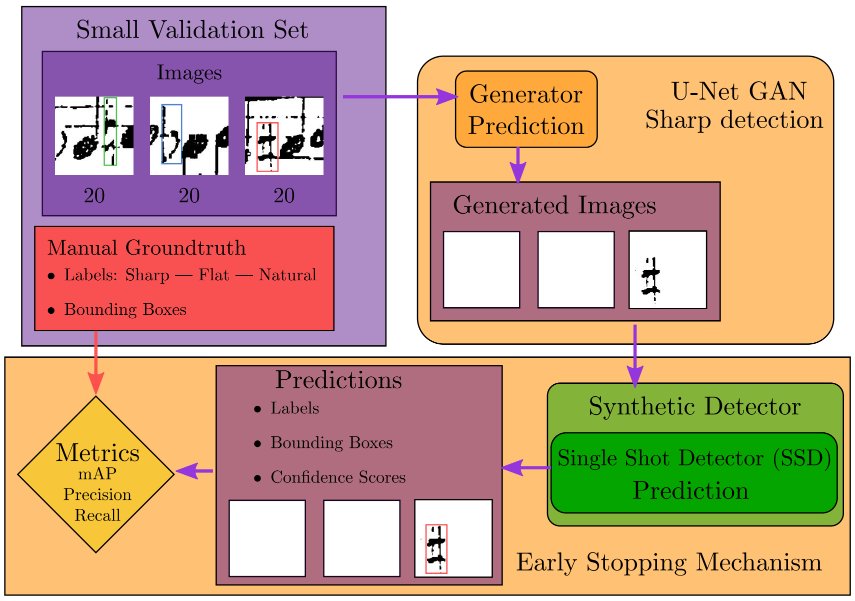


Figure 4.11 – Isolating-GAN evaluation architecture. By joining the trained generator as shown in section 4.5.2.1 and detector pretrained on our synthetic detection dataset, we create a detection model capable of localizing symbols in real historical music score images. The predictions of the SSD detector can then be evaluated using our manually constituted small validation dataset.

a training and validation set. The validation set is manually annotated while the training set is directly used for training the Isolating-GAN model without any manual annotations. We also use bootstrapping as presented in section 2.2.1 in order to artificially augment our validation set and improve our estimator. Finally, we use the mean Average Precision (mAP) detection metric as presented by Everingham et al. [Everingham 10] with an Intersection over Union (IoU) threshold of 0.75.

The goal of the validation set is to be able to give relatively accurate measure of the detection performance of the Isolating-GAN + SSD architecture while being as small as possible. Although it won't give an exact estimation of the detection performance of the method, we can at least use this estimation to fine tune our hyperparameters and choose the best performing Isolating-GAN out of multiple trainings.

GAN models are well-known for their training instability and the mode collapsing problem has been identified since the first introduction of the model [Goodfellow 14] and is explained by a saturated discriminator which is able to reject the generator examples so strongly that no gradients are produced for the generator training. In our work, we want to evaluate how well our Isolating-GAN is able to transform images from our source domain, patch images of real music scores, into our target domain, isolated music symbol of classes we want to detect, in relation to the detection that will be performed by our SSD model. But this evaluation can be difficult when we only have access to a small evaluation dataset and the training is blocked in a mode collapsed state. To improve our evaluation protocol, we decide to retrain the same Isolating-GAN model with the same hyperparameters set multiple times using a different random seed each time. This common practice in machine learning gives us some insight about the basic stability of the training method. Because of the mode collapsing problem, retraining our model multiple time is essential to have a smoother estimation of the performance of the model where we can count the number of mode collapse. Next, we explain how we use our small validation set to do an early stopping of the training.

4.5.3.3 Early Stopping

Due to the unstable nature of the GAN training, multiple training of the same model with the same hyperparameters but with different random seeds can actually produce significantly different results. While the results can vary quite a bit, the best model out of a set of trained model can performed satisfactorily. However, there is a need to identify this particular run and particular training epoch that gave the best result. This is why we use an early stopping mechanism together with a multi-run approach using a small validation set that we presented previously. The early stopping mechanism enables us to stop the Isolating-GAN training before overfitting happens. At every epoch, we compute the mAP metric on our small validation set and if this metric does not improve after a number of successive epochs, also called the patience parameter, we stop the training. The last epoch that gave the best results will then be used to produce detection prediction on the whole dataset.

4.6 Conclusion

We now have presented in details our novel method for unsupervised music symbol detection in historical printed music scores using only isolated music symbols. By using isolated symbols to create a simplified intermediate representation, we can use a three steps method to isolate music symbols in real music scores. The first step is to use a syntactical method fed with simple primitives and previous detection in order to identify RoIs that have a higher probability of containing the music symbols we want to detect. Secondly, we use our generative Isolating-GAN method to simplify the graphical representation of the RoIs and isolate target music symbols. While we build on the existing U-Net GAN model, we propose an original combination of an isolated symbols dataset and real historical music score dataset to train this model. Moreover, since the training of this step is particularly unstable, we enhance our GAN model training with an additional training objective. Our final step is the detection of music symbols previously isolated in the RoIs, using a simple detector previously trained on our synthetic detection dataset generated using an existing isolated music symbol dataset.

Using both the Isolating-GAN and a simple detector, we can produce detection predictions with which we will be able to evaluate our method. Since no ground truth is available to choose any training hyperparameters of the Isolating-GAN model, we devise a hyperparameter optimization strategy based on small validation dataset, manually produced, of few examples per symbol classes and augmented with bootstrapping. This small validation dataset is then used to compare the impact of different training hyperparameters and choose near optimal training hyperparameters values. Finally, since the same set of hyperparameters can produce trained generator models of large varying detection performance because of the GAN training instability, we can use this small validation set to elect the best performing Isolating-GAN model at the best performing epoch between a set of trained Isolating-GAN models using the same set of hyperparameters. However, this requires a constant monitoring of the Isolating-GAN model during training.

In the next chapter, we validate our method, first with architectural experiments in section 5.2 to show the impact of different improvements we made to the GAN architecture and training strategy. While we validate our architectural experiments on a small homogeneous dataset for easier and faster development iterations, we also validate the detection performance of our Isolating-GAN method in section 5.3 on a large heterogeneous dataset spanning a larger number of music scores of varying styles. In appendix A, in section A.1, we propose additional experiments that tests on one hand the impact of symbol distribution in our historical printed music scores RoIs and on the other hand the sizes range of isolated symbols used to generate our synthetic detection dataset.

CHAPTER 5

EXPERIMENTS

5.1 Introduction

Following the presentation of our new Isolating-GAN method in chapter 4 for unsupervised music symbol detection, we now present a large set of experiments using our small and homogeneous accidental dataset [Choi 18a], evaluating the different architectural improvements in section 5.2 as explained in section 4.5.2.Then, in section 5.3, we use a much larger and heterogeneous accidental dataset to demonstrate that our method can generalize to a much more challenging detection task.

5.2 Architectural Experiments

This section presents our experimental procedure to develop and validate our new strategy for symbol detection. The task we set out to do is the detection of accidental symbols in images produced by the first step of our method, explained in section 4.3. However, this task can be very challenging, especially in an unsupervised setting, because of the lack of accidental symbols to detect. From what we could gather after manually annotating accidental symbols, only 10 to 20% of images would contain symbols to detect and this number is probably overestimated because we chose to annotate music score pages with high concentration of accidental symbols. Moreover, this lack of symbols to detect seems to highly impact the stability of our method, the fewer the symbols are, the more instable our method gets.

To control the complexity of our task and the instability of our method, we used an iterative development strategy and started with a simple model, reduced dataset and a simple detection task. We then gradually improve and stabilize the model by modifying our loss function or by modifying the data used during training. Since our method is unsupervised, only the data available such as isolated symbols contains the knowledge we want the GAN to learn. Therefore, it is of paramount importance how we use that data to influence the learned knowledge and training stability of the GAN. An overview of the architectural experiment together with what we consider our contribution to the original GAN architecture is shown in fig. 5.1. Our main architectural contributions from the original is the use of an additional image reconstruction training objective able to stabilize the training of the GAN and improve the detection quality of our method.

The first goal of section 5.2.1 is to validate the capacity of the Isolating-GAN architecture to correctly transform input image patches of real image scores to isolated music symbol to detect. That is why, we start by using a very simple experimental setting with only images containing one class of accidental symbol to detect. We then gradually stabilize and improve the training method, first by tuning hyperparameters, secondly by changing the loss function and data used during the training by adding more classes of accidental symbols and images without symbols to detect. Then, we investigate in section 5.2.2 how we can partially evaluate our method using a very small validation dataset together with a pretrained SSD to be able to correctly tune the Isolating-GAN training without using a fully annotated dataset.

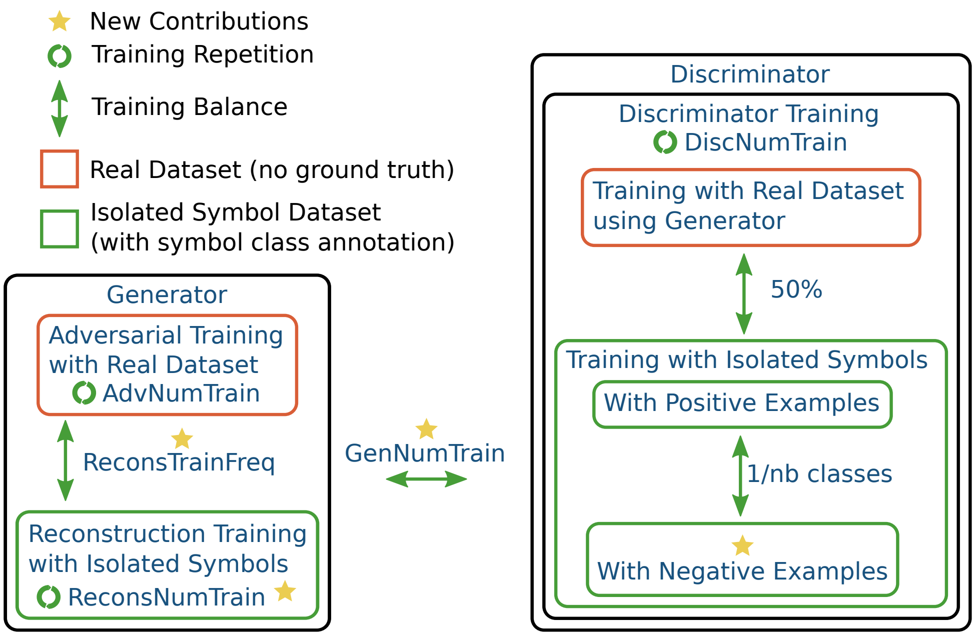


Figure 5.1 – Overview of architectural experiments of the Isolating-GAN method. Stars indicate what we consider our contributions. Double arrows indicate a notion of balance and circled arrows repetition.

5.2.1 Stabilize and Improve Isolating-GAN Training

Our first set of experiment concentrates on the incremental development of our method, starting from the original GAN architecture and adding successively the secondary reconstruction loss and the use of negative examples for stabilizing the training. We also propose a multi-class version of our architecture where we train our Isolating-GAN to isolate multiple classes of accidentals at the same time.

We first investigate how to balance the different learning rates used by the Isolating-GAN in section 5.2.1.1. Next, we present the iterative experiments that validate each of our proposed improvements to the vanilla architecture such as adding a reconstruction loss in section 5.2.1.2 and using negative samples in section 5.2.1.3. Finally, in section 5.2.1.4, we compare a multi-class version of our Isolating-GAN with our previously trained single-class version of our Isolating-GAN.

5.2.1.1 Balancing Discriminator and Adversarial Loss

Objectives In this section, we present the exploratory search of tuning the learning rates and training balance between the generator and discriminator of the Isolating-GAN method driven by isolated music symbols. We propose to first adjust the learning rates for the adversarial and discriminator losses using an independent search on a range of learning rate values. We then fine-tune this balance between the discriminator and generator by modifying the `AdvNumTrain` and `DiscNumTrain` parameters which regulate the number of consecutive time the generator and discriminator are trained using the same batch of data. The goal of these experiments is to find a set of parameters which produces a stable GAN training and good generation performance.

Datasets We start our experiments using the vanilla Isolating-GAN model and use two different datasets to do the adversarial training. The first dataset is the Isolated and Printed Music Symbol Dataset presented and illustrated in table 4.1. This dataset is used to automatically produce larger images of size 128×128 pixels with a white background. A single isolated music symbol is distorted using a random size coefficient and then pasted on the white background. This dataset will be used to inject the knowledge of what kind of symbols (with its shape and size) should the GAN retain or remove as explained previously in section 4.5.1. Since this dataset is automatically generated, we can derive the class and bounding box annotations of each symbols in the image which will later be used to pretrain a detector model.

The second dataset used is constituted of image patches of real music scores as explained in section 4.3 and containing hard segmentation problems like the ones presented in fig. 1.4. For these preliminary experiments, we artificially control the classes and distribution of classes used during training. For this section, we restrict this dataset only to images that contains the accidental symbol which we want to detect. It is the same dataset as the accidental dataset used for supervised accidental detection presented in table 2.1 except for the absence of the Reject class. For the experiments done in this section and for the purpose of further simplifying the detection task, we restrain the task to single class detection task and take the accidental sharp class as the class of symbol we want to detect. Later, starting from section 5.2.1.2, we generalize our approach to two other accidental classes. We also reuse the bootstrapping techniques as mentioned in section 4.3 to artificially augment this dataset to produce a total of 200k images.

Training Algorithm The GAN training uses the same algorithm as the original GAN work [Goodfellow 14] where we alternatively train the discriminator and generator. We start by using both datasets: automatically produced images containing isolated symbols and real images of music scores. The real images are then fed to the generator which transform them and ideally remove background noise and symbols and only keep the symbol to detect. Both sets of images are then fed to the discriminator and the discriminator loss is applied to the discriminator only by annotating the images containing isolated symbols as the **True** set while images produced by the generator are annotated as the **False** set. This step allows us to train the discriminator to search for symbols resembling isolated symbols on a white background in real images of music scores. This operation is then repeated `DiscNumTrain` times. In the second step, only images of real music scores are used. Images are fed from the generator to the discriminator and the adversarial loss is applied to the generator only by annotating this set of images as the **True** set. This operation is then repeated `AdvNumTrain` times. This step allows us to train the generator to fool the discriminator by generating images containing only isolated symbols on a white background. A more complete description of this algorithm is done in section 4.5.2.1.

Evaluation To directly evaluate the generation quality of the Isolating-GAN generator, we compute the pixel accuracy between the generated image and an artificially generated image which represent the ideal output of the generator. For example, if the input image patch extracted from a real music score contains both a symbol to detect and background noise and symbols, the ideal output of the generator would be an image where only the symbol to detect remains and everything else was removed and replaced by a white background. This artificial task allows us to evaluate the background filtering capability of the U-Net as well as the ability to keep and maintain the shape, size and position of the symbol we want to detect. In this case, the original pixel accuracy metric is heavily affected by the balance between black and white pixels in the ground truth data. That is why, in all our experiments, the pixel accuracy results shown are measures where the contribution of white and black pixels were balanced.

Although both our dataset constitution and task definition uses manually annotated ground truth information, the training of the Isolating-GAN is entirely done without any ground truth information. The ability to measure directly the generation capability of the GAN greatly helped us during the development of this method, and we believe that these results are showing very interesting properties of the GAN training behavior.

In order to gather a maximum of information from this pixel accuracy metric during a training, we propose to compute the minimum, maximum and average pixel accuracy across all the epochs of a training, except for the first 10 epochs. The maximum pixel accuracy of a training is the value that is classically retained in Deep Learning experiments and informs us about the best theoretical results that our method can achieve, although actually identifying the epoch that yield this maximum pixel accuracy value without using a fully annotated dataset is a difficult task which we discuss in section 4.5.3 and experiment in section 5.2.2. The average and minimum pixel accuracy of a training is a quick and synthetic way for us to describe the actual training curve/behavior of the model and therefore, the training stability of the model. The main interests of these preliminary experiments is to characterize and evaluate the initial stability of the model and the allowed range of values hyperparameters could take while still having a stable trainable model. The fact that we do not take into account the first 10 epochs of training is to be able to compute a meaningful minimum pixel accuracy value, since all trainings starts with an initial pixel accuracy of 0% (or in our case 50% since untrained models very often generate white or black images).

Knowing that our method is unstable and can produce widely varying results for the same set of hyperparameters, we repeat the training for the same set of hyperparameters

5 times and compute the average and standard deviation of the 5 runs on our three metrics: maximum, average and minimum pixel accuracies.

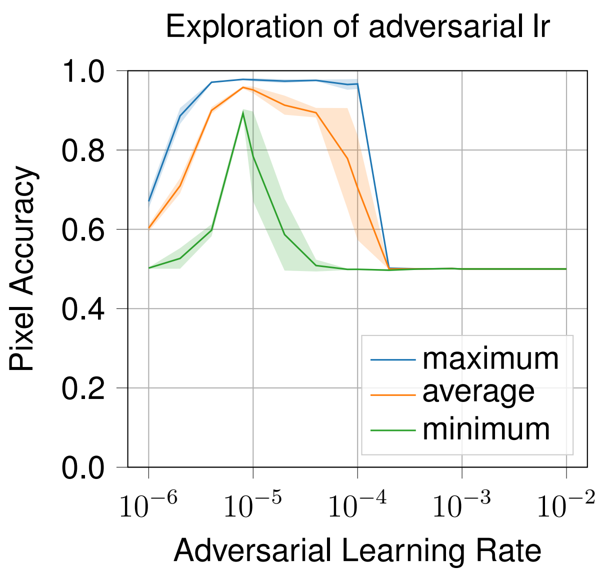
Isolating-GAN-AdvLr/DiscLr Experimental Results These preliminary experiments focus on the tuning of the learning rates parameters for the GAN adversarial training. From the original adversarial training, we need to tune both the adversarial learning rate and discriminator learning rate. The adversarial learning rate sets the speed at which the generator network will learn while the discriminator learning rate will set the speed of the discriminator training. Moreover, we fine tune this training balance between the generator and the discriminator by using two additional parameters: the adversarial number of training step (`AdvNumTrain`) and the discriminator number of training step (`DiscNumTrain`). These two parameters specify the number of successive training each network receive for a single batch of data. Although these parameters are not necessary and we could only try to tune the learning rates, we found that it was easier to use these additional parameters to fine tune the GAN performance.

First of all, the Isolating-GAN-AdvLr experiment explore the adversarial learning rate parameter with values between $[1 \times 10^{-6}, 1 \times 10^{-2}]$. Other parameters for this experiment are specified in table 5.1. The exploration is done on a log scale and as shown in fig. 5.2a, we found a typical bell curve showing that a learning rate of 8×10^{-6} gives the best results. In order to find this optimal value, we explored adversarial learning rate values in multiple consecutive experiments, by going smaller and smaller. By the time we found an optimal learning rate of 8×10^{-6}, later experiments like Isolating-GAN-DiscLr/AdvNumTrain/DiscNumTrain were already done using an adversarial learning of 1×10^{-5}. Since the two values were so close, we did not chose to redo all the experiments since it would have been too costly to rerun them for only marginal improvements in pixel accuracy results. Although we can not generalize the use of this value for other datasets or task, we show that a shallow grid-search is sufficient to stabilize the GAN training and produce reasonably good results. The same can be said for the discriminator learning rate with an optimal value of 2×10^{-4}. However, the training of the GAN still show a lot of instability as seen by the minimum pixel accuracy standard deviation in fig. 5.2b.

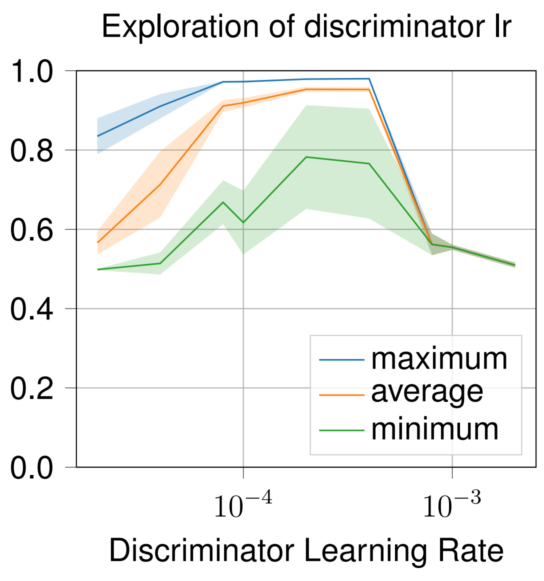
One interesting observation of these experiments is that the optimal adversarial learning rate is much smaller than the optimal discriminator learning rate. This behavior is coherent with the original GAN work [Goodfellow 14] which observe that the dis-

Table 5.1 – Parameter values for all Isolating-GAN experiments. Values between square brackets are the minimal and maximal bounds explored for the parameter.

Experiment	AdvLr	DiscLr	AdvNumTrain	DiscNumTrain
Isolating-GAN-AdvLr	$[1 \times 10^{-6}, 1 \times 10^{-2}]$	2×10^{-4}	1	1
Isolating-GAN-DiscLr	1×10^{-5}	$[2 \times 10^{-5}, 2 \times 10^{-3}]$	1	1
Isolating-GAN-AdvNumTrain	1×10^{-5}	2×10^{-4}	$[1, 16]$	1
Isolating-GAN-DiscNumTrain	1×10^{-5}	2×10^{-4}	4	$[1, 16]$



(a) Isolating-GAN-AdvLr



(b) Isolating-GAN-DiscLr

Figure 5.2 – Exploration of the adversarial/discriminator learning rate for the Isolating-GAN training with best AdvLr at 8×10^{-6} and best DiscLr at 2×10^{-4}. See section 5.2.1.1 for an explanation of the pixel accuracy metric. See table 5.1 on experiment Isolating-GAN-AdvLr/DiscLr for the value of other parameters. The maximum, average and minimum values are computed from 5 identical repeated training (see section 5.2.1.1).

criminator of the GAN should normally be trained optimally at every batch step before retraining the generator. In practice, we shorten the training of the discriminator for the training to be computationally tractable in a reasonable time. However, the use of a much larger discriminator learning rate guaranties us that the discriminator learns and adapt much faster than the generator.

Isolating-GAN-AdvNumTrain/DiscNumTrain Experimental Results In the following experiments Isolating-GAN-AdvNumTrain/DiscNumTrain, we try to fine-tune even further the balance of training between the generator and the discriminator by exploring respectively the `AdvNumTrain` and `DiscNumTrain` parameters. As explained before, these parameters regulate the number of successive training steps the generator or the discriminator has. We explore values between $[1, 16]$ and do a complete grid-search for these two parameters as shown in table 5.2.

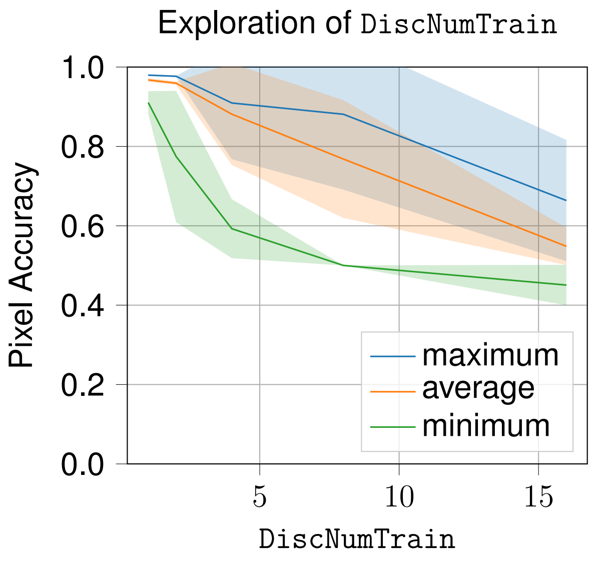
Figure 5.3 shows the optimal combination of using either 4 steps for `AdvNumTrain` or 1 step for `DiscNumTrain`. However, these results do not show that a maximum was reached for the `DiscNumTrain` parameter. Indeed, we can not explore values below 1 for the `DiscNumTrain` parameters which represent the number time the discriminator is trained with a batch of data.

Conclusion In this section, we have shown how we explored values for tuning the adversarial and discriminator learning rates. During the course of these experiments, we explored a total of 44 different combinations of parameters such as the discriminator and adversarial learning rate and the `DiscNumTrain` and `AdvNumTrain` parameters. Knowing that we repeated each training 5 times to evaluate the stability of our method, a total of 220 trainings had to be done. With each training taking 2 hours, it took around 3 weeks of computation time to complete all experiments. The fine-tuning of these parameters are essential to be able to train a GAN model with stability and produce good performance on the generation task. Our best pixel accuracy results at around 96% shows that the GAN learns to do the task of keeping the sharp symbol to detect while removing background noise and symbols. However, this task was learned by the GAN model without using a direct formulation of this task in the form of a training objectives, which in turns allows to learn this task without manually annotated ground truth. This fact is the main interest of our unsupervised symbol detection method.

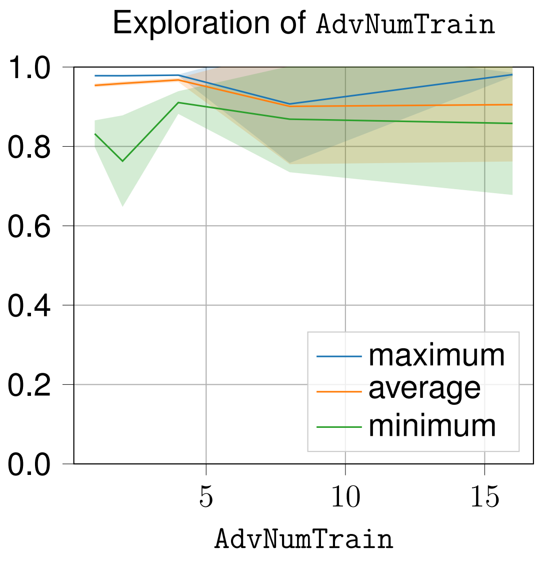
In the next section, we show how we can further improve the generation quality of

Table 5.2 – Grid-search parameters values for `AdvNumTrain` and `DiscNumTrain` parameters measuring maximum, average and minimum pixel accuracy during Isolating-GAN training (see section 5.2.1.1). The GAN was trained 5 times using the same set of parameters and we show the average and standard deviation pixel accuracy over these 5 training. The `AdvNumTrain` and `DiscNumTrain` parameters are used to tune the number of consecutive time the generator and discriminator are trained during batch training step.

Parameters		Pixel Accuracy: Average $\pm$ Std		
AdvNumTrain	DiscNumTrain	Maximum	Average	Minimum
1	1	0.978 ± 0.000	0.953 ± 0.004	0.831 ± 0.033
1	2	0.972 ± 0.002	0.920 ± 0.011	0.647 ± 0.107
1	4	0.968 ± 0.004	0.845 ± 0.018	0.499 ± 0.000
1	8	0.785 ± 0.068	0.545 ± 0.020	0.411 ± 0.036
1	16	0.844 ± 0.011	0.555 ± 0.018	0.379 ± 0.036
2	1	0.978 ± 0.001	0.958 ± 0.004	0.762 ± 0.115
2	2	0.979 ± 0.001	0.948 ± 0.005	0.684 ± 0.124
2	4	0.973 ± 0.003	0.875 ± 0.011	0.499 ± 0.000
2	8	0.760 ± 0.176	0.616 ± 0.115	0.467 ± 0.037
4	1	0.979 ± 0.001	**0.967** ± 0.005	0.910 ± 0.028
4	2	0.976 ± 0.001	0.959 ± 0.004	0.774 ± 0.165
4	4	0.909 ± 0.141	0.881 ± 0.127	0.592 ± 0.074
4	8	0.881 ± 0.190	0.768 ± 0.148	0.500 ± 0.000
4	16	0.663 ± 0.152	0.548 ± 0.047	0.450 ± 0.050
8	1	0.907 ± 0.148	0.900 ± 0.145	0.868 ± 0.133
8	2	0.979 ± 0.000	0.963 ± 0.005	0.804 ± 0.107
8	4	0.979 ± 0.001	0.963 ± 0.004	0.810 ± 0.123
8	8	0.977 ± 0.001	0.937 ± 0.017	0.633 ± 0.153
8	16	0.912 ± 0.133	0.842 ± 0.104	0.540 ± 0.058
16	1	0.981 ± 0.004	0.905 ± 0.143	0.858 ± 0.180
16	2	0.922 ± 0.119	0.913 ± 0.115	0.876 ± 0.101
16	4	0.979 ± 0.001	0.965 ± 0.010	0.813 ± 0.169
16	8	0.981 ± 0.001	0.958 ± 0.010	0.674 ± 0.192
16	16	0.979 ± 0.003	0.878 ± 0.059	0.533 ± 0.119



(a) Isolating-GAN-DiscNumTrain



(b) Isolating-GAN-AdvNumTrain

Figure 5.3 – Exploration of `DiscNumTrain`/`AdvNumTrain` parameters with the best DiscNumTrain at 1 and the best AdvNumTrain at 4. See section 5.2.1.1 for an explanation of the pixel accuracy metric. See table 5.1 on experiment Isolating-GAN-DiscNumTrain/AdvNumTrain for the value of other parameters.

the Isolating-GAN by using an additional reconstruction loss together with an improved training algorithm.

5.2.1.2 Improved Generator Training Using Isolated Symbols Reconstruction Loss

Objectives We now propose to investigate the use of an additional reconstruction loss in order to further improve the generation quality of the Isolating-GAN. We model a reconstruction task for the generator using the isolated music symbol dataset since isolated symbols are the only knowledge we have at hand. The isolated symbols allow us to give the Isolating-GAN explicit information of the symbols shapes and sizes we want to detect. As for the training of the discriminator, we automatically generate blank images which contains a deformed isolated symbol. However, in this case, the class set of isolated symbols is the three accidental classes: sharp, flat and natural. Since the GAN model is trained to detect a single symbol class, this reconstruction loss will model two tasks at the same time. First, if the class of the isolated symbol present in the image is the symbol class to detect, the generator will have to produce the exact same image and will have to learn an identity transformation. On the other hand, if the class of the isolated symbol present in the image is not a symbol class to detect, the generator will have to remove this symbol from the image and produce a white empty image. This behavior is illustrated in fig. 4.10.

With the addition of the reconstruction, we also propose to improve the training algorithm to better balance the training of the discriminator and the generator. In this experiment, we explore different values for the reconstruction learning rate to better adjust the contribution of this new loss to the training of the generator.

In contrast to previous preliminary experiments presented in section 5.2.1.1, we now conduct our following set of experiments with a more realistic modeling of the real image dataset, introducing images without symbols to detect and experimenting on more classes of accidentals. As we have explained before, the introduction of images with no symbols to detect degrades heavily the stability of our GAN training, which in turns degrades the detection results. However, this gets us closer to a realistic detection task as the one we propose in section 5.3 and therefore highlights the improved stability and detection performance of our newly added reconstruction loss. We also show the results using the mean Average Precision detection metric which allows a better evaluation of our detection task.

Datasets In addition to the two datasets used in the previous experiments section 5.2.1.1, we build another dataset based on the isolated music symbol dataset. Although the generation of the isolated symbol detection dataset does not change from our previous experiment explained in section 5.2.1.1, the synthetic detection dataset only contained images with symbols to detect. We now better model the distribution of classes in the dataset constituted of real images produced by our first preprocessing step section 4.3 by including images without symbols to detect. For all our following experiments, we propose to use a realistic 1 to 9 ratio between images containing an accidental symbol and rejection images that does not contain any accidental symbols. In the 10% of images containing an accidental symbol, the ratio between different accidental classes follows the real distribution of accidental classes produced by our syntactical method. We present more extensive evaluations of the impact of the frequency of reject images in appendix A, in section A.1.1.

For our new reconstruction loss to train the generator, we automatically generate images in the same manner as for the discriminator training using isolated music symbols. However, we also add isolated accidental symbols to reject and automatically produce ground truth images. The ground truth images are either identical to the input image if there is a simple symbol to detect or are blank images for input images with symbols to reject. In our case, we use the three accidental classes: sharp, natural and flat, but we only train the GAN to detect one of the symbol class at a time.

Training Algorithm We modify the original GAN training algorithm to include this reconstruction loss as presented in fig. 4.10. The goal of this new training algorithm is to be able to interleave the different training objective and correctly balance the training of the generator and the discriminator. This algorithm is described in detail in section 4.5.2.4.

Evaluation To better evaluate the detection task of our method, we propose to use the evaluation method presented in section 5.2.2. Since the end-goal of our method is to detect music symbol with a bounding box and a class label, we propose to use a Single-Shot Detector model pretrained on our automatically generated images containing Isolated Music Symbol Dataset. Using this setup, we can measure a mean Average Precision metric with an Intersection over Union over 0.75 (mAP [IoU > 0.75]). Using this metric, we can now compare our approach to a classic fully-supervised detector model. The evaluation architecture is constructed by forwarding produced images by the generator to the SSD model and then computing the mAP using detection ground truth annotations. As usual, we use this manually annotated ground truth only for evaluation in order to develop our method. Moreover, we show in section 5.2.2.1 how we can minimize the use of this ground truth data while still being able to correctly tune and improve the Isolating-GAN method. Our evaluation dataset consists of our image dataset extracted from real music scores but without restraining images to contain a symbol of the class we want to detect.

We measure the mAP metric at every epoch of the GAN training and report the maximum mAP attained during a training. For every different reconstruction learning rate values, we redo the training 10 times and report the average and standard deviation of the maximum mAP with IoU > 0.75. By reporting the maximum mAP value over a training, we can select the best configuration for our method and show the best theoretical upper limit of our method. This evaluation is done using a fully manually annotated detection ground truth which will not be available during the real application of our method We will show in the next section 5.2.2.1 how we can mitigate this issue and evaluate the real performance we can obtain with this method.

Isolating-GAN-ReconsLr Experimental Results In this paragraph, we now show the results using our additional reconstruction loss for our Isolating-GAN method. We propose to explore the behavior of the GAN training with 6 different reconstruction loss learning rates: $[1 \times 10^{-5}$, 3×10^{-5}, 5×10^{-5}, 7×10^{-5}, 1×10^{-4}, $7 \times 10^{-4}]$ as shown in fig. 5.5. Moreover, we show the results for three different class of accidental symbols: Flat, Natural and Sharp. We compare these results with a baseline experiment shown in fig. 5.4 where the experimental settings are identical to the Isolating-GAN-ReconsLr experiment but without the use of the reconstruction loss and without modification of the training algorithm. Our baseline results shows that our changes for a more realistic real dataset as input to the generator will effectively prevent the GAN to achieve any meaningful detection results with almost all runs producing a mAP of 0 except for one or two training outliers. This collapse is mainly linked to the addition of images with no symbols to detect, which were not present in the previous experiments. This shows the necessity of improving the performance and stability of the training of the GAN model.

In contrast, by using our additional reconstruction objective, we can see that both the natural and sharp class obtain their best results using reconstruction learning rate of 5×10^{-5} with respectively $\sim$80% and $\sim$89% of mAP with IoU > 0.75 while minimizing the difference between their 1st and 3rd quartiles. The flat class also shows a big improvement by achieving 64% of mAP average with a very small differences between their 1st and 3rd quartiles using a reconstruction learning rate of 1×10^{-4}. Since every training were repeated 10 times and we tested on 3 accidental classes and 6 different learning rates, the baseline experiment consisted of 30 training and this experiment consisted of 180 trainings, totaling to 210 trainings.

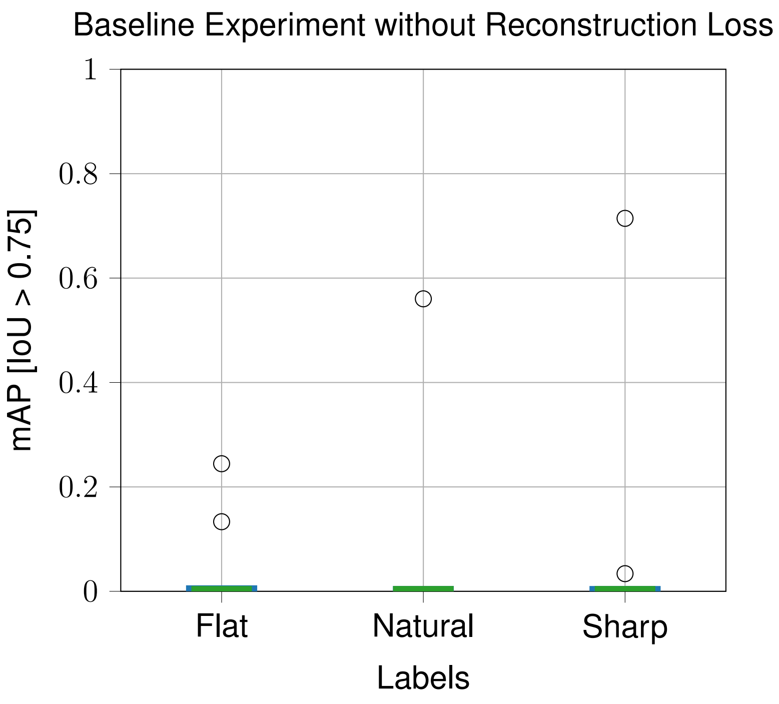


Figure 5.4 – Baseline results without using reconstruction loss. The experimental setting is identical to previous experiments except for the addition of images without symbols to detect, which drives most runs to collapse to 0% of mAP except 1 or 2 outliers. See section 5.2.1.2 for a detailed explanation of the mAP metric.

Conclusion In this experiment, we have shown the behavior of the Isolating-GAN when using an additional reconstruction loss together with a modified training algorithm on a new dataset including images without symbols to detect. We first start by showing that the original GAN architecture collapses on almost all trainings except a few. By using an additional reconstruction loss together with a modified training algorithm, we then start to obtain usable results for the task of music symbol detection. Moreover, we show that using a reconstruction task for the generator actually enables our method to

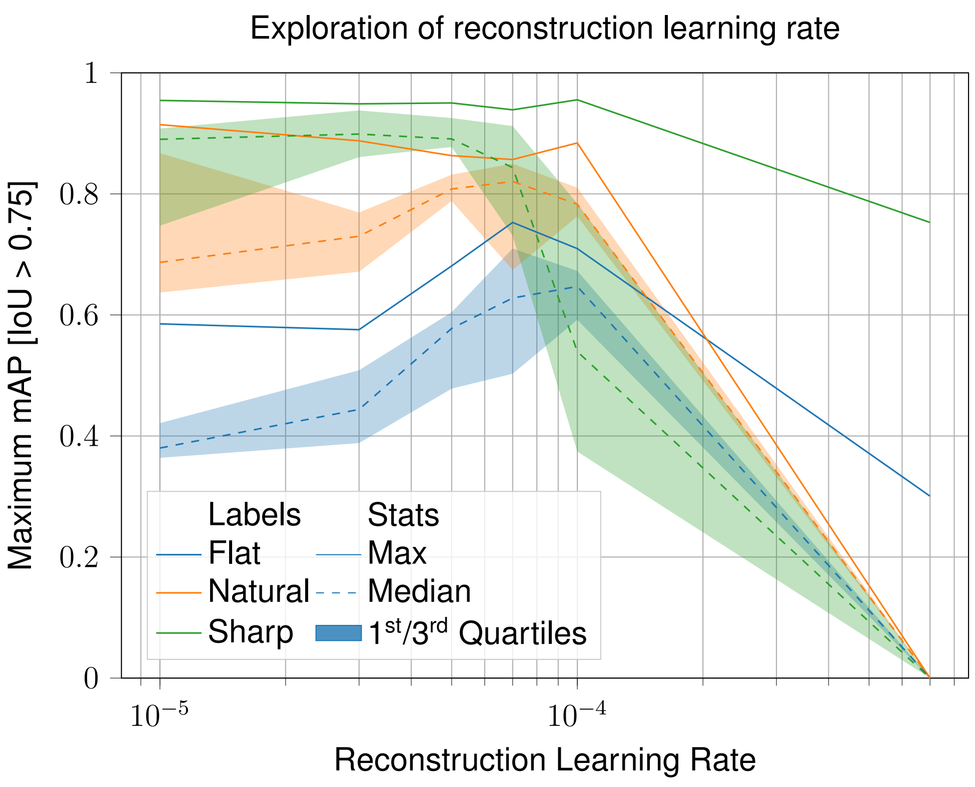


Figure 5.5 – Exploration of the reconstruction learning rate for the Isolating-GAN training. We show the maximum (solid lines), the median (dashed lines) and 1st and 3rd quartiles (limits of transparent areas) over 10 identical training of the maximum mAP achieved during a training. Best runs show 64% at 1×10^{-4} of learning rate, 80% at 5×10^{-5} and 89% at 5×10^{-5} of mAP for respectively the Flat, Natural and Sharp class. These optimum were chosen because of their small differences between the 1st and 3rd quartiles. See section 5.2.1.2 for a detailed explanation of the mAP metric.

work. The idea of using negative examples with isolated symbol to be removed by the generator can also be extended to the training of the discriminator. We show in the next section how we implement this improvement.

5.2.1.3 Improved Discriminator Training Using Negative Examples of Isolated Symbols

Objectives The purpose of this experiment is to present the last improvement of our Isolating-GAN detection performance by using negative examples of isolated music symbols for the discriminator training. For this experiment, we focus on the discriminator loss training objective which is used to train the discriminator network. In the original GAN training algorithm, the discriminator is made to recognize the origin of given images, which are either produced by the generator or from an existing dataset. In our case, the discriminator has to differentiate between real image patches of music scores transformed by the generator or images containing isolated music symbol of the class we want to detect. With this training objective, we give to the GAN the explicit information of which kind of symbol **should be present** in images produced by the generator. However, since we also have examples of isolated symbols which we do not want to detect, we can also give this information to the GAN, which in turn should improve the stability of the training and detection performance of the method. This way, we also give the GAN the explicit information of which kind of symbol **should not be present** in images produced by the generator.

Datasets For this experiment, we build on the previous Isolating-GAN-ReconsLr experimentation datasets described in section 5.2.1.2.

We modify the automatically produced images with isolated symbols used by the discriminator by adding two accidental classes. Therefore, images can contain any of the three accidental classes: flat, natural and sharp while the GAN is trained to detect only one of the accidental class at a time. When annotating these images for the discriminator loss, we annotate the images containing isolated symbols we want to detect as the **True** set while images containing other isolated symbols and images produced by the generator are annotated as the **False** set. The dataset produced from isolated symbols is balanced by the isolated symbol classes, which means that there will always be equal amount of symbols of each class.

Training Algorithm We reuse the training algorithm describe in our previous experiment Isolating-GAN-ReconsLr in section 5.2.1.2.

Evaluation To evaluate our experimentation, we use the same mAP with an IoU > 0.75 metric as presented in section 5.2.1.2. As said earlier, by reporting the maximum mAP value over a training, we can select the best configuration for our method and show the best theoretical upper limit of our method. This evaluation is done using a fully manually annotated detection ground truth which will not be available during the real application of our method However, we explain how we overcome this limitation by using a small validation set in section 5.2.2.1.

Isolating-GAN-ReconsLr+Neg Experimental Results In this experiment, we compare the results of using additional negative examples of isolated symbols in the discriminator loss shown in fig. 5.6 with an ablated experiment using the same hyper-parameters except for the use of negative examples. The use of negative examples shows an improved maximum mAP of 1.6% for the sharp class, 4.1% for the natural class while the maximum mAP for the flat class stays constant. However, the results for the flat class is clearly more stable with an improved minimum mAP of 61.8% compared to 45.3% previously and a reduced 1st and 3rd quartile variation going from 20.7% difference to 6.4% difference between the 1st and 3rd quartile. This clearly shows that the use of negative examples helps the GAN to better filter and extract symbols to be detected by our pretrained SSD. For this experiment, we repeated our trainings 10 times on 3 different accidental classes, totaling to 30 trainings.

Conclusion We have shown that the use of the negative examples of isolated symbols in the discriminator training help the Isolating-GAN to better filter and extract music symbols we want to detect. By using a combination of improvement such as an additional training loss and modified training algorithm, careful tuning and balance of the training of the generator and discriminator and use of additional information through an isolated symbol dataset, we obtain competitive detection results using a totally unsupervised training method for the task of music symbol detection. However, during these experiments, we use a manually annotated ground truth dataset which was necessary to **evaluate** and tune the hyperparameters of our method. Since the goal of our method is to be used in a completely unsupervised setting, we need to resolve

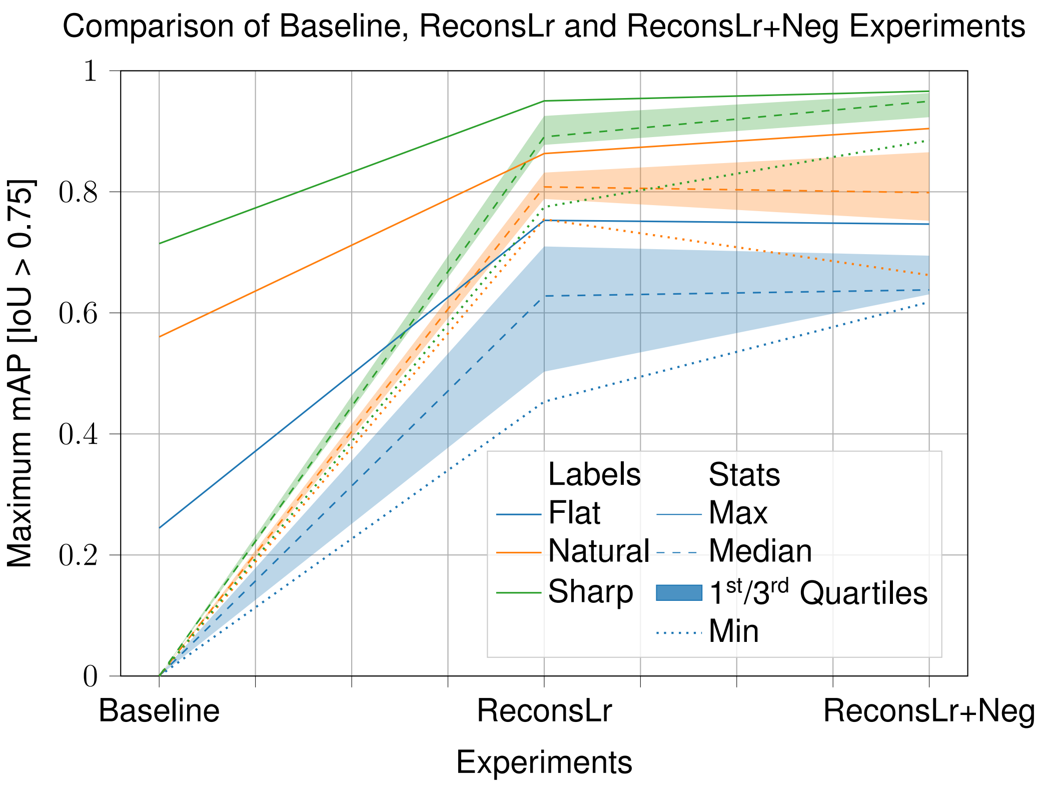


Figure 5.6 – Comparison of results obtained using a simple GAN (Baseline), GAN + Reconstruction Loss (Reconslr) and GAN + Reconstruction Loss + Negative Examples (ReconsLr+Neg). We show the results for three accidental classes over 10 identical training of the maximum mAP achieved during a training. The use of an additional reconstruction loss improves dramatically the results while the use of negative examples stabilize the Flat class training and slightly improves the Sharp class. See section 5.2.1.3 for a detailed explanation of the mAP metric.

the problem of evaluating our model when no manual annotations are available. In our next section, we mitigate this issue by using a small manually annotated validation dataset which can be used to tune the hyperparameters of our method and select the best trained GAN out of a pool of trained GANs.

5.2.1.4 Multi-class Isolating-GAN

Objectives Until now, we trained the GAN part of our Isolating-GAN to extract only one class of accidentals. Although this choice allowed us to simplify our experimental settings, it has the drawback of having to train multiple times the same GAN model to extract each classes of accidentals. Moreover, it is not clear if training a single GAN to extract multiple classes of accidentals would improve or degrade the overall generation quality of the GAN generator. The purpose of this experiment is to evaluate the benefit of training the GAN part of our Isolating-GAN to extract multiple classes of accidentals simultaneously.

Datasets In order to convert our GAN generator from a single class extractor model to a multiple class extractor model, we only have to modify the data distribution of the isolated symbol dataset used for training the GAN discriminator. Instead of using only one class of isolated accidentals, we just modify the data distribution to evenly include isolated accidentals of multiple classes (Sharp, Natural and Flat).

Experimental Settings For this experiment, we build on our previous single class experiment explained in section 5.2.1.3. However, the single class experiment had different values for the learning rate of the reconstruction loss applied to the generator for the different accidental classes. Therefore, we resolved the differences by using the learning rate of the best performing accidental, the sharp in our case, and reused the value of 5×10^{-5}.

Mono-class vs Multi-class Isolating-GAN Results To evaluate our experimentation, we first compare the results between our previous mono-class experiment section 5.2.1.3 and our new multi-class experiment results. In this case, we use the AP computed on the whole training dataset and do not use any early stopping mechanism.

In fig. 5.7 we can see a big improvement for the Flat class which previously gave weaker results than the other accidental classes in a single class training setting. For the multi-class setting, the Flat class now gives comparable results to other accidental classes and the Natural class results are considerably more stable than in a single class training setting. The Sharp class shows similar results in the mono-class vs multi-class settings.

For this Multi-class experiment, we repeated each training 10 times but trained on all accidentals simultaneously, reducing the number of training to 10.

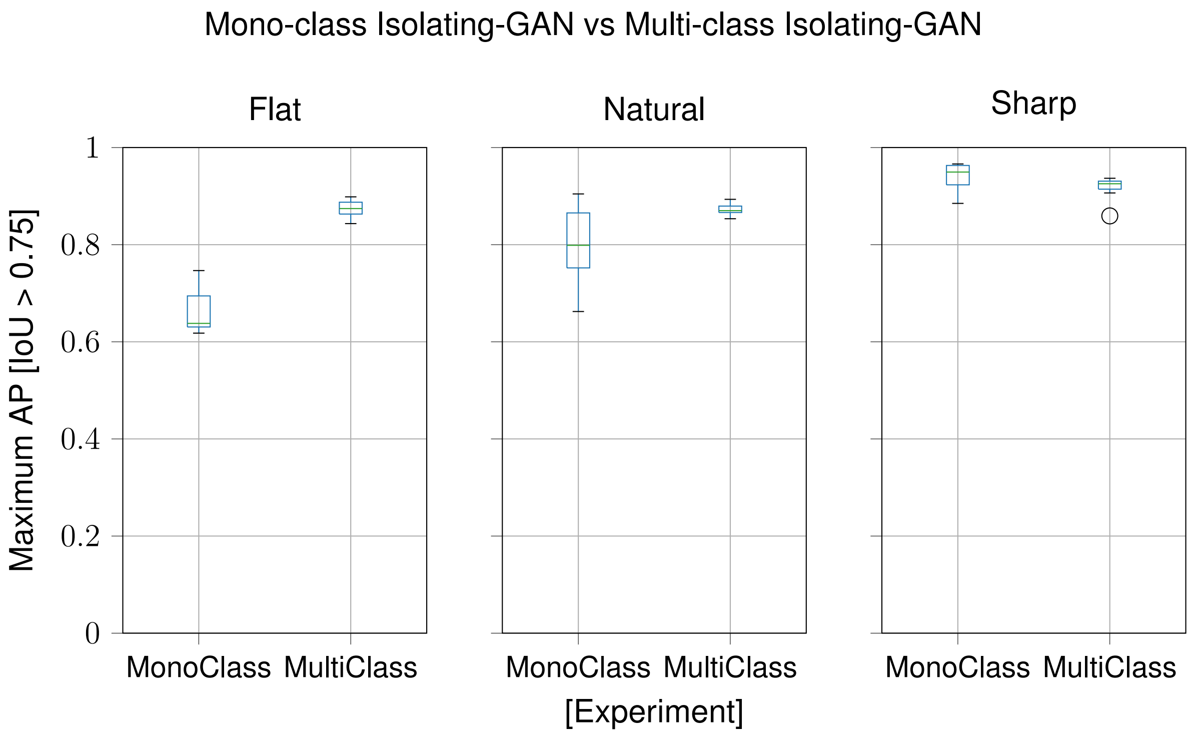


Figure 5.7 – Per class AP comparison of the Mono-class Isolating-GAN vs Multi-class Isolating-GAN. AP computed on the whole training dataset (MaxFull as shown in fig. 5.8). The boxplots are computed from 10 repeated trainings using the same set of hyperparameters. In a multi-class setting, the maximum mAP of the Flat class has improved from 75% to 87% and the Natural class stability has greatly improved with a mAP standard deviation reduced from 7.5% to 1.1%.

5.2.1.5 Conclusion

In this section, we have presented the key experiments that drove the design of our Isolating-GAN method. We started the implementation of our method by using the original GAN architecture as a base architecture and showed that the architecture could be trained to correctly isolate accidental symbols using appropriate learning rate parameters. While the images generated by the original GAN architecture indicated that our method could indeed isolate real accidental symbols while keeping their sizes intact, the actual detection performance is not sufficient for an OMR system. Moreover, the dataset used was not realistic as it only contained images with symbols to detect. Once we added images without symbols to detect in training of our method, the GAN training collapses and our method is not able to detect accidental symbols. Therefore, we propose to improve the original GAN architecture using an additional reconstruction training objective based on isolated music symbols that improved the detection quality dramatically. Another additive improvement we have shown is the use of negative examples both in the generator training and discriminator training that slightly improved and stabilized the training process.

Finally, we show that our GAN architecture can be trained to isolate multiple classes of accidentals at the same time by simply changing the isolated symbol data used for the training of the discriminator, with no changes to the actual GAN architecture. Moreover, the task of isolating multiple classes of accidentals actually improved significantly the detection quality of the flat accidental class and stabilized the detection quality of the natural class, showing that the information learned from one class of accidental symbol is actually beneficial for the detection of other accidental symbol classes.

The question that remains in order for our Isolating-GAN to be complete is the inability to evaluate the detection performance of our method, even for fine-tuning the hyperparameters. In the case of a real usage scenario, no ground truth or test set will be available to evaluate the detection performance of the model. With the added observation of the relative instability of our method giving large variation of detection performance for repeated training with the same hyperparameters, we need an evaluation strategy that minimize the use of manual annotation while being able to identify the best performing set of weights between multiple trained models and during the training process of each model as well. In the next section, we show series of experiments that present and validate the use of the evaluation strategy we propose to use in section 4.5.3.

5.2.2 Isolating-GAN Evaluation Strategy

Since we have now presented the experiments validating the architecture and training methodology of our Isolating-GAN method, we focus this section on our ability to evaluate our method in spite of the lack of annotated data and instability of our model.

Our end-goal task is the accurate detection of music symbol in real historical printed music scores, we previously proposed to use a pretrained SSD model to evaluate our Isolating-GAN method as explained in section 4.5.3.1. However, to be complete, our method need to be usable in a scenario where no manually annotated ground truth is available for the training data used.

Therefore, in section 5.2.2.1, we will see how we can use a very small evaluation dataset together with an early stopping mechanism to select the best performing model and therefore correctly tune the hyperparameters of our model. Afterwards, we then also show the use of this early stopping mechanism applied to the multi-class version of the Isolating-GAN in section 5.2.2.2.

5.2.2.1 Early Stopping Using Small Validation Set

Objectives Using a SSD model to produce detection results from images generated by the GAN allows us to evaluate our Isolating-GAN method with the common mean Average Precision metric for detection. However, we still have the problem of having no ground truth information to compare our prediction with.

In the case of real usage of our method and given the relative instability of our training method, the primary evaluation goal is to have the ability to correct and fine-tune the hyperparameters of our Isolating-GAN, while the precise evaluation of the performance of the method can be done at a later stage when sufficient ground truth has been gathered using the results of our method. Therefore, we propose to use a very small validation dataset which should be sufficient to discriminate the best trained GAN for a set of different hyperparameters. Because of the instability of our method, we go even further and use this small validation dataset to discriminate the best trained GAN out of a pool of trained GAN with identical hyperparameters. In the same manner, because of the instability of the GAN model during the training itself, we use this small validation set to choose the best epoch to stop the training using an early stopping mechanism.

We illustrate this mechanism in fig. 5.8 by showing an example of using both a fully annotated training dataset and a small validation dataset of 10 examples per class to evaluate the mAP metric during a GAN training. First of all, we can see that the training is very unstable which makes it critical to stop the GAN training at an epoch that maximize the performance of the GAN (MaxFull in fig. 5.8) and not after a fix number of epoch. In order to study our evaluation method, we compute the optimal GAN performance MaxFull by using a fully annotated training dataset with detection ground truth. However, in a real use-case situation, no such ground truth will be available. Therefore, we compare this optimal performance with a mAP estimated MaxSmall using a small validation dataset. This estimation will then be used to decide when the GAN training should be stopped using a simple early stopping mechanism with a patience parameter. We also report the actual performance of the GAN computed on the fully annotated training dataset at the epoch which maximized the mAP on the small validation dataset called EarlyStop in fig. 5.8. This shows us the loss in precision between using a fully annotated training dataset and a small validation dataset.

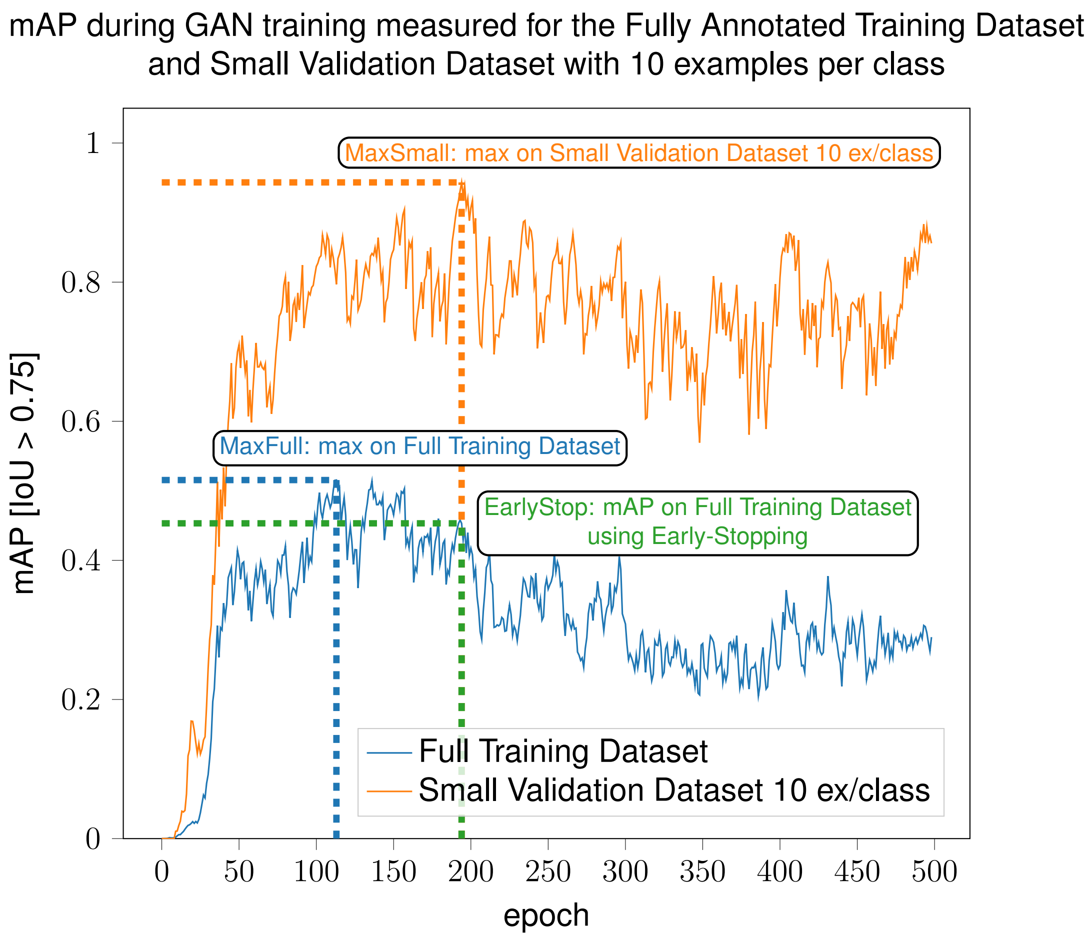


Figure 5.8 – Example of mAP computed using fully annotated training dataset and a small validation dataset with 10 examples per class during a typically unstable GAN training. We also show how we report the real GAN performance when using the small validation dataset by extracting the mAP on the fully annotated training dataset at the epoch that gave the maximum mAP on the small validation dataset.

Unfortunately, we do not have a definitive answer on the size of this small validation set. We believe this size will be dependent of the training dataset used and the manual annotation capacity of the user. However, we believe that a single user with a few hours of manual annotations should be largely sufficient to produce this small validation dataset.

Dataset The following experiment evaluates our Isolating-GAN using the dataset previously presented in section 5.2.1.3 and using different sizes for the small validation set. We show the evolution of the real and estimation of the detection performance using a fully manually annotated training dataset and a small validation dataset of variable size. We start with a validation size of 10 symbol examples per class, which gives us 40 ground truth examples equally distributed between three accidental classes and a reject class.

Results Figure 5.9 reports the real mAP computed on the full training dataset at the epoch that maximize the mAP on the small validation dataset. We show 10 trainings using the same experimental parameters but with a different random seed for each different target accidental class to detect. In this case, such a small validation dataset does not give an enough accurate estimation of the performance of the GAN training. We can also see a high discrepancy between the maximum mAP achieved by the fully annotated training dataset and the mAP given by the epoch which maximized the mAP on the small validation dataset.

The same experiment was done with a slightly larger validation dataset with 20 examples per class shown in fig. 5.10. Although we start to see some improvements on the estimation using the small validation dataset, the chosen run for the Flat class and Natural class is still not the best run possible.

Next, we propose to reuse our bootstrapping method as explained in section 2.2.1 to artificially augment the small validation dataset. We show the results using a base of 10 examples per class fig. 5.11 and 20 examples per class fig. 5.12. As can be seen in the results, using bootstrapping with 20 examples per class significantly improve the reliability of the mAP estimation given by the small validation dataset. It especially improves the early stopping mechanism of the GAN training by identifying an epoch that maximize the real precision of the GAN on the fully annotated training dataset.

In this section, we show how we can use a small validation dataset of 20 examples together with an augmentation technique such as bootstrapping to accurately evaluate

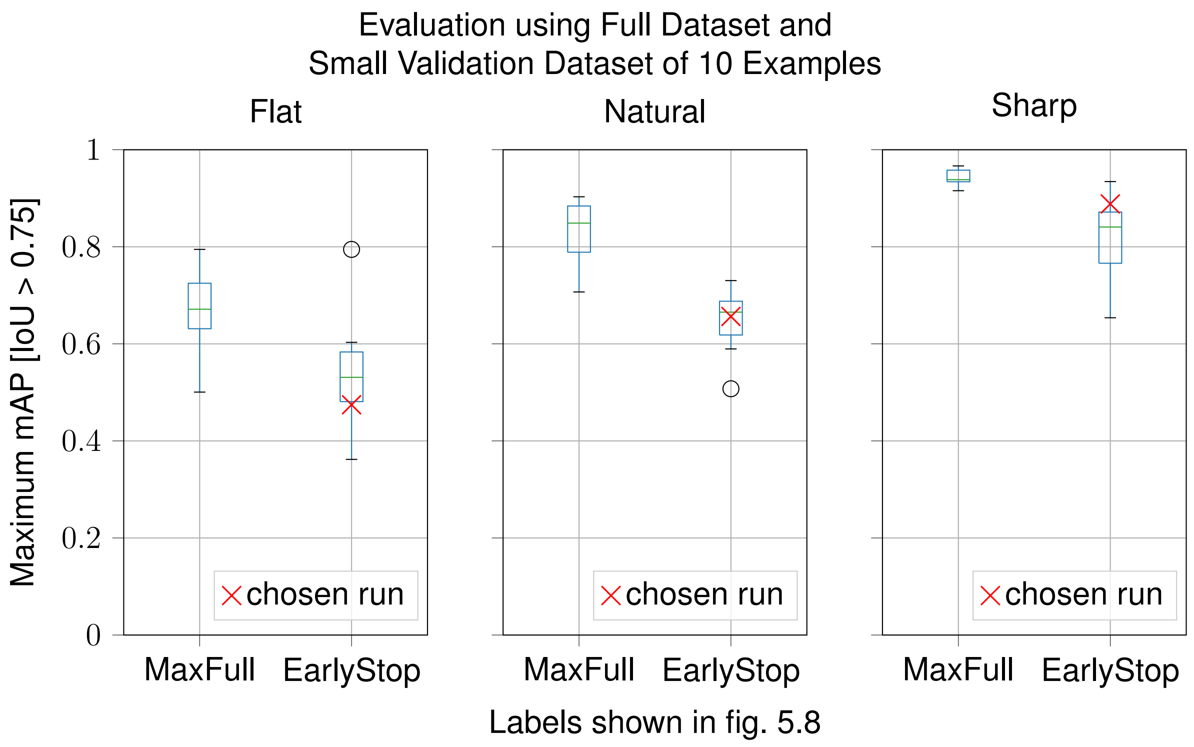


Figure 5.9 – Estimation of the Isolating-GAN mAP using early stopping with a small validation set of 10 examples per class. 10 examples per class is not sufficient to correctly estimate the detection performance of the model. There is a large discrepancy between the mAP computed between the whole ground truth (MaxFull) and the small validation set of 10 examples (EarlyStop). The chosen run is also not the best trained run.

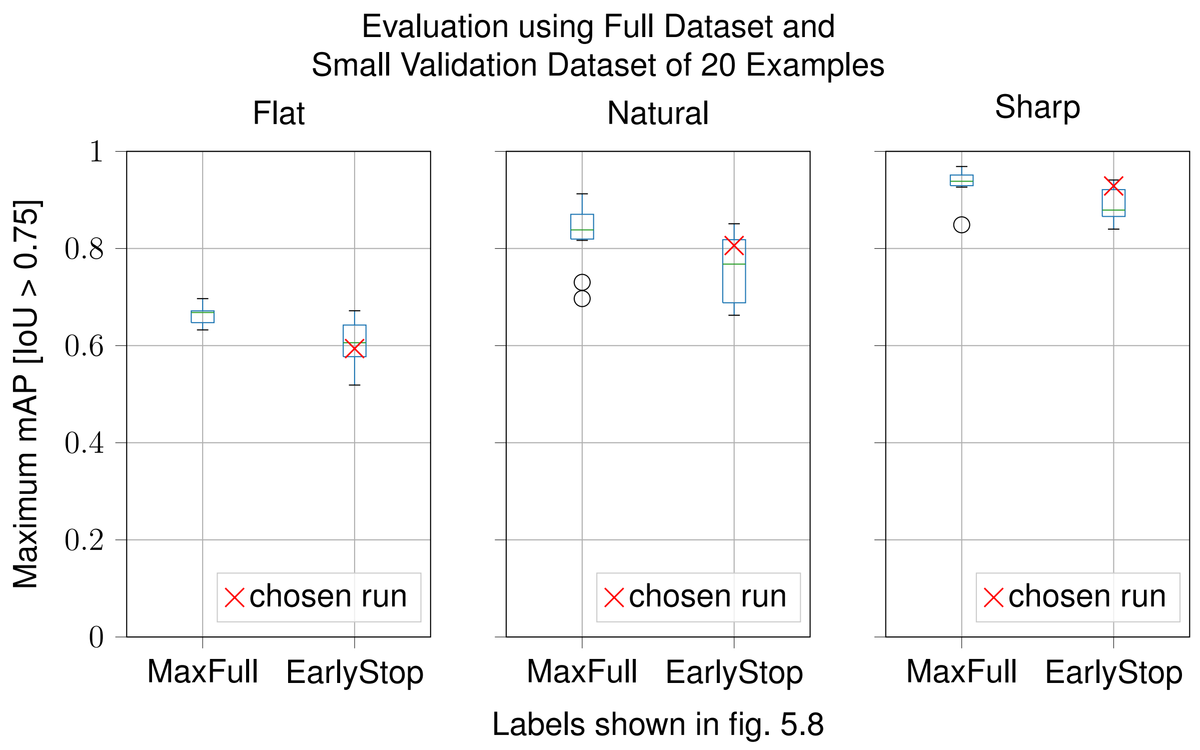


Figure 5.10 – Estimation of the Isolating-GAN mAP using early stopping with a small validation set of 20 examples per class. Improved mAP estimation compared to a validation set of 10 examples per class but the chosen run is still not the best trained model available.

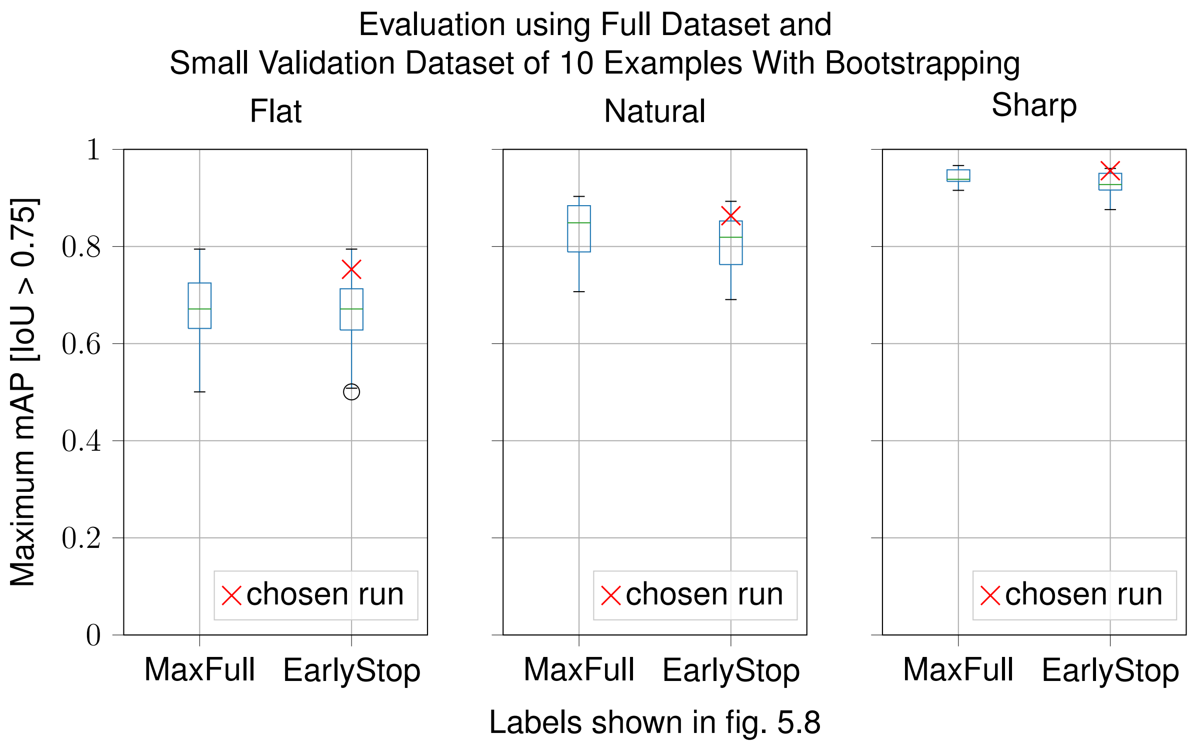


Figure 5.11 – Estimation of the Isolating-GAN mAP using early stopping with a small validation set of 10 examples per class augmented with bootstrapping. The mAP estimation is already much better than fig. 5.9 but the chosen run is still not the best available run for the flat and natural class.

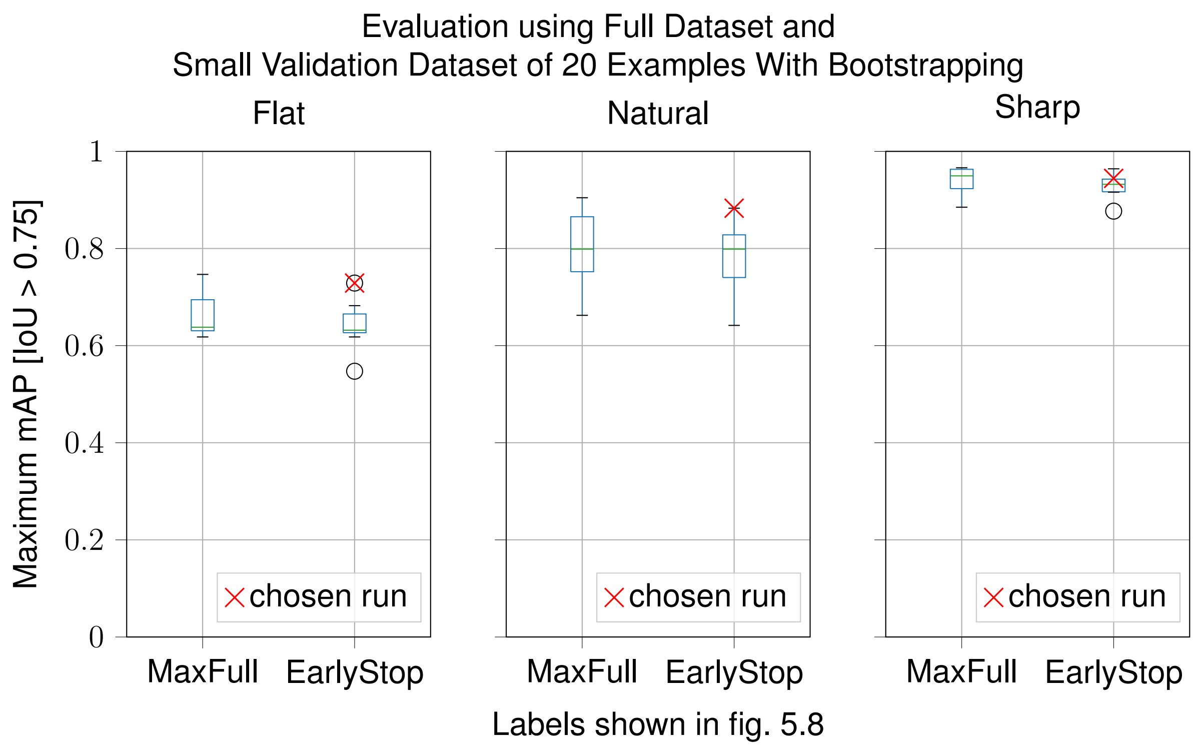


Figure 5.12 – Estimation of the Isolating-GAN mAP using early stopping with a small validation set of 20 examples per class augmented with bootstrapping. In the context of this dataset, 20 examples per class seems sufficient to correctly estimate the mAP of trained model and choose the correct (almost) best trained model for all classes.

our Isolating-GAN method without a fully annotated dataset. Using this method, we are able to do an early stopping of the GAN training that maximize the real mAP of the GAN while keeping the manual annotation effort at a minimum. Since we repeated every training 10 times, tested on 3 different accidental classes and tried 4 different validation dataset, this experiment took 120 trainings to complete.

5.2.2.2 Multi-class Isolating-GAN Early Stopping Results

Objectives We also show the use of our early stopping mechanism in a multi-class settings. The objective in this experiment is to show that our early stopping mechanism and the use of a small validation dataset of 20 examples per class can also be used to correctly evaluate and fine-tune the multi-class version of our Isolating-GAN method.

Experimental Settings This experiment is identical to the previously presented multi-class experiment in section 5.2.1.4. For the small validation set, we use the same validation set of 20 examples per class with bootstrapping as presented in section 5.2.2.1. One difference in the evaluation protocol from the single class experiment previously presented in section 5.2.2.1 is that we only have to identify a single model for all three class of accidentals instead of one model per class. Therefore, instead of maximizing the Average Precision (AP) of each accidental class independently, we maximize the overall mean Average Precision (mAP) which is the average of each accidental Average Precision.

Results The results of the experiment as shown by fig. 5.13 demonstrate that our evaluation protocol with a minimal validation set of 20 examples per class with bootstrapping is able to accurately estimate the performance of our model for the Flat class identifying the best performing training epoch and best performing model out of ten trained models.

For the other Sharp and Natural class, the early-stopping mechanism is working correctly since the maximum mAP given by the early-stopping mechanism is similar to the maximum mAP given using the whole annotated dataset. However, the identification of the best performing model is not as accurate as the Flat class. We believe this is the effect of maximizing the mAP instead of maximizing the AP of each accidental classes. This shows that the model that performs the best across the different classes of accidentals as shown by the results on the mAP does not necessarily present the best results for each accidental classes.

For this multi-class experiment, we repeated each training 10 times and simultaneously on all classes of accidentals, reducing the number of trainings to 10.

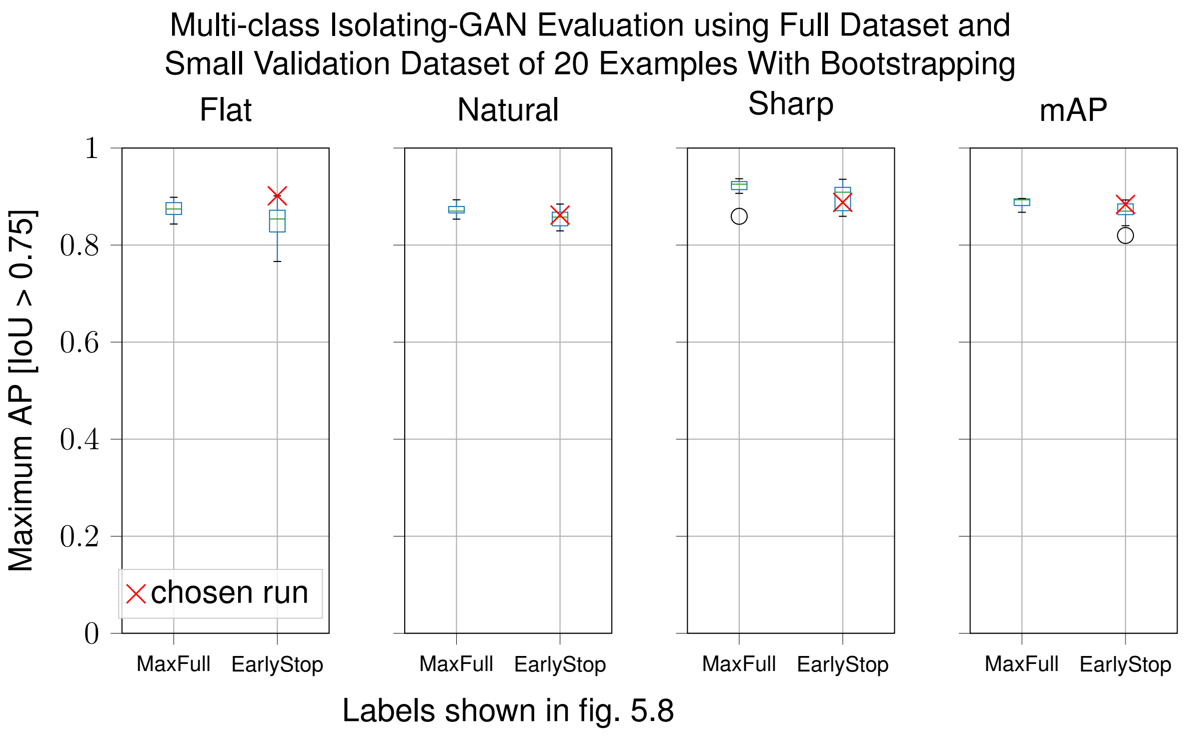


Figure 5.13 – Estimation of the Multi-class Isolating-GAN mAP using early stopping with a small validation set of 20 examples per class with bootstrapping. Our early-stopping strategy is able to distinguish one of the best performing model relative to the mAP, however, the model presenting the best mAP results does not necessarily present the best AP for each class of accidentals.

In the next section, we present our final set of experiments evaluating our method using the best set of hyperparameters and comparing the results on two different accidental detection datasets: our small accidental dataset where we controlled the amount of images without symbols to detect and a large accidental dataset with the real distribution of images with and without symbols to detect.

5.3 Overall Isolating-GAN Evaluation

The experiments that we have presented until now have shown in section 5.2 the development of the Isolating-GAN architecture and our evaluation strategy in regard to the instability of the method. Because of the unsupervised and unstable nature of our method, we believe it is important to present the process of designing such method where the evaluation of each step guides the overall design of the method.

With all the experiments done previously, we can now identify a single set of hyper-parameters (see section 5.3.1) which will hopefully produce a highly accurate accidental symbol detector. Before we present the final results of our method, we start this experiment by presenting an ablation study of the method in section 5.3.2 where we remove the generative transformation capability of our method. This ablation study will allow us to measure the impact of the second step of image-to-image translation, which we propose as our main contribution. Subsequently, we compare in section 5.3.3 the ablated results with the final and complete iteration of our method.

5.3.1 Experimental Settings

In this experiment, we gather the set of hyperparameters that provided us with the most accurate accidental symbol detector. For this, we use the multi-class version of the Isolating-GAN as shown in section 5.2.1.4 together with the same learning rates and batch size settings. As before, we repeat the training of the Isolating-GAN 10 times and compute the mAP detection metric to evaluate the detection performance of our method. We use the same process for selecting the best performing training epoch and the best performing model between the 10 run as before, using the small SSD and our small validation dataset as presented in section 5.2.2.1. For additional insights, we also propose to show in detail the precision and recall (in the context of detection as explained by Everingham et al. [Everingham 10]) as it will allow a better understanding of the amount of symbols correctly detected and the amount of false positives produced by the detection model.

However, for this experiment, we propose two main changes in order to better evaluate the method. First, in section 5.3.1.1, we propose to use another larger accidental detection dataset to evaluate our method against a more challenging dataset. Indeed, our previously used dataset has been manually modified to reduce the number of images without symbols to detect. This limitation allowed us to simplify our task and progressively tune and stabilize our model training. However, in this final section, we experiment on a new large accidental detection dataset with the real distribution of symbols in images. Secondly, in section 5.3.1.2, we improve the pretrained isolated symbol detector by using a larger and more accurate detector architecture and consequently improve the general detection accuracy of the method.

5.3.1.1 Large Scale IMSLP Dataset

All previous experiments used a small accidental detection dataset [Choi 18a] which was fully manually annotated. However, as mentioned in section 5.2.1.1, we artificially modified the balance of images containing accidental symbols to detect and empty images containing no symbol to detect. This modification was done to ease the development of our method, since the amount of images without symbols impacted heavily the stability of our method.

For this experiment, we remove this artificial modification by showing the application of our method on a more challenging, 100 times larger, accidental detection dataset. This dataset is constituted of 58 public domain scores gathered from the IMSLP library [1] and each score was chosen to maximize the amount of different music score publishers from the late 18th century to the 20th century. Scores for piano, quartet and orchestra were chosen because of their challenging characteristics such as the use of complex music notation structure, high symbol density and image degradation artifacts caused by the imprinting technique and time. However, all scores have a reasonably good image resolution with a minimum of 20 pixels for the height of an interline (correspond to approximately 300 DPI). The dataset amounts to 1,812 pages of music scores and after the application of our first processing step of identifying RoIs presented in section 4.3, we identified a total of 292,463 RoIs which is a 100 times more than our small accidental detection dataset.

Given the sheer amount of data for this new dataset, it was impossible for a single annotator to manually annotate the whole dataset. Instead, we propose to annotate only a subset of the total dataset such as one page out of 21 of the most challenging music scores. As a result, we have annotated 4,563 RoIs and 818 accidental symbols. A comparative table between our small and large accidental dataset can be found in table 5.3.

1. petrucci, *IMSLP/Petrucci Music Library: Free Public Domain Sheet Music*, July 19, 2017, URL: `http://imslp.org/` (visited on 07/19/2017).

Table 5.3 – Small and large accidental detection dataset.

Dataset	Small	Large
Editors	1	43
Scores	5	58
Pages	70	1,812
Region of Interests	2,955 (limited amount of empty images)	292,463 (real distribution)
Annotated Page/Region	70/2,955	21/4,563
Annotated Symbols	2,150	818

For the training of our Isolating-GAN method with this large scale dataset, we also introduced some variations on the synthetic generation process of the isolated symbol detection dataset. These modifications illustrate how our method can quickly adapt to a new dataset with little effort. After noticing musical notes being confused with flat symbol, we added isolated musical note symbol as reject examples in the isolated symbol detection dataset in the same way we added negative examples in section 5.2.1.3. Moreover, we noticed lots of detection confusion at the borders of image patches, where the GAN model would either remove symbols with more than 75% of their area inside the image or keep symbols with less than 75% of their area in the image. To help the training of the GAN with images on the border, we also modified a little our isolated detection dataset synthetic generation to include partial symbols at the border of the image, where isolated symbols with less than 75% of their area inside the image are treated as reject examples and symbols with more than 75% of their area inside the image as positive examples. These modifications were added only to our new large accidental dataset, while we kept the previously small accidental dataset unchanged.

5.3.1.2 Improved Isolated Symbol Detector

The evaluation of our method rely on an isolated symbol detector pretrained using our isolated symbol dataset. The overall detection result depends both on the quality of the image produced by our Isolating-GAN generator and the detection quality of the isolated symbol detector. Until now, we have used for our experiments a very small Single Shot Detector (SSD) which is only able to take as input a small image patch of 128×128 pixels and uses a small feature extractor network with 7 convolutional layers.

The motivation for taking such a small and fast detector was to gain speed during the training of our experiments because the detector needs to run on all annotated images at every epoch of the training in order to apply our early-stopping strategy.

However, once our Isolating-GAN trained, we propose to evaluate the detection performance of our method using a more accurate but slower isolated symbol detector. For this, we propose to reuse the original SSD architecture able to take as input an image of 300×300 pixels and uses the original VGG-16 feature extractor network. This new detector is pretrained using the same isolated symbol detection dataset as the previous smaller detector. Note that this detector was **not** used for our early stopping mechanism presented in section 5.2.2.1.

5.3.2 Ablation Study

Objectives Before presenting the detection results obtained using our Isolating-GAN method, we propose to first evaluate our method without applying the second step of image-to-image translation, see section 4.5, by removing the GAN model in our processing pipeline. Therefore, in this ablation study, we first identify the RoIs using the first step of our method, see section 4.3, then apply the pretrained isolated symbol detector which is the third step detecting isolated symbols previously presented in section 4.4. This process is illustrated in fig. 5.14. In this case, there is a discrepancy between the images extracted from real music scores using the RoIs that can contain background symbols, noises and degradation, and the clean images containing only isolated symbols on a white background of the isolated symbol detection dataset used to pretrain the isolated symbol detector.

Ablation Study Results We show the ablation study results on both the small and large accidental detection dataset. Figure 5.15 shows the precision, recall and AP for the three accidental classes Flat, Natural and Sharp as well as the mAP. We can see that the recall measure is very good with an average of 96% for the small dataset and 88% for the large dataset. This result shows that the detector is able to correctly detect accidental symbols when present in the image even when background information, noises or degradations are present in the image.

On the other hand, the precision results are really low, with very unbalanced results across the different accidental classes for the small dataset with only 22% of precision for the flat class with an average precision of 65%. The large dataset has an even

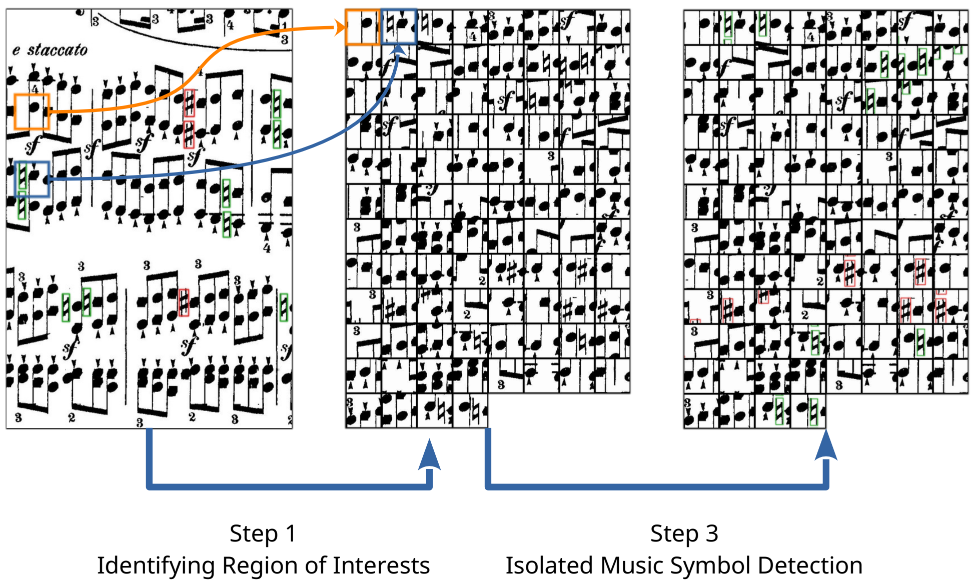

Step 1
Identifying Region of Interests
Step 3
Isolated Music Symbol Detection

Figure 5.14 – Ablation study processing pipeline. Only the first RoIs identification step and third isolated symbol detection step is used while the second GAN generator transformation step is left out. See fig. 4.1 for the original 3 steps processing pipeline.

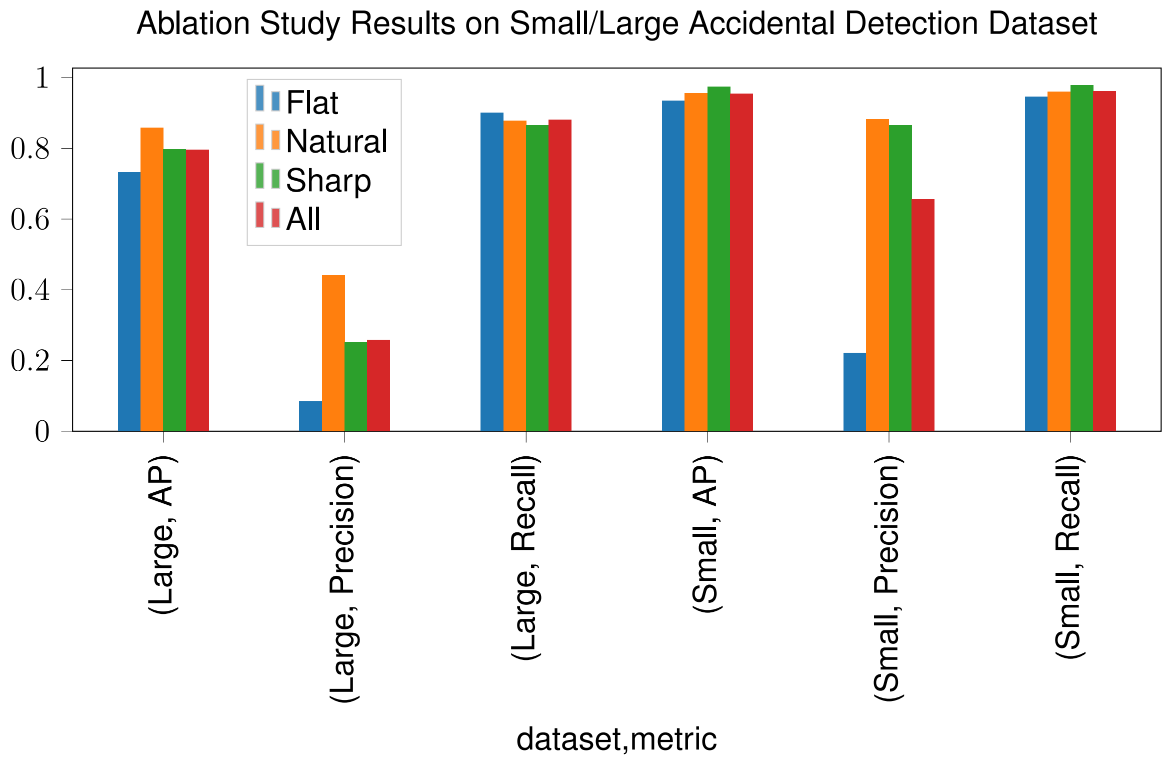


Figure 5.15 – Ablation study results on small and large accidental dataset. Recall is high while precision is very low showing that the isolated symbol detector is able to detect existing symbols but also incorrectly detect symbols in background.

lower precision with an extremely low precision of 8% for the Flat class and an average precision of 26%. These results show that the isolated symbol detector is not able to correctly differentiate between background information and accidental symbol to detect. Figure 5.16 shows the amount of correct detections versus the amount of incorrect detections. For the small dataset, the isolated symbol detector produced 1,190 incorrect detection, and for the large dataset 2,696 incorrect detections. Even with such small amount of data, it will be very time-consuming to manually correct these predictions. Therefore, we can not use the predictions from only the isolated symbol detector for our OMR processing pipeline.

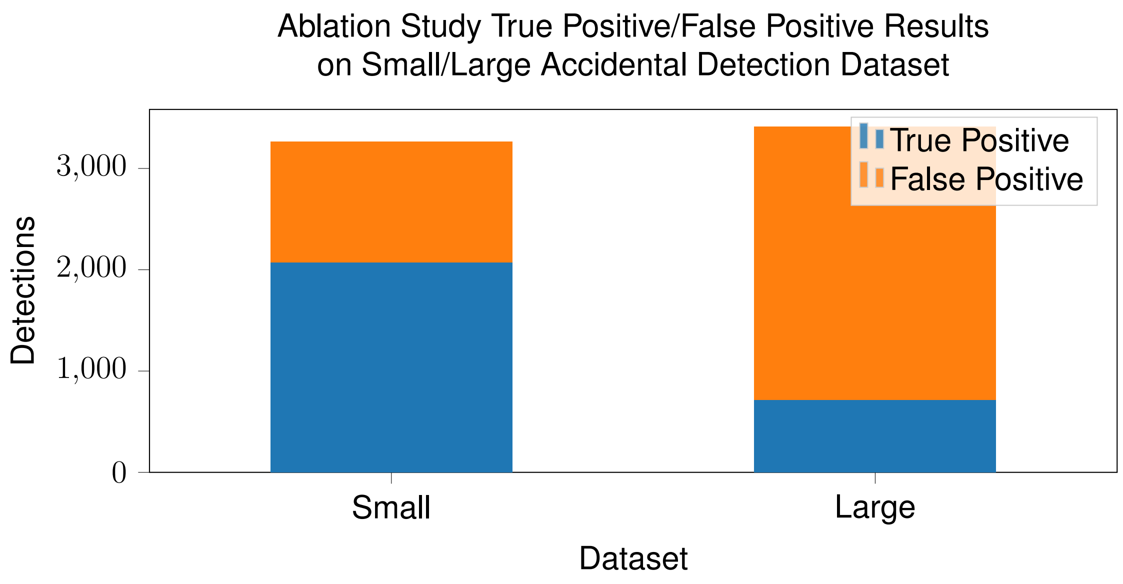


Figure 5.16 – True positive and false positive detection for ablation study on large and small accidental dataset. The large amount of false positive means the detections can not be used for OMR.

Even though the isolated symbol detector also produces a confidence score for each detection prediction, it is hard to find the correct threshold to reject incorrect detection when no ground truth is available. From this study, we can see that the AP measure is only slightly affected by the very bad precision results because it also uses the confidence score to balance the impact of each prediction and is therefore not a very good metric to evaluate our task. Future work could be done to try and guess this confidence score threshold using our small validation dataset presented in section 5.2.2.

5.3.3 Quantitative Results

Small Dataset Results We now present the final results of our Isolating-GAN method. In fig. 5.17, we show the AP, precision and recall of our Isolating-GAN method using the small accidental detection dataset. We can see that we obtain significantly better and consistent results with very compressed box plots and a maximum AP, precision and recall all at or above 95%. However, we can still see a large variation in the precision of the Flat class with 28% difference between the minimum and maximum precision. We can see that our selection mechanism does not select the best run for every class of symbols but, as noted before in section 5.2.2.2, this behavior stems from the fact that we optimize our model selection using the mAP, which is the average of the different symbol class AP. On the other hand, this non-optimal selection is done on very stable runs with very similar results and our selection mechanism is able to select a run producing an acceptable precision for the Flat class showing unstable results. The final chosen model shows an mAP of 94.8%, precision of 95.2% and a recall of 95.3%. We believe these detection results are more than sufficient for the task of detecting music symbols in an OMR pipeline. Although this experiment is not exactly comparable with our previous fully supervised music symbol detection experiment shown in section 2.2.5, we actually obtain results comparable with fully supervised methods, only 4% of mAP behind the best performing R-FCN fully supervised model.

Large Dataset Results For the large accidental detection dataset, the results shown in fig. 5.18 are generally slightly lower with a maximum mAP at 88%. The box plots are generally larger, especially for the Flat class still showing very unstable results across the 10 runs. However, this instability is countered using our early-stopping and model selection mechanism, which is able to select the run that produces the best combination of precision and recall. The chosen run shows a mAP of 82.5%, a precision of 91.4% and a recall 83.2%. We believe the general lower results from the large accidental dataset compared to the small dataset is because of the much lower frequency of symbols to detect in the dataset as shown in table 5.3.

The table 5.4 shows the complete results for our small and large accidental dataset.

Comparison Ablation Results In fig. 5.19 and fig. 5.20, we compare the results between our Isolating-GAN method and the ablation experiments. We can see that, by using our Isolating-GAN, we keep a similar high AP and recall while greatly improving

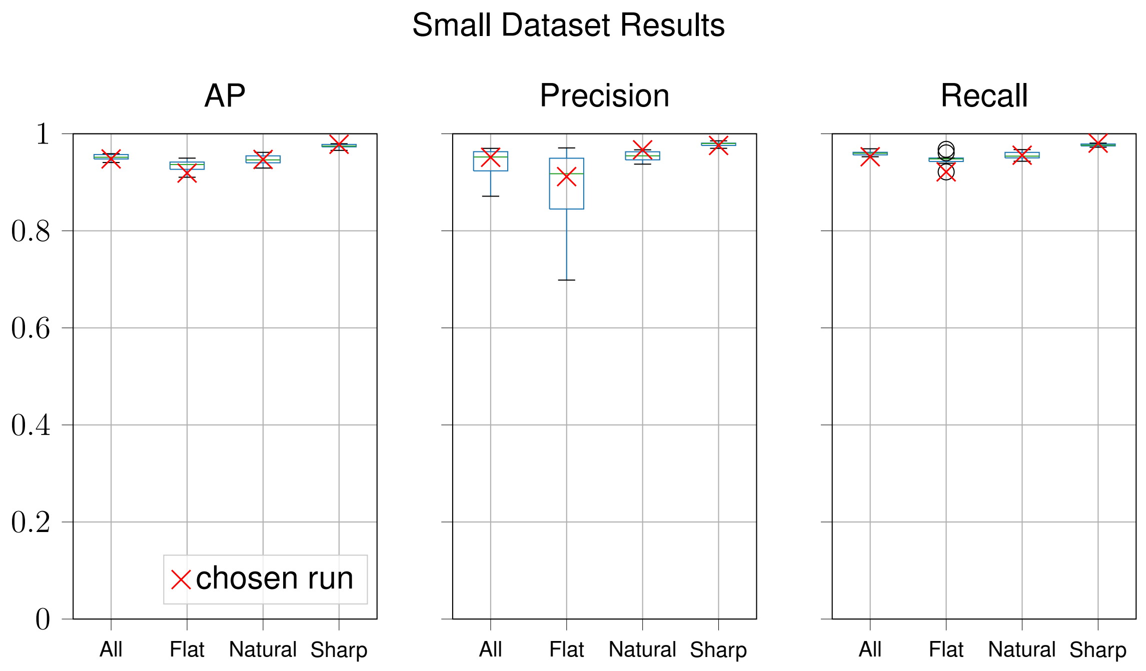


Figure 5.17 – Final results of our unsupervised multi-class Isolating-GAN on our small accidental detection dataset. Our method shows very good AP, precision and recall results with a median over 90%. Only the stability of the precision for the Flat class is a little bit weaker with a difference between the minimum and maximum precision of 28%.

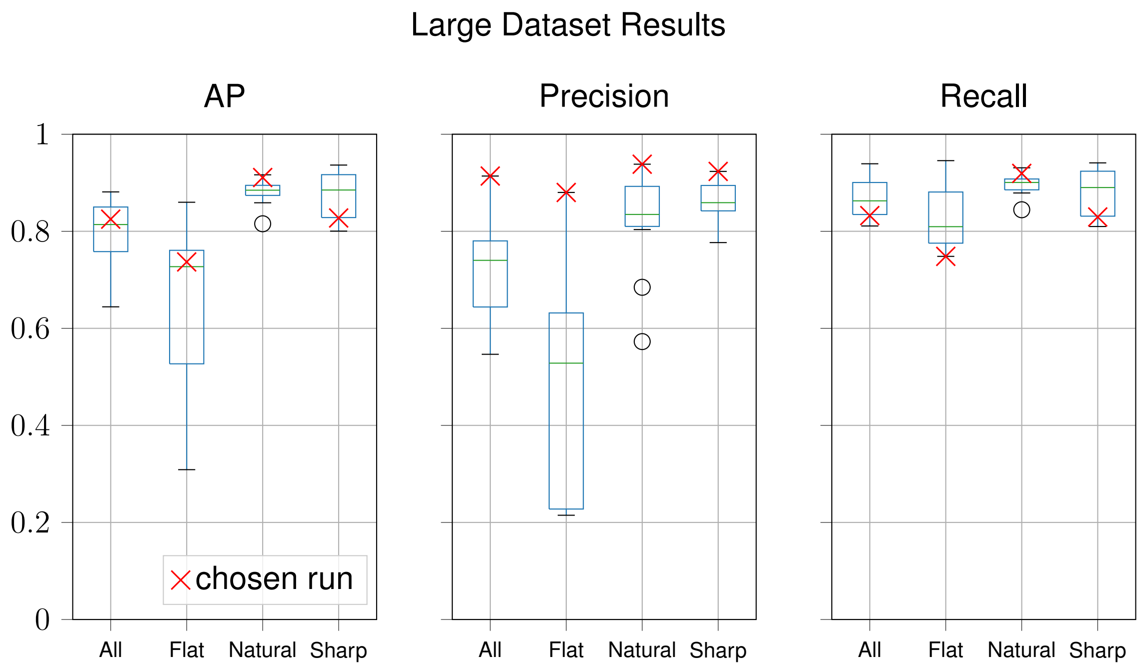
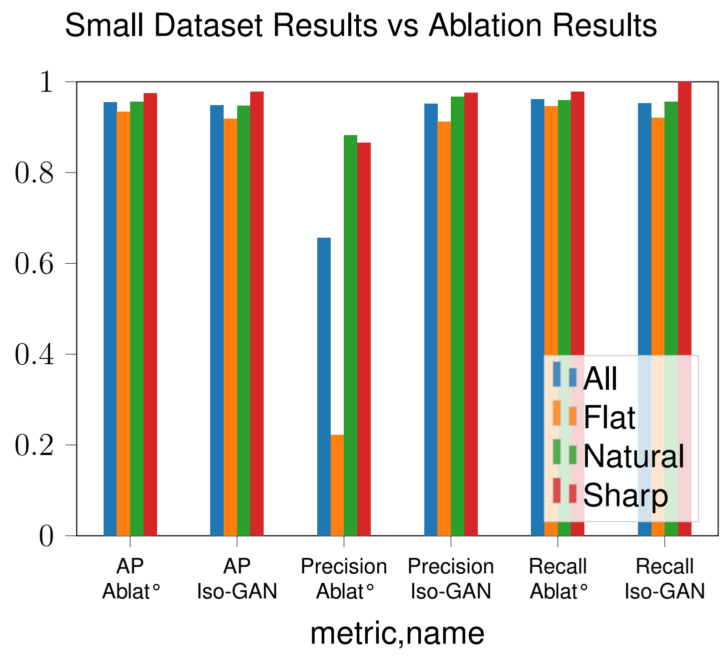


Figure 5.18 – Final results of our unsupervised multi-class Isolating-GAN on our large accidental detection dataset. Even on this challenging dataset, our Isolating-GAN maintains a good maximum mAP of 88%. However, the results are much more unstable. However, this instability is mitigated with our early-stopping and selection mechanism which is actually able to select the model that produces the best precision across the three classes.

Table 5.4 – Detailed final results for the small and large accidental detection dataset. We show the AP, Precision and Recall results for each accidental classes on 10 repeated identical trainings. Green trainings gave the best theoretical mAP results while yellow trainings were chosen using our small validation set. Bold results highlight the best results out of the 10 trainings (over a row).

Dataset	Label	Metric	#0 (%)	#1 (%)	#2 (%)	#3 (%)	#4 (%)	#5 (%)	#6 (%)	#7 (%)	#8 (%)	#9 (%)
Large	Flat	AP	30.9	73.7	**86.0**	44.5	55.6	75.7	51.7	71.7	82.2	76.2
		Precision	21.5	**88.0**	50.2	22.1	21.5	55.5	24.8	62.2	83.0	63.5
		Recall	76.9	74.8	90.5	79.6	**94.6**	89.8	80.3	76.2	83.0	81.6
	Natural	AP	81.5	91.1	88.0	88.9	**91.6**	89.2	87.3	87.6	89.5	85.9
		Precision	68.5	**93.8**	82.9	83.5	57.3	80.4	83.4	89.5	91.0	88.4
		Recall	84.4	**91.9**	89.6	90.5	93.1	90.8	88.5	88.8	90.8	87.9
	Sharp	AP	80.9	82.7	90.3	90.1	**93.6**	92.1	83.1	80.0	92.6	86.9
		Precision	83.9	**92.3**	85.3	86.5	85.2	89.8	88.2	77.7	91.9	83.7
		Recall	82.0	83.0	91.1	90.5	94.1	92.8	83.6	81.0	**93.1**	87.5
	All	mAP	64.4	82.5	88.1	74.5	80.3	85.7	74.1	79.8	**88.1**	83.0
		Precision	57.9	**91.4**	72.8	64.0	54.6	75.2	65.5	76.5	88.6	78.5
		Recall	81.1	83.2	90.4	86.9	93.9	91.1	84.1	82.0	**89.0**	85.7
Small	Flat	AP	93.9	91.0	92.6	94.2	**95.0**	92.9	94.2	93.4	94.6	91.9
		Precision	95.3	83.6	86.6	92.4	**97.1**	69.8	94.0	83.8	96.0	91.2
		Recall	93.9	95.0	94.3	95.0	95.0	**96.8**	94.3	96.1	94.6	92.1
	Natural	AP	94.4	93.9	94.5	**96.2**	92.9	95.6	95.9	93.6	95.0	94.7
		Precision	96.6	95.6	94.9	95.9	95.3	94.5	94.2	93.7	96.4	**96.7**
		Recall	95.2	95.1	95.0	96.5	94.3	96.3	**96.7**	94.9	95.6	95.6
	Sharp	AP	97.7	97.3	97.0	97.3	**97.9**	96.6	97.6	97.3	**97.9**	97.8
		Precision	97.9	97.0	98.2	97.6	**98.5**	97.0	98.1	98.1	98.1	97.6
		Recall	97.7	97.7	97.2	97.5	97.9	97.6	97.6	97.3	97.9	**98.1**
	All	mAP	95.3	94.1	94.7	**95.9**	95.3	95.0	95.9	94.8	95.8	94.8
		Precision	96.6	92.1	93.2	95.3	**97.0**	87.1	95.4	91.9	96.8	95.2
		Recall	95.6	95.9	95.5	96.3	95.8	**96.9**	96.2	96.1	96.1	95.3

the precision. This translates to a great reduction of the number of false-positives from 10 times for the small accidental detection dataset and 20 times for the large accidental detection dataset.



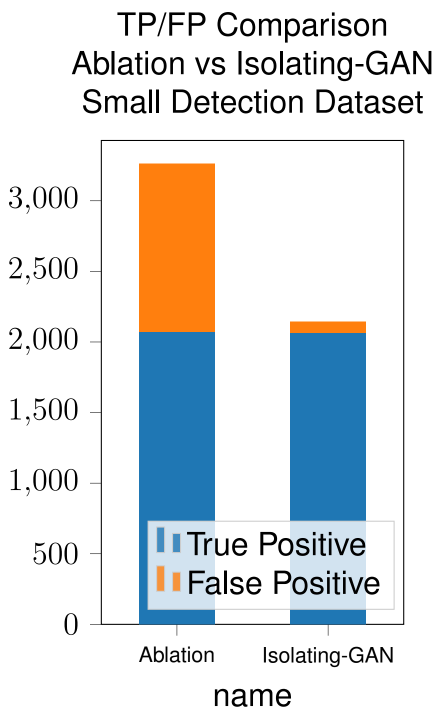


Figure 5.19 – Comparison between our Isolating-GAN model selected using our small validation set and ablation experiment results on our small accidental detection dataset. By using our Isolating-GAN, we improve significantly the precision while keeping a similar AP and a slightly worse Recall. This can be seen concretely by reducing the amount of False Positive from 1,190 to 79.

5.3.4 Qualitative Results

Well Localized Symbols In the following figs. 5.21 to 5.25, we look in details the results for the large accidental detection dataset. In fig. 5.21, we show examples of true positive detection for the three accidental classes and also examples of multi-symbol detections. We can see that the generator of our Isolating-GAN is able to very cleanly separate the information between background and the symbol we want to isolate. This mechanism works even for under-segmented symbols as well as for broken symbols. In some cases, for the natural class in particular, we observed some generation defects

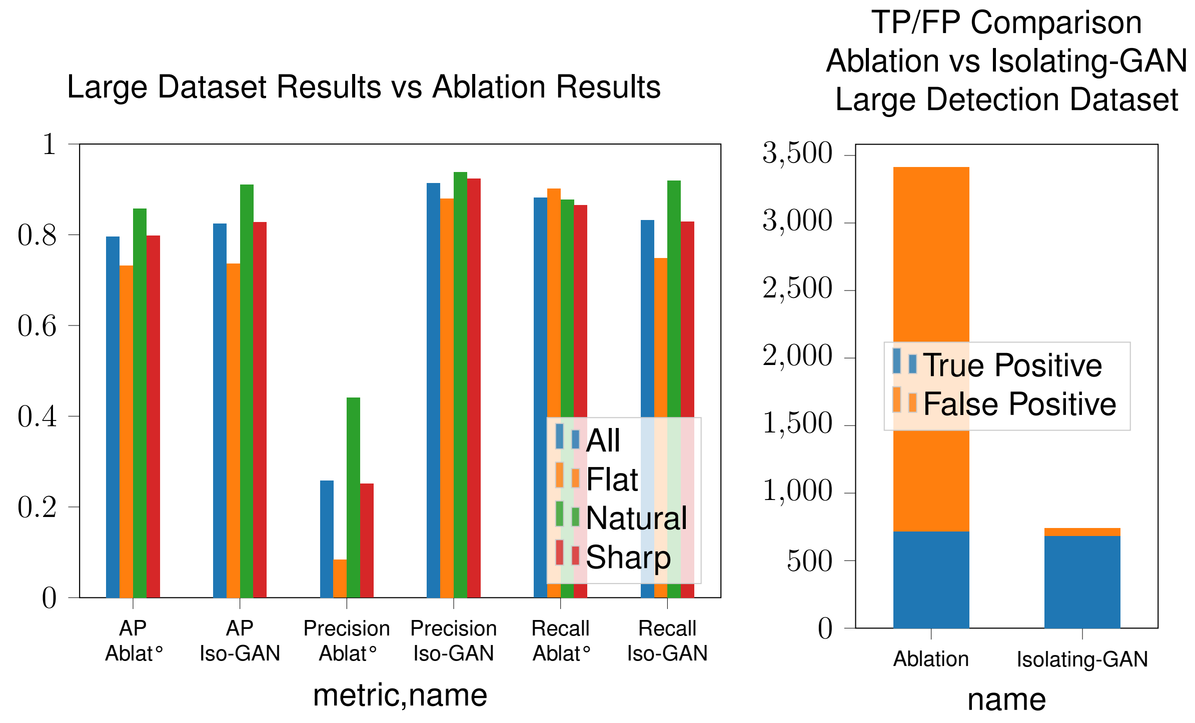


Figure 5.20 – Comparison between our Isolating-GAN model selected using our small validation set and ablation experiment results on our large accidental detection dataset. By using our Isolating-GAN, we improve significantly the precision while keeping a similar AP and Recall. This can be seen concretely by reducing the amount of False Positive from 2,696 to 57.

where straight lines very close to a natural symbol will be kept and lower the precision of the isolated symbol detector. From the multi-symbol detections examples, we can see that our method can actually correctly detect tightly packed symbols with high level of accuracy. The isolated symbol detector is very accurately detecting isolated symbols, even damaged symbols, but can be easily troubled by additional background noises.

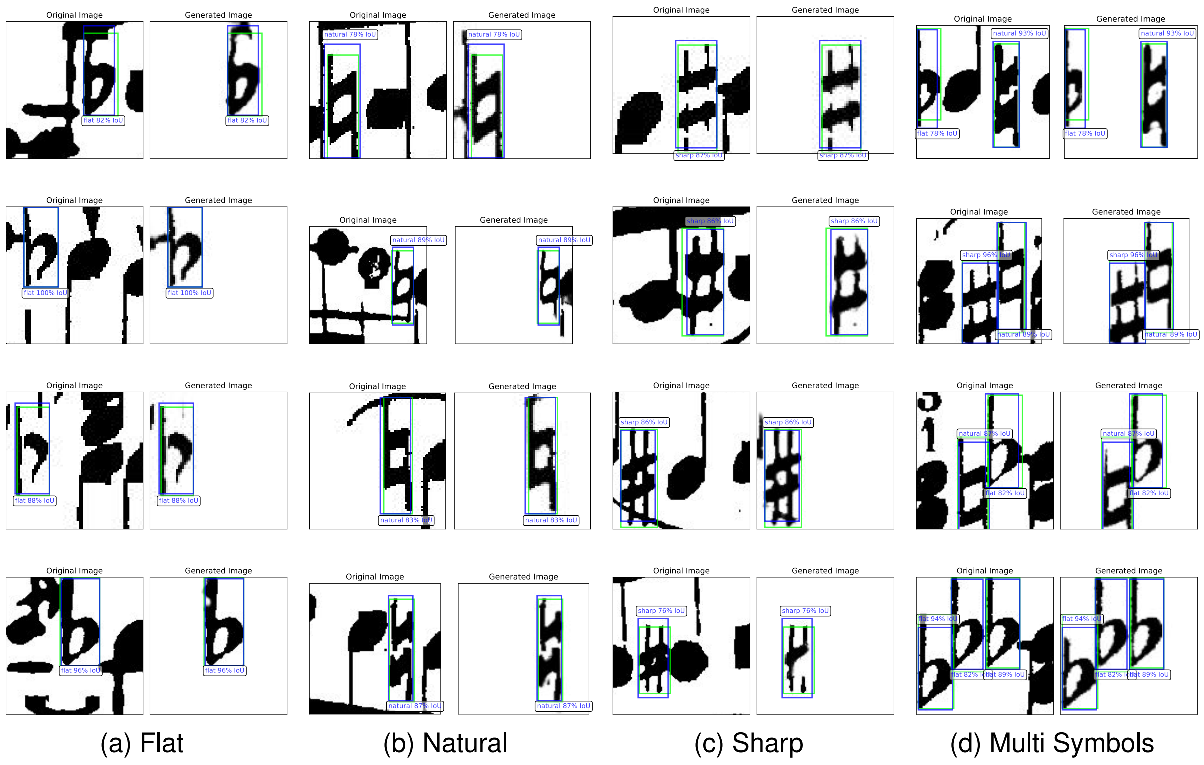


Figure 5.21 – Example of true positive detections. Green boxes are ground truth, blue boxes are detections. We can see that our Isolating-GAN method is able to detect heavily damaged and broken symbols, symbols touching other symbols and multiple symbols at the same time.

Symbols On The Edges One thing we have noticed is that symbols on the edge of the image patch have a significant impact on our metric as shown by table 5.5. Figure 5.22 shows that almost half of the remaining false positives are in fact caused by symbols on the edge of the image, for which the ground truth box were removed because not enough of the symbol area was inside the image patch (less than 75% of their area inside the image). However, the GAN generator managed to keep part of the symbol on the edge and the detector produced a detection box for those symbols parts, leading to false positive detections. While these detections are considered as false positive by our metric, we believe these detections are less problematic than the other false positive detections, because they actually detected a part of a real symbol and those detections could later be corrected by merging with detections on other windows at the page level of the score.

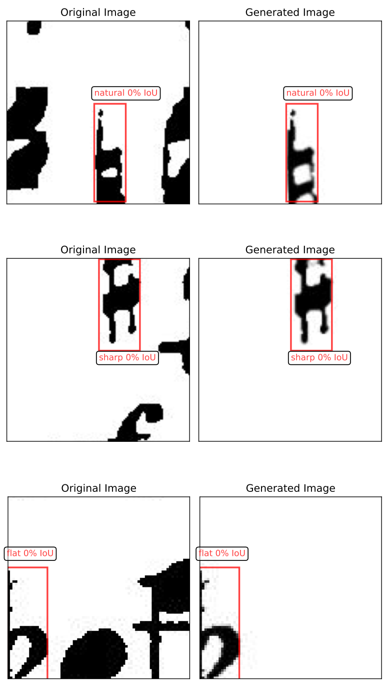
Table 5.5 – Large dataset results of the chosen model using our small validation dataset. We show the detailed results of symbols completely inside the image versus symbols at the edge of the image. Ground truth boxes of symbols at the edge are either kept or discarded if their areas are least 75% inside the image. We can see that almost half of False Positive (FP) are caused by symbol on the edge and one third of missed symbols are also symbol on the edge of the image.

Symbol	Ground truth	TP	FP	Missed
inside the image	Kept	515	29	67
on edge of the image	Kept	167	5	25
on edge of the image	Removed	X	23	X
Total		682	57	92

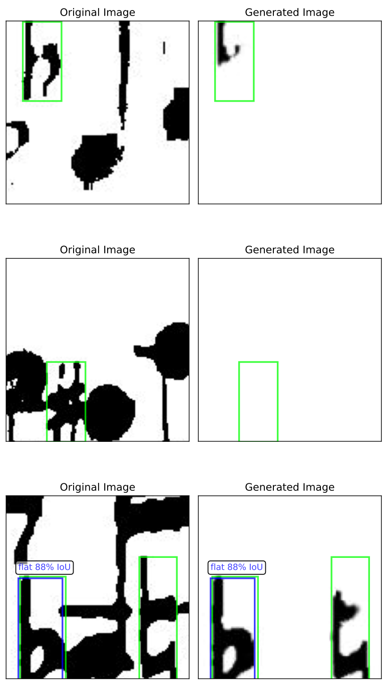
A third of the symbol not found by our detector are also symbols on the edge of the image, which were kept because 75% of their area were inside the image. The ground truth boxes for these symbols were then clipped to the side of the image. For most of these clipped symbols, the generator was not able to correctly isolate the symbol and consequently the detector was not able to detect the symbol.

Missed Symbols Finally, we show some examples of the remaining detection errors. The majority of our errors are missed symbols as shown in fig. 5.23 which are often caused by the GAN generator completely removing the symbol from the image. These symbols often presents heavy degradations or uncommon shapes. We believe these kinds of symbols could later be added to the isolated symbol set to further improve the recall of our method.

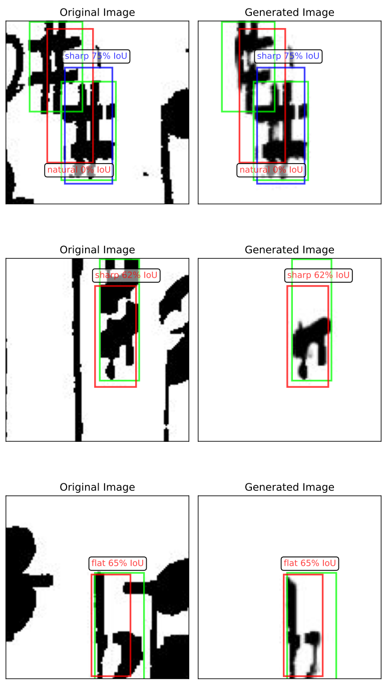
Badly Localized Symbols We have a few cases of bad localization as shown in fig. 5.24, often caused by high density situation where some shapes or combination of shapes can confuse the GAN generator leading to a badly generated symbols. Moreover, since our IoU threshold for classifying a detection as a true positive is very high (75%), the predictions of some symbols with a ground truth slightly less adjusted can be



(a) Removed GT with FP (23)

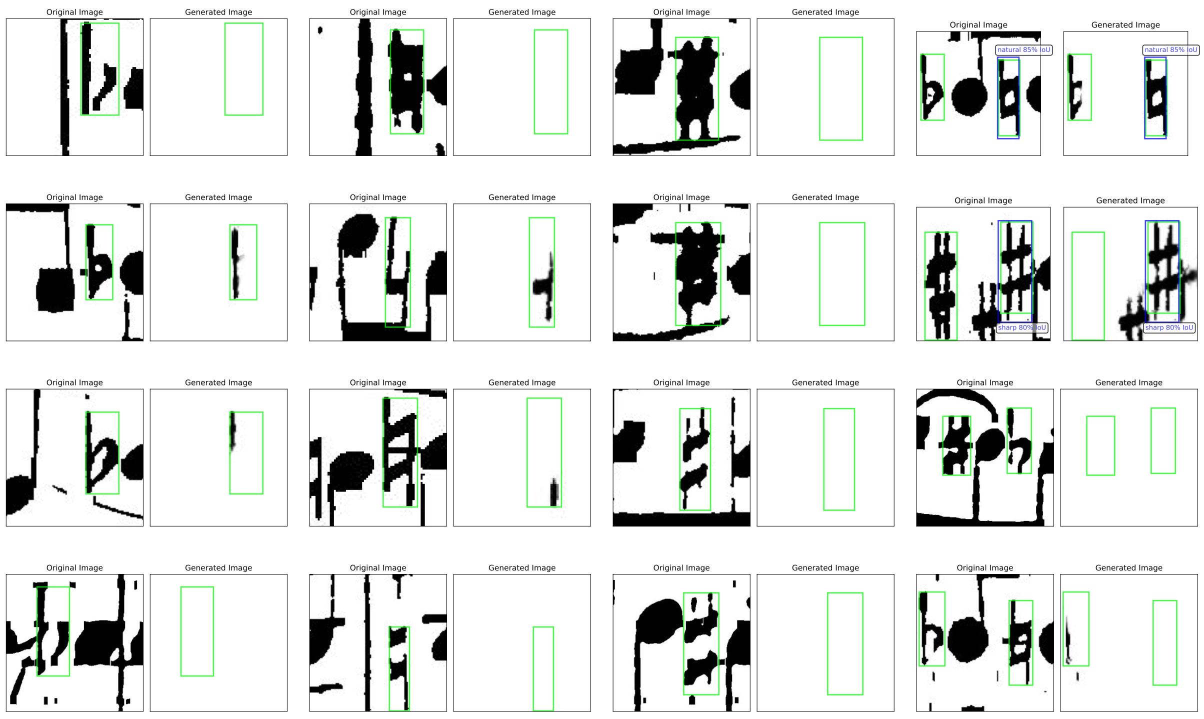


(b) Clipped GT Missed (25)



(c) Clipped GT with FP (5)

Figure 5.22 – Examples of detection errors caused by symbols on the edge of the image. Green boxes are Ground Truth (GT), blue boxes are true positive detection, red boxes are False Positive (FP) detections. Symbols on the edge of the image can cause problems with almost half of FP detections caused by symbols on the edge of the image. 27% of missed symbols are also symbols clipped on the edge of the image.



(a) Flat (b) Natural (c) Sharp (d) Multi-Symbols

Figure 5.23 – Symbols missed excluding symbols on the edge of the image. Green boxes are ground truth. The largest set of detections errors are missed symbols with 67 symbols often caused by heavily damaged symbols or symbols with uncommon shapes.

classified as false positive, even though the quality of these detections is sufficient for later OMR tasks. Heavily damaged symbols where only half of the symbol is present in the original image is also source of confusion, because the ground truth box only frame the remaining parts of the symbol, while the detector tries to frame the original symbol shape.

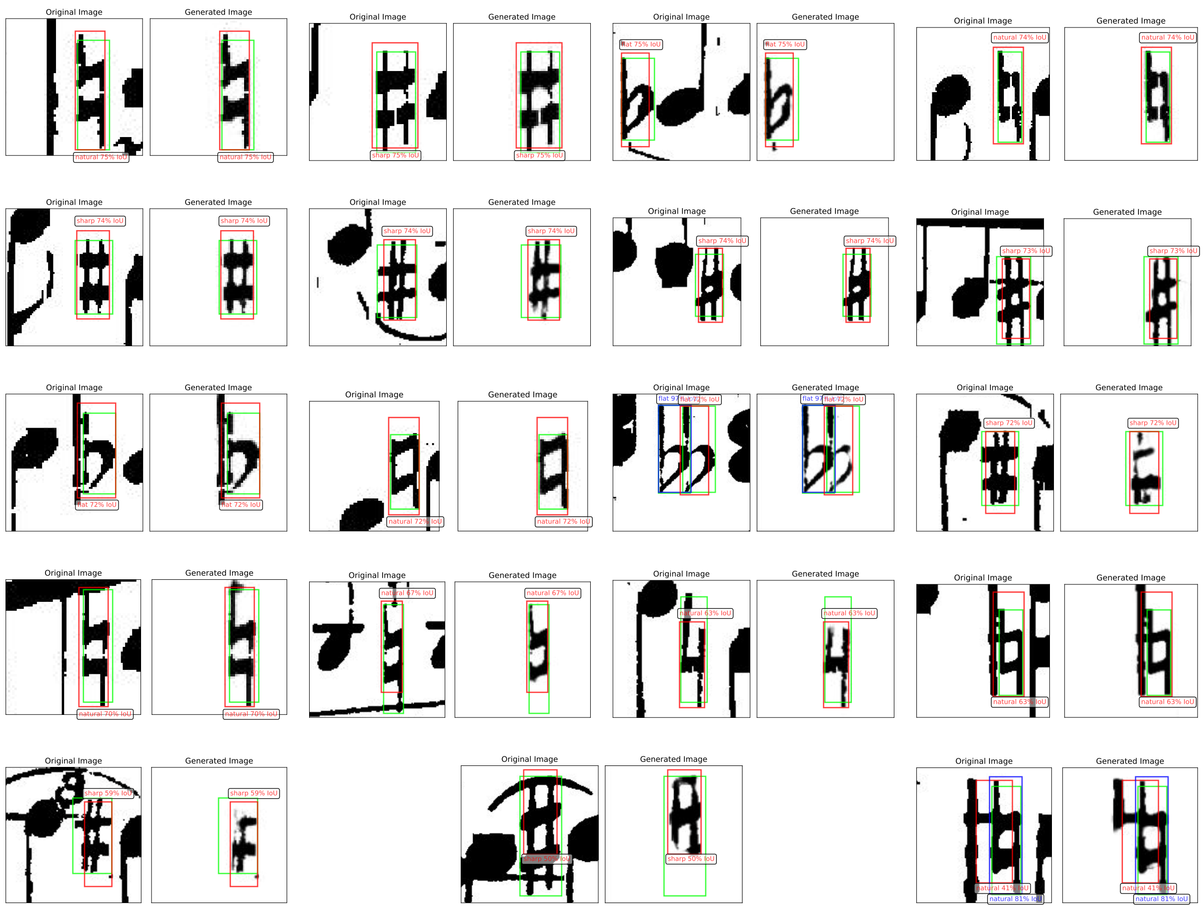


Figure 5.24 – Bad localization errors excluding symbols on the edge of the image. Green boxes are ground truth, blue boxes are true positive and red boxes are false positive. Bad localization with insufficient IoU with 19 symbols are often caused by the background not correctly removed.

Hallucinations Finally, we have some rare cases of hallucination by the GAN generator as shown in fig. 5.25, trying to produce a symbol out of background symbols or shapes. These hallucinations are a side effect of our method training objectives, where we only use the shape of isolated symbols with no additional context to isolate symbols in real images. These hallucinations could later be mitigated by adding negative examples shapes in the isolated symbol detection dataset generation process. Moreover, later music notation reconstruction step using our grammatical method would be used to rule out these out-of-place detections.

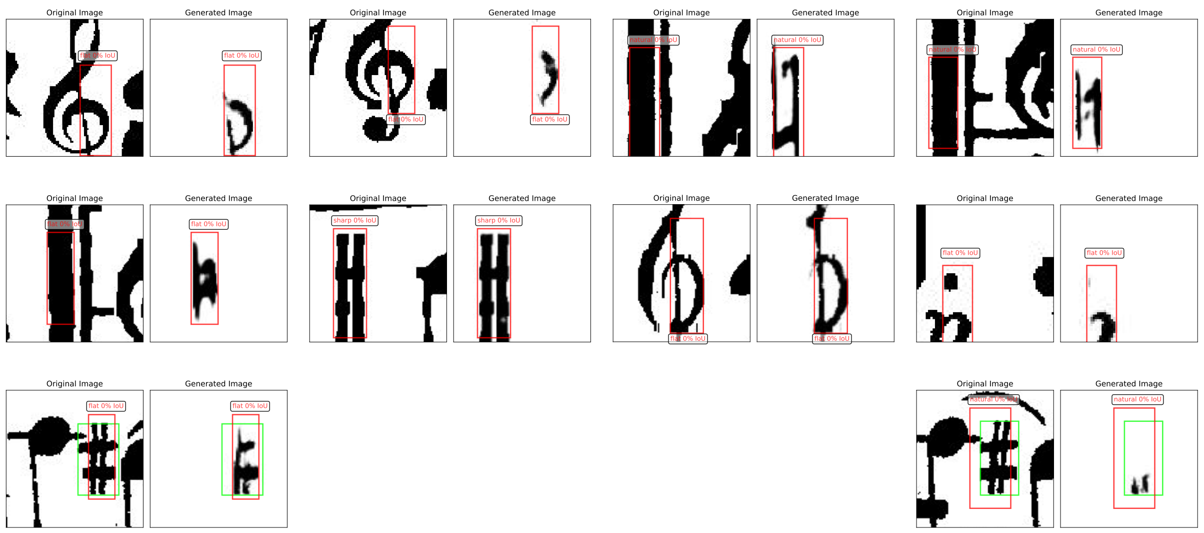


Figure 5.25 – Hallucinations and class confusions. Red boxes are false positive. There are still some rare case of hallucination by the GAN generator where the generator is trying to synthesize a symbol from the background.

Overall, we have presented in details the qualitative results on the large accidental dataset. We have shown the high quality of the true positive detection obtaining an IoU score of over 75%. We have shown for false positive detection and missed symbols, the significant influence of symbols on edge, representing 28% of all the detection or missed detection counted as errors by our metric. Knowing that we only annotated 20 pages out of the 1,812 pages constituting the dataset, we used our best trained Isolating-GAN detector to detect symbols on most of the remaining dataset of 1,774 pages and detected 38,908 symbols. Out of these 38,908 symbols, we therefore estimate that approximately 3,000 symbols (7.7% of all detections) are false positives, of which 1,500 symbols (50% of the false positives) are symbols on the edges which would be corrected by the music grammar at the page level. We also estimate that we missed around 4,624 symbols (11% of the true positive + missed symbols), of which 1,256 symbols (27% of the missed symbols) would be on the edge and could be later found at the page level. In order to detect the remaining 3,400 missed symbols, another fully supervised detector could be trained using the found symbols by our Isolating-GAN while not being affected by missed symbols and be able to better detect accidental symbols.

5.4 Conclusion

In this chapter, we have shown an extensive set of experiment demonstrating how our method was developed and evaluated.

In section 5.2, we have shown the successive experiments that helped us design our method. Starting from a very simple experimental setting with a single class of symbol to detect in a small filtered dataset, we have shown how the use of an additional training objective and the use of additional negative examples improves the training of our Isolating-GAN. Moreover, because of the flexibility of our method, we propose a multi-class version of our method, simply by changing the isolated symbol detection dataset with no change to the model itself.

Because of the unsupervised nature of our method and the inherent instability of our training, it is essential to be able to use our method in a situation with no ground truth at hand for evaluation. As a solution, we propose to use a small validation dataset to evaluate our method during training and enabling an early stopping mechanism. This small validation dataset can also be used after the training of multiple models, to choose the best performing model.

Finally, in section 5.3, we compare the use of our method on two different accidental detection datasets, one small dataset used throughout the chapter and a larger detection dataset, reflecting a more realistic use case of our method with a much harder detection task because of the lower frequency of symbols to detect. We compare those results with an ablated experiment where no GAN generator is used to isolate symbols prior the detection step. The comparison shows that the use of our Isolating-GAN generator is having a huge impact in the reduction of the number of false positive, improving the precision of our method significantly. Overall, we show detection performance of 94.8% mAP on the small detection dataset, which is very close to the performance of fully supervised detector presented in chapter 2. Our method produces a still acceptable detection performance of 82.5% mAP on the significantly harder large detection dataset. In order to complete this whole set of experiments, a total of 102 different hyperparameter combinations were explored for which 810 trainings had to be done and took around 2.5 months of pure training time.

After a detailed analysis of the results, we believe we have shown the robustness and accuracy of our method while identifying the remaining challenges of our method which we discuss in the next chapter of this manuscript.

CHAPTER 6

ISOLATING-GAN USAGE AND FUTURE WORKS

6.1 Introduction

We have now presented the entirety of the work accomplished during this thesis. Starting from the study of fully supervised music symbol detectors in chapter 2, we introduced a new unsupervised music symbol detection method that we called Isolating-GAN in chapter 4 combining a generative model and a detector model.

In chapter 5, we presented the whole set of experiments that guided our method design together with an extensive evaluation of our method relative to important hyperparameters and symbol detection datasets of varying difficulty. In these experiments, we demonstrated the high precision of the detections produced by our method and showed that our method can approach very closely the detection quality of fully supervised detectors without using any manually annotated ground truth.

Nonetheless, much work is still needed to improve the quality of our method, which we discuss in the next section

6.2 Improving Isolating-GAN

Improving Training Objectives While our multi-loss objectives for the Isolating-GAN training combining the classical GAN losses and our image reconstruction loss is able to train our GAN model, the implementation could be simplified by combining the adversarial loss and the reconstruction loss into a single loss using a single loss balancing factor. This would reduce the amount of hyperparameters to tune, reduce training time and simplify the implementation and hyperparameters tuning of our model.

Training Instability The most problematic aspect of our method is the instability of the training, producing models with large variation in the generation quality while being trained with the exact same hyperparameters. In this work, we mitigate this undesirable aspect by training multiple models with the same hyperparameters and choosing the best performing model using a small validation dataset. However, this strategy is very wasteful in computation effort, multiplying the training time by the number of duplicated model.

The search for stability of GAN models was already discussed by the literature and presented in section 1.6.4. GAN architectures like the Wasserstein GAN, aims to stabilize the training using an improved training objective and some architectural changes guarantying non-zero gradients and therefore avoiding gradient vanishing effects. Another approach to improve stability is discussed by Karras et al. [Karras 18] where the GAN architecture is trained layer by layer while also using statistics of the minibatch training data to guaranty that the generated data has the same variation as the real data to mimic. However, it is not clear if the increase in stability of our method will correlate with improved detection results. The fact is that the actual training objective of the GAN, which is to transform images extracted from real historical printed music scores into images containing only isolated symbols on a white background, is not the same as our symbol isolating task we want the GAN generator to accomplish. The differences in symbols size and shape between the real historical printed scores and the isolated symbol dataset leads to irreconcilable differences, and it is yet to be proved that an actual stable training can be done using an adversarial learning strategy.

Given the unknowns of using yet another GAN architecture, we kept in this work the use of a classical GAN architecture and propose to explore the use of newer and possibly stabler GAN model in future works.

Another path to improve the stability of our training method would be to better regroup our training data and presenting a more uniform corpus of real data during the GAN training. This uniformity would simplify the image-to-image translation task of the generator and therefore improve the generation quality and detection quality of our method. However, this approach have the main drawback of increasing the computation cost of our method because of the retraining of the GAN for each new corpus of music scores.

Larger Symbol Class Set During this work, we exclusively worked on a reduced accidental class set with three accidental symbol class: Flat, Natural and Sharp. Now that we have shown that our method can produce highly precise detection on a relatively small class set of symbols, we believe that our method can be adapted to a much larger class set of symbols, including note heads, flags, attack signs, rests, clefs. . . To adapt to new symbol classes, the first step of our method explained in section 4.3 will have to be adapted but thanks to the flexibility of the grammatical description of the musical notations, these changes will be simple and straightforward.

We also believe that our method could be use for symbol recognition in handwritten music scores. However, the variability of each writer would have to be taken into account, maybe by using isolated symbol examples of the same writers. Moreover, we believe that any kind of structured documents with segmentation problems such as electrical circuit design documents could be the target of our method.

Larger Image Size For now, we only have tested our method with a relatively small image patches of 128×128 pixels. We started with a small image because of the known difficulty to train a generative model with large images. However, the literature has now shown that GAN model can scale to larger images and produce high resolution images.

Future work could be done to improve this aspect of our method coincidentally reducing the effect of symbols on the edge of images, which we found was a significant cause for detection errors. Larger images also mean that fewer images can be used to evaluate a whole page of document, reducing the total time needed to process music score page.

Another approach to expand the use of our method to a window larger than 128×128 pixels would be to process the entire page of a music score with the trained generator using a sliding window method before doing the detection stage. This in turn would resolve the problem of symbols on the edge for the generator, although we would still have to find a strategy to apply the detector on the entire page of music score. From preliminary testing, we found out that the generator does generalize well to unseen background shapes during training and manage to remove most of the music score background while keeping the relevant symbols to detect. In our specific study of accidental symbols, we found out that after some preliminary testing that while we trained our generator on examples of accidentals always attached to a note head, the generator was able to correctly filter and isolate accidentals used in the key signature.

However, by using a sliding window method with a stride smaller than the window itself and adding the intensity of resulting images, we intensified all responses of the generator, the correct isolating behavior of the generator as well as the incorrect hallucination behavior of the generator.

6.3 Autonomous Symbol Detection System

Taking a step back, we believe our work is a first step toward designing an entirely autonomous symbol detection system, where no or very few ground truth examples are needed to apply our detection framework to a new type of documents. By only using isolated symbols for the training of the detection model, we are able to bootstrap the use of data hungry Deep Learning model while maintaining sufficiently accurate symbol detection results. Even if the detection results are not accurate enough, we hypothesize that the information gathered using our Isolating-GAN method could be used in a second stage training using a fully supervised Deep Learning detector of high precision such as a Faster R-CNN.

Finally, we could entirely automatize the full recognition of a new corpus of music scores using a syntactical method as the DMOS method, where the parsing of the music notation could be done in multiple stages, with each stages concentrating on a few symbol classes and using previously detected symbols. Each stages would produce relevant RoIs to be used by our Isolating-GAN method together with a few isolated symbol examples.

CONCLUSION

We believe the recognition of historical printed music scores is essential to ease the use and study of such scores by musicians and musicologist. However, the difficulty of the detection of music symbols and reconstruction of the music notation caused by the high density of music symbols, the complexity of the music notation and degradations due to printing methods and age of the document has prevented the use of traditional OMR method. With the introduction of new Deep Learning-based computer vision model, we show in this work how such Deep Learning models can be beneficial for the early graphical recognition OMR step that is music symbol detection.

During this work, we studied the detection of music symbols in the context of Optical Music Recognition of historical printed music scores. In chapter 1, we gave an overview of what is a music score, followed by a presentation of the different steps that constitute an OMR system. We especially focused on thoroughly presenting the task of music symbol detection since this is the main task we are interested in this work. We then propose to explore the various Deep Learning model that will help us achieve our music symbol detection task, presenting both fully supervised detection models such as the Faster R-CNN as well as generative models that can be used in for training unsupervised tasks.

Chapter 2 presents the beginning of our work where we present the use of state-of-the-art Deep Learning detection model as well as a custom original detection model for a small and focused accidental detection task. For this task, we show that state-of-the-art detectors can produce very good detection results such as 98.73% of mAP with an IoU threshold of 75%. Therefore, we broaden the scope of the detection task on applying state-of-the-art detectors on a much more complicated and difficult handwritten music score dataset: MUSCIMA++. Even on such a broad and complicated dataset, we also show that the detection results can be very good at 80% of mAP. However, one of the main remaining difficulties in a general music symbol detection task is to account for the imbalance in the frequency of symbol classes to detect.

While we have shown that Deep Learning-based detectors can produce highly accurate detection of music symbols, we discussed in chapter 3 the impact of manual annotations in the process of training such Deep Learning methods. Indeed, the production of such manual annotations are very costly and slow to produce. In the document recognition community, this bootstrapping problematic of producing ground truth has often been solved by generating synthetic data, often involving complex generation procedures. For music scores, one could use music typesetting software to produce almost an infinite amount of varying music scores. However, there is no guaranty that trained music symbol detection model on synthetic scores would be able to perform as well on real historical printed music scores since a lot of variation that we can see in historical music scores due to the engraving techniques and age of the score are not taken into account by typesetting software. Therefore, we propose a hybrid method of using a small amount of synthetic data generated using only isolated music symbols on a white background and a generative method able to transform real images of historical music scores into a simpler graphical representation.

In order to avoid as much as possible the use of costly manual annotations, we proposed in chapter 4 our new Isolating-GAN method for unsupervised music symbol detection. Our new method consists of three iterative steps that gradually simplify the tasks without ever needing manually annotated ground truth for symbol detection:

1. Step 1: Identify Region of Interests
2. Step 2: Isolate music symbols using Isolating-GAN
3. Step 3: Detect isolated music symbols

At the heart of the method, we use isolated music symbols to create a simplified domain of representation where isolated symbols are printed in a blank image at varying position and size. This simplified representation allows us to pretrain a fully supervised detector on this trivial detection task and transform real image of historical scores into generated images containing only symbols to detect in an empty background. Then, we also discussed how we can use and evaluate our method in a real use setting. The goal of our method is to be used in a situation where no annotated data is available to evaluate our model. Therefore, we design an evaluation method using only a few manually annotated examples and augmented using a simple bootstrapping technique.

In chapter 5, we demonstrated the effectiveness and robustness of our new method by presenting an extensive set of experiments evaluating the development of our method and the accuracy of the method on two music symbol detection datasets of varying difficulty. The first small accidental dataset, we limited the difficulty of the task by artificially removing and later limiting images without symbols to detect. While the training of our method is done entirely without manually annotated data, the manually produced ground truth of this small dataset allowed us to develop, evaluate and tune our method architecture, as well as design an effective way to tune our method with almost no ground truth. For the second larger accidental dataset, we constituted a 100 times larger dataset and annotated 20 pages of the dataset for evaluation. This dataset constitutes a much more challenging detection task, since the dataset is larger, more heterogeneous and we did not restrict the amount of images with no symbols to detect. In order to synthesize our simplified representation of isolated symbols in a blank canvas, we use a small isolated symbol dataset of 541 symbols manually annotated with class label information. With our approach, we obtained a mAP of 94.8% on the small simplified dataset and a mAP of 82.5% on the large difficult dataset for a detection task with three accidental classes. We also demonstrated that using the Isolating-GAN to filter and isolate symbols before the detection operation reduces the number of false positives from 2,696 to 57 on the large dataset. We bootstrapped the annotations of our large difficult dataset by applying our method on 1,774 pages of historical music scores and detecting automatically 3,8908 new accidentals. In order to complete this whole set of experiments, a total of 102 different hyperparameter combinations were explored for which 810 trainings had to be done and took around 2.5 months of pure training time.

Finally, in chapter 6, we discussed various improvements of our method that could be made in the future. Indeed, much work is still needed to improve the training efficiency and stability of our method. We believe that new GAN models from the literature such as the Wasserstein GAN could help stabilizing our training methodology. We also plan to expand our method to a larger class set, larger image sizes but also to other kind of structured documents such as handwritten music scores or even electrical circuit design documents in order to reduce the computational cost and generalize the application of our method.

From our very focused study of the task of detecting music symbols, we believe that this work is only the first step towards for the exploration of unsupervised detection methods. Moreover, many fields of application, especially niche types of documents with less manual annotations efforts, can benefit from the use of our unsupervised detection method. Finally, entirely autonomous systems could be designed using our method that can adapt the model to detect entirely new class of symbols using no ground truth and only isolated symbols. This system could be bootstrapped in successive recognition stages by a syntactical method such as DMOS which would be able to construct layer by layer the syntactical structure of the document.

PERSONAL PUBLICATIONS

[Choi 17] Kwon-Young Choi, Bertrand Couasnon, Yann Ricquebourg, and Richard Zanibbi, « Bootstrapping Samples of Accidentals in Dense Piano Scores for CNN-Based Detection », in: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, Kyoto, Japan: IEEE Computer Society, Nov. 1, 2017, pp. 19–20, ISBN: 978-1-5386-3586-5, DOI: `10.1109/ICDAR.2017.257` (cit. on p. 65).

[Choi 18a] Kwon-Young Choi, *Accidental Detection Dataset – Équipe Intu-iDoc*, Mar. 2018, URL: `https://www-intuidoc.irisa.fr/en/choi_accidentals/` (visited on 08/13/2018) (cit. on pp. 14, 37, 55, 96, 117, 148).

[Choi 18b] Kwon-Young Choi, Bertrand B. Coüasnon, Yann Ricquebourg, and Richard Zanibbi, « Music Symbol Detection with Faster R-CNN Using Synthetic Annotations », in: *1st International Workshop on Reading Music Systems*, Paris, France, Sept. 2018, pp. 9–10 (cit. on p. 82).

[Choi 19] Kwon-Young Choi, Bertrand Couasnon, Yann Ricquebourg, and Richard Zanibbi, « CNN-Based Accidental Detection in Dense Printed Piano Scores », in: *15th International Conference on Document Analysis and Recognition (ICDAR)*, Sydney, Australia: IEEE Computer Society, Sept. 1, 2019, pp. 473–480, ISBN: 978-1-72813-014-9, DOI: `10.1109/ICDAR.2019.00082` (cit. on p. 65).

[Pacha 18b] Alexander Pacha, Kwon-Young Choi, Bertrand Coüasnon, Yann Ricquebourg, Richard Zanibbi, and Horst Eidenberger, « Handwritten Music Object Detection: Open Issues and Baseline Results », in: *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, Vienna, Austria: IEEE, Apr. 2018, pp. 163–168, ISBN: 978-1-5386-3346-5, DOI: `10.1109/DAS.2018.51` (cit. on pp. 14, 32, 75).

BIBLIOGRAPHY

[Ares Oliveira 18] Sofia Ares Oliveira, Benoit Seguin, and Frederic Kaplan, « dhSegment: A Generic Deep-Learning Approach for Document Segmentation », in: *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, Aug. 2018, pp. 7–12, DOI: 10.1109/ICFHR-2018.2018.00011 (cit. on p. 43).

[Arjovsky 17a] Martin Arjovsky and Léon Bottou, « Towards Principled Methods for Training Generative Adversarial Networks », in: Palais des Congrès Neptune, Toulon, France, Jan. 17, 2017, arXiv: 1701.04862 [cs, stat] (cit. on p. 50).

[Arjovsky 17b] Martin Arjovsky, Soumith Chintala, and Léon Bottou, « Wasserstein GAN », in: (Jan. 26, 2017), arXiv: 1701.07875 [cs, stat] (cit. on pp. 17, 50).

[Baumann 95] S. Baumann, « A Simplified Attributed Graph Grammar for High-Level Music Recognition », in: *Proceedings of 3rd International Conference on Document Analysis and Recognition*, vol. 2, Aug. 1995, 1080–1083 vol.2, DOI: 10.1109/ICDAR.1995.602096 (cit. on p. 32).

[Bitteur] Hervé Bitteur, *Audiveris Pages*, URL: https://audiveris.github.io/audiveris/ (visited on 11/02/2020) (cit. on p. 35).

[Borji 19] Ali Borji, « Pros and Cons of GAN Evaluation Measures », in: *Computer Vision and Image Understanding* 179 (Feb. 1, 2019), pp. 41–65, ISSN: 1077-3142, DOI: 10.1016/j.cviu.2018.10.009 (cit. on p. 112).

[Brock 18] Andrew Brock, Jeff Donahue, and Karen Simonyan, « Large Scale GAN Training for High Fidelity Natural Image Synthesis », in: Sept. 27, 2018 (cit. on p. 90).

[Byrd 15] Donald Byrd and Jakob Grue Simonsen, « Towards a Standard Testbed for Optical Music Recognition: Definitions, Metrics, and Page Images », in: *Journal of New Music Research* 44.*3* (July 3, 2015), pp. 169–195, ISSN: 0929-8215, DOI: 10.1080/09298215.2015.1045424 (cit. on p. 35).

[Calvo-Zaragoza 17] Jorge Calvo-Zaragoza, Gabriel Vigliensoni, and Ichiro Fujinaga, « Staff-Line Detection on Grayscale Images with Pixel Classification », in: *SpringerLink*, Springer, Cham, June 20, 2017, pp. 279–286, DOI: 10.1007/978-3-319-58838-4_31 (cit. on p. 31).

[capela 20] capela, *Capella-Scan - Scan Notes with Text, Open PDF, Run Recognition and Edit in Capella - Capella-Software AG (English)*, Nov. 2, 2020, URL: https://www.capella-software.com/us/index.cfm/products/capella-scan/info-capella-scan/ (visited on 11/02/2020) (cit. on p. 35).

[Cardoso 09] J. dos Santos Cardoso, A. Capela, A. Rebelo, C. Guedes, and J. Pinto da Costa, « Staff Detection with Stable Paths », in: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.*6* (June 2009), pp. 1134–1139, ISSN: 1939-3539, DOI: 10.1109/TPAMI.2009.34 (cit. on p. 31).

[Carter 92] Nicholas P. Carter and Richard A. Bacon, « Automatic Recognition of Printed Music », in: *Structured Document Image Analysis*, ed. by Henry S. Baird, Horst Bunke, and Kazuhiko Yamamoto, Berlin, Heidelberg: Springer, 1992, pp. 456–465, ISBN: 978-3-642-77281-8, DOI: 10.1007/978-3-642-77281-8_21 (cit. on p. 31).

[Chu 17] Casey Chu, Andrey Zhmoginov, and Mark Sandler, « CycleGAN, a Master of Steganography », in: Dec. 8, 2017 (cit. on p. 100).

[Coüasnon 01] Bertrand Coüasnon, « DMOS : A Generic Document Recognition Method, Application to an Automatic Generator of Musical Scores, Mathematical Formulae and Table Structures Recognition Systems », in: *Sixth International Conference on Document Analysis and Recognition, 2001. Proceedings*, 2001, pp. 215–

220, DOI: 10.1109/ICDAR.2001.953786 (cit. on pp. 12, 29, 38, 91).

[Dai 16] jifeng Dai, Yi Li, Kaiming He, and Jian Sun, « R-FCN: Object Detection via Region-Based Fully Convolutional Networks », in: *Advances in Neural Information Processing Systems 29*, ed. by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Curran Associates, Inc., 2016, pp. 379–387 (cit. on p. 41).

[dAndecy 94] V. P. d'Andecy, J. Camillerapp, and I. Leplumey, « Kalman Filtering for Segment Detection: Application to Music Scores Analysis », in: *Proceedings of 12th International Conference on Pattern Recognition*, vol. 1, Oct. 1994, 301–305 vol.1, DOI: 10.1109/ICPR.1994.576283 (cit. on p. 31).

[Deng 09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, « ImageNet: A Large-Scale Hierarchical Image Database », in: *2009 IEEE Conference on Computer Vision and Pattern Recognition*, June 2009, pp. 248–255, DOI: 10.1109/CVPR.2009.5206848 (cit. on pp. 44, 67, 102).

[Everingham 10] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman, « The Pascal Visual Object Classes (VOC) Challenge », in: *International Journal of Computer Vision* 88.*2* (June 1, 2010), pp. 303–338, ISSN: 0920-5691, 1573-1405, DOI: 10.1007/s11263-009-0275-4 (cit. on pp. 35, 42, 62, 70, 113, 147).

[Fahmy 93] Hoda Fahmy and Dorothea Blostein, « A Graph Grammar Programming Style for Recognition of Music Notation », in: *Machine Vision and Applications* 6.*2* (Mar. 1, 1993), pp. 83–99, ISSN: 1432-1769, DOI: 10.1007/BF01211933 (cit. on p. 32).

[Fornés 06] Alicia Fornés, Josep Lladós, and Gemma Sánchez, « Primitive Segmentation in Old Handwritten Music Scores », in: *Graphics Recognition. Ten Years Review and Future Perspectives*, ed. by Wenyin Liu and Josep Lladós, vol. 3926, Lecture Notes in Computer Science, Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 279–290, ISBN: 978-3-540-34712-5, DOI: 10.1007/11767978_25 (cit. on p. 31).

[Fornés 08] Alicia Fornés, Josep Lladós, and Gemma Sánchez, « Old Handwritten Musical Symbol Classification by a Dynamic Time Warping Based Method », in: *Graphics Recognition. Recent Advances and New Opportunities*, ed. by Wenyin Liu, Josep Lladós, and Jean-Marc Ogier, Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2008, pp. 51–60, ISBN: 978-3-540-88188-9, DOI: 10.1007/978-3-540-88188-9_6 (cit. on p. 37).

[Fornés 12] 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 », in: *International Journal on Document Analysis and Recognition (IJDAR)* 15.*3* (Sept. 1, 2012), pp. 243–251, ISSN: 1433-2833, 1433-2825, DOI: 10.1007/s10032-011-0168-2 (cit. on pp. 14, 35, 68).

[Fornés 13] Alicia Fornés, Anjan Dutta, Albert Gordo, and Josep Lladós, « The 2012 Music Scores Competitions: Staff Removal and Writer Identification », in: *Graphics Recognition. New Trends and Challenges*, ed. by Young-Bin Kwon and Jean-Marc Ogier, Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2013, pp. 173–186, ISBN: 978-3-642-36824-0, DOI: 10.1007/978-3-642-36824-0_17 (cit. on p. 75).

[Fornés 14] Alicia Fornés and Gemma Sánchez, « Analysis and Recognition of Music Scores », in: *Handbook of Document Image Processing and Recognition*, ed. by David Doermann and Karl Tombre, Springer London, 2014, pp. 749–774, ISBN: 978-0-85729-859-1, DOI: 10.1007/978-0-85729-859-1_24 (cit. on pp. 11, 20, 28–30, 32, 35).

[Fujinaga 04] Ichiro Fujinaga, « Staff Detection and Removal », in: *Visual Perception of Music Notation: On-Line and Off Line Recognition*, IGI Global, 2004, pp. 1–39, ISBN: 9781591402985, DOI: 10.4018/978-1-59140-298-5.ch001 (cit. on p. 30).

[Goodfellow 14] Ian Goodfellow et al., « Generative Adversarial Nets », in: *Advances in Neural Information Processing Systems 27*, ed. by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Curran Associates, Inc., 2014, pp. 2672–2680 (cit. on pp. 16, 45, 46, 50, 100, 102, 104, 110, 113, 120, 122).

[google 20] google, *TensorFlow*, Dec. 13, 2020, URL: `https://www.tensorflow.org/` (visited on 12/13/2020) (cit. on p. 69).

[Grosicki 08] Emmanuèle Grosicki, Matthieu Carre, Jean-Marie Brodin, and Edouard Geoffrois, « RIMES Evaluation Campaign for Handwritten Mail Processing », in: *ICFHR 2008 : 11th International Conference on Frontiers in Handwriting Recognition*, Montreal, Canada: Concordia University, Aug. 2008, pp. 1–6 (cit. on p. 80).

[Hajič 17] J. Hajič and P. Pecina, « The MUSCIMA++ Dataset for Handwritten Optical Music Recognition », in: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 01, Nov. 2017, pp. 39–46, DOI: `10.1109/ICDAR.2017.16` (cit. on pp. 14, 32, 37, 68, 89).

[Hajic 18] Jan Hajic, Matthias Dorfer, Gerhard Widmer, and Pavel Pecina, « Towards Full-Pipeline Handwritten OMR with Musical Symbol Detection by U-Nets », in: *ISMIR*, 2018 (cit. on pp. 32, 34).

[Huang 17] Jonathan Huang et al., « Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors », in: 2017, pp. 7310–7311 (cit. on pp. 59–61, 69, 74).

[Inoue 18] Naoto Inoue, Ryosuke Furuta, Toshihiko Yamasaki, and Kiyoharu Aizawa, « Cross-Domain Weakly-Supervised Object Detection Through Progressive Domain Adaptation », in: 2018, pp. 5001–5009 (cit. on p. 47).

[Jaderberg 15] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and koray kavukcuoglu, « Spatial Transformer Networks », in: *Advances in Neural Information Processing Systems 28*, ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Curran Associates, Inc., 2015, pp. 2017–2025 (cit. on pp. 13, 39, 40, 57).

[Kang 20] L. Kang, M. Rusiñol, A. Fornés, P. Riba, and M. Villegas, « Unsupervised Adaptation for Synthetic-to-Real Handwritten Word Recognition », in: *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Mar. 2020, pp. 3491–3500, DOI: 10.1109/WACV45572.2020.9093392 (cit. on p. 48).

[Karras 18] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen, *Progressive Growing of GANs for Improved Quality, Stability, and Variation*, Feb. 26, 2018, arXiv: 1710.10196 [cs, stat], URL: http://arxiv.org/abs/1710.10196 (visited on 04/07/2021) (cit. on p. 168).

[Kato 92] Hirokazu Kato and Seiji Inokuchi, « A Recognition System for Printed Piano Music Using Musical Knowledge and Constraints », in: *Structured Document Image Analysis*, ed. by Henry S. Baird, Horst Bunke, and Kazuhiko Yamamoto, Berlin, Heidelberg: Springer, 1992, pp. 435–455, ISBN: 978-3-642-77281-8, DOI: 10.1007/978-3-642-77281-8_20 (cit. on p. 30).

[Krizhevsky 17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, « ImageNet Classification with Deep Convolutional Neural Networks », in: *Communications of the ACM* 60.*6* (May 24, 2017), pp. 84–90, ISSN: 0001-0782, DOI: 10.1145/3065386 (cit. on p. 42).

[labri 18] labri, *DocCreator an Application to Generate Sythetic Document Images for Performance Evaluation and Retrainning*, July 12, 2018, URL: http://doc-creator.labri.fr/ (visited on 07/12/2018) (cit. on pp. 79, 82).

[LeCun] Yann LeCun, *MNIST Handwritten Digit Database, Yann LeCun, Corinna Cortes and Chris Burges*, URL: http://yann.lecun.com/exdb/mnist/ (visited on 02/13/2020) (cit. on p. 100).

[Lin 14] Tsung-Yi Lin et al., « Microsoft COCO: Common Objects in Context », in: *Computer Vision – ECCV 2014*, ed. by David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, Lecture Notes in Computer Science, Cham: Springer International Publishing, 2014, pp. 740–755, ISBN: 978-3-319-10602-1, DOI: 10.1007/978-3-319-10602-1_48 (cit. on p. 67).

[Lin 17] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár, « Focal Loss for Dense Object Detection », in: (Aug. 7, 2017), arXiv: 1708.02002 [cs] (cit. on p. 75).

[Liu 16] Wei Liu et al., « SSD: Single Shot MultiBox Detector », in: vol. 9905, Amsterdam, 2016, pp. 21–37, DOI: 10.1007/978-3-319-46448-0_2, arXiv: 1512.02325 (cit. on pp. 41, 42, 98).

[Long 15] Jonathan Long, Evan Shelhamer, and Trevor Darrell, « Fully Convolutional Networks for Semantic Segmentation », in: 2015, pp. 3431–3440 (cit. on p. 42).

[Marti 02] U.-V. Marti and H. Bunke, « The IAM-Database: An English Sentence Database for Offline Handwriting Recognition », in: *International Journal on Document Analysis and Recognition* 5.*1* (Nov. 1, 2002), pp. 39–46, ISSN: 1433-2833, DOI: 10.1007/s100320200071 (cit. on p. 80).

[Modayur 93] Bharath R. Modayur, Visvanathan Ramesh, Robert M. Haralick, and Linda G. Shapiro, « MUSER: A Prototype Musical Score Recognition System Using Mathematical Morphology », in: *Machine Vision and Applications* 6.*2* (Mar. 1, 1993), pp. 140–150, ISSN: 1432-1769, DOI: 10.1007/BF01211937 (cit. on p. 32).

[musitek 20] musitek, *MUSITEK - Music Scanning Software*, Nov. 2, 2020, URL: https://www.musitek.com/ (visited on 11/02/2020) (cit. on p. 35).

[myriad 20] myriad, *OMeR*, Nov. 2, 2020, URL: https://www.myriad-online.com/en/products/omer.htm (visited on 11/02/2020) (cit. on p. 35).

[neuratron 20] neuratron, *PhotoScore Music Scanning Software from Neuratron*, Nov. 2, 2020, URL: https://www.neuratron.com/photoscore.htm (visited on 11/02/2020) (cit. on p. 35).

[Pacha] Alexander Pacha, *Optical Music Recognition Datasets*, URL: https://apacha.github.io/OMR-Datasets/ (visited on 02/12/2020) (cit. on p. 95).

[Pacha 18a] Alexander Pacha and Jorge Calvo-Zaragoza, « Optical Music Recognition in Mensural Notation with Region-Based Convolutional Neural Networks », in: *ISMIR*, 2018, DOI: 10.13140/rg.2.2.28624.40965 (cit. on pp. 32, 34).

[Pacha 20] Alexander Pacha, *Apacha/MusicObjectDetector-TF: Music Object Detector with TensorFlow*, Dec. 13, 2020, URL: https://github.com/apacha/MusicObjectDetector-TF (visited on 12/13/2020) (cit. on p. 69).

[Pan 19] Zhaoqing Pan, Weijie Yu, Xiaokai Yi, Asifullah Khan, Feng Yuan, and Yuhui Zheng, « Recent Progress on Generative Adversarial Networks (GANs): A Survey », in: *IEEE Access* PP (Mar. 14, 2019), pp. 1–1, DOI: 10.1109/ACCESS.2019.2905015 (cit. on p. 45).

[petrucci 17] petrucci, *IMSLP/Petrucci Music Library: Free Public Domain Sheet Music*, July 19, 2017, URL: http://imslp.org/ (visited on 07/19/2017) (cit. on p. 148).

[Prearu 70] D. S. Prearu, « Computer Pattern Recognition of Standard Engraved Music Notation », in: *Ph.D thesis, Massachusetts Institute of Technology* (1970) (cit. on p. 31).

[Pruslin 66] D. Pruslin, « Automatic Recognition of Sheet Music », Massachusetts Institute of Technology, 1966 (cit. on pp. 11, 20, 28, 30).

[Pugin 06] Laurent Pugin, « Optical Music Recognitoin of Early Typographic Prints Using Hidden Markov Models. », in: *ISMIR*, 2006, pp. 53–56 (cit. on pp. 31, 74).

[Radford 16] Alec Radford, Luke Metz, and Soumith Chintala, « Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks », in: (Jan. 7, 2016), arXiv: 1511.06434 [cs] (cit. on p. 102).

[Rebelo 09] A. Rebelo, G. Capela, and Jaime S. Cardoso, « Optical Recognition of Music Symbols », in: *International Journal on Document Analysis and Recognition (IJDAR)* 13.*1* (Nov. 17, 2009), pp. 19–

31, ISSN: 1433-2833, 1433-2825, DOI: 10.1007/s10032-009-0100-1 (cit. on pp. 32, 37, 96).

[Rebelo 12] 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 », in: *International Journal of Multimedia Information Retrieval* 1.*3* (Mar. 2, 2012), pp. 173–190, ISSN: 2192-6611, 2192-662X, DOI: 10.1007/s13735-012-0004-6 (cit. on pp. 20, 28, 29, 32).

[Redmon 16] Joseph Redmon and Ali Farhadi, « YOLO9000: Better, Faster, Stronger », in: (Dec. 25, 2016), arXiv: 1612.08242 [cs] (cit. on p. 70).

[Remez 18] Tal Remez, Jonathan Huang, and Matthew Brown, « Learning to Segment via Cut-and-Paste », in: 2018, pp. 37–52, arXiv: 1803.06414 (cit. on p. 49).

[Ren 15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, « Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks », in: *Advances in Neural Information Processing Systems 28*, ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Curran Associates, Inc., 2015, pp. 91–99 (cit. on p. 41).

[Reyna] Rosendo Reyna, *Music Engraving - History and Process of Engraving Music*, URL: https://musicprintinghistory.org/about-music-engraving/ (visited on 11/11/2020) (cit. on p. 27).

[Rıos-Vila 20] Antonio Rıos-Vila, Jorge Calvo-Zaragoza, and Jose M Inesta, « Exploring the Two-Dimensional Nature of Music Notation for Score Recognition with End-to-End Approaches », in: *2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2020, p. 6, DOI: 10.1109/ICFHR2020.2020.00044 (cit. on p. 34).

[Roach 88] J. W. Roach and J. E. Tatem, « Using Domain Knowledge in Low-Level Visual Processing to Interpret Handwritten Music: An Experiment », in: *Pattern Recognition* 21.*1* (Jan. 1, 1988), pp. 33–

44, ISSN: 0031-3203, DOI: 10.1016/0031-3203(88)90069-6 (cit. on p. 31).

[Ronneberger 15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, « U-Net: Convolutional Networks for Biomedical Image Segmentation », in: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, Lecture Notes in Computer Science, Springer, Cham, Oct. 5, 2015, pp. 234–241, ISBN: 978-3-319-24574-4, DOI: 10.1007/978-3-319-24574-4_28 (cit. on pp. 16, 43, 102).

[Sánchez 16] J. A. Sánchez, V. Romero, A. H. Toselli, and E. Vidal, « ICFHR 2016 Competition on Handwritten Text Recognition on the READ Dataset », in: *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, Oct. 2016, pp. 630–635, DOI: 10.1109/ICFHR.2016.0120 (cit. on p. 80).

[ScanScore-Team] ScanScore-Team, *Sheet Music Scanner | SCANSCORE Sheet Music Scanning Software*, URL: https://scan-score.com/en/ (visited on 11/02/2020) (cit. on p. 35).

[Schweer 18] Werner Schweer, *MuseScore Is an Open Source and Free Music Notation Software. For Support, Contribution, Bug Reports, Visit MuseScore.Org. Fork and Make Pull Requests!*, MuseScore, July 6, 2018 (cit. on p. 82).

[Simonyan 15] Karen Simonyan and Andrew Zisserman, « Very Deep Convolutional Networks for Large-Scale Image Recognition », in: (Apr. 10, 2015), arXiv: 1409.1556 [cs] (cit. on pp. 42, 102).

[SMuFL 13] SMuFL, *Standard Music Font Layout*, 2013, URL: https://www.smufl.org/ (visited on 07/06/2018) (cit. on pp. 81, 89).

[Szegedy 15] C. Szegedy et al., « Going Deeper with Convolutions », in: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 1–9, DOI: 10.1109/CVPR.2015.7298594 (cit. on p. 42).

[Tuggener 18a] L. Tuggener, I. Elezi, J. Schmidhuber, M. Pelillo, and T. Stadelmann, « DeepScores-A Dataset for Segmentation, Detection and Classification of Tiny Objects », in: *2018 24th International Con-*

ference on Pattern Recognition (ICPR), Aug. 2018, pp. 3704–3709, DOI: 10.1109/ICPR.2018.8545307 (cit. on p. 37).

[Tuggener 18b] Lukas Tuggener, Ismail Elezi, Jürgen Schmidhuber, and Thilo Stadelmann, « Deep Watershed Detector for Music Object Recognition », in: Society for Music Information Retrieval, 2018, DOI: 10.21256/zhaw-3760 (cit. on p. 32).

[vdWel 17] Eelco van der Wel and Karen Ullrich, « Optical Music Recognition with Convolutional Sequence-to-Sequence Models », in: Suzhou, China, July 16, 2017, arXiv: 1707.04877 (cit. on pp. 30, 34).

[visiv 20] visiv, *Scan with Visiv SharpEye Music Scanning*, Nov. 2, 2020, URL: http://www.visiv.co.uk/ (visited on 11/02/2020) (cit. on p. 35).

[wikipedia 20] wikipedia, *List of Musical Symbols*, in: *Wikipedia*, Nov. 16, 2020 (cit. on p. 25).

[Yang 17] Jianwei Yang, Anitha Kannan, Dhruv Batra, and Devi Parikh, « LR-GAN: Layered Recursive Generative Adversarial Networks for Image Generation », in: (Mar. 5, 2017), arXiv: 1703.01560 [cs] (cit. on p. 49).

[Zhu 17] J. Zhu, T. Park, P. Isola, and A. A. Efros, « Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks », in: *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 2242–2251, DOI: 10.1109/ICCV.2017.244, arXiv: 1703.10593 (cit. on p. 47).

Appendix A

Experiments

A.1 Isolating-GAN Robustness Evaluation

We now measure the impact of some of the most important hyper-parameters of our method. First, we measure the impact on the amount of images in the real dataset that does not contain any symbol we want to detect. Then, we also show the impact of the isolated symbols sizes generated using the minimum and maximum size bounds hyper-parameters.

A.1.1 Impact of Symbol Frequency in Real Dataset

The stability of our GAN model depends not only on the architecture but also on the data used for training the model. In this section, we study the impact of the amount of rejection images in the real dataset used as input to the generator on the stability of the GAN model.

Objectives What we call rejection images are empty images produced by our first preprocessing step of simplifying the music notation of real music scores explained in section 4.2.1 by identifying RoIs that has a high probability of containing the type of symbols we want to detect. This step will obviously always have false-positive examples with images that do not contain any symbols to detect. Since there are no way to know in advance of the ratio between the amount of rejection images and images that do contains symbols to detect, we investigate different ratios where we artificially modified the amount images that do contain symbols to detect and rejection images. This experiment is based on the experimental settings of the Isolating-GAN-ReconsLr+Neg experiment in section 5.2.1.3 and only modify the composition of the real dataset used as input to the generator.

Datasets We artificially modify the ratio between images that contains a symbol to detect and rejection images. We explore five different ratio values of 1%, 5%, 10%, 50% and 90% of images with symbols to detect while keeping in mind that after manually annotating a lot of music score pages, we believe that around 10% of images of a real dataset contains symbols to be detected. Images with accidental symbols that we do not want to detect were also removed, since we noticed that the GAN model can sometimes confuse symbols with different accidental classes as the same class. This simplification is done in order to isolate the sole effect of rejection images on the training of the GAN model.

Evaluation In order to evaluate our method with different rejection ratios, we use our early stopping mechanism as presented in section 5.2.2.1 where we use both a small validation dataset of 20 examples per symbol classes with bootstrapping and the fully annotated training dataset for evaluation. We report results using a mAP metric with IoU > 0.75 computed on the fully annotated training dataset but at epoch that maximized the results on the small validation dataset. The training is repeated using the same rejection ratio 10 times with a different random seed and present the median, first and third quartile mAP results. We also show the mAP computed on the full training dataset of the model that had the best results on the small validation dataset.

Isolating-GAN-RejectRatio Results Figure A.1 shows the results of using different rejection ratio where we observe different behavior for the three different accidental classes. The model produces the most stable results for the sharp and natural classes when at least 5% of all images contains symbols to be detected. For the Flat class, the model needs at least 25% of all images to contain symbols for the training to be stable.

However, this instability can be overcome using our early-stopping mechanism with our small validation set by selecting the best performing model out of a pool of 10 trained model. Using this selection mechanism, we can maintain a good detection performance of $\sim$91% of mAP for the Flat class for ratios of 5% and 10%.

In our experimental settings and with the dataset used here, we show that the detection performance degrades significantly when only 1% of the images contain symbols to detect. We believe this is due to the discrepancy in its training objective where the synthetic isolated dataset has systematically an isolated symbol inside the blank image while the real images seen by the generator has very few images with symbols to isolate. This experiment shows that with our experimental settings and our specific set of datasets, our method is able to cope with this discrepancy when at least 5% of images contains symbols to detect which should be enough considering that we estimate that around 10% of images of a real dataset will contain symbols to detect. For this experiment, we repeated each training 10 times, on 3 different accidental classes and explored 5 different ratio values and resulted in a total of 150 trainings to do.

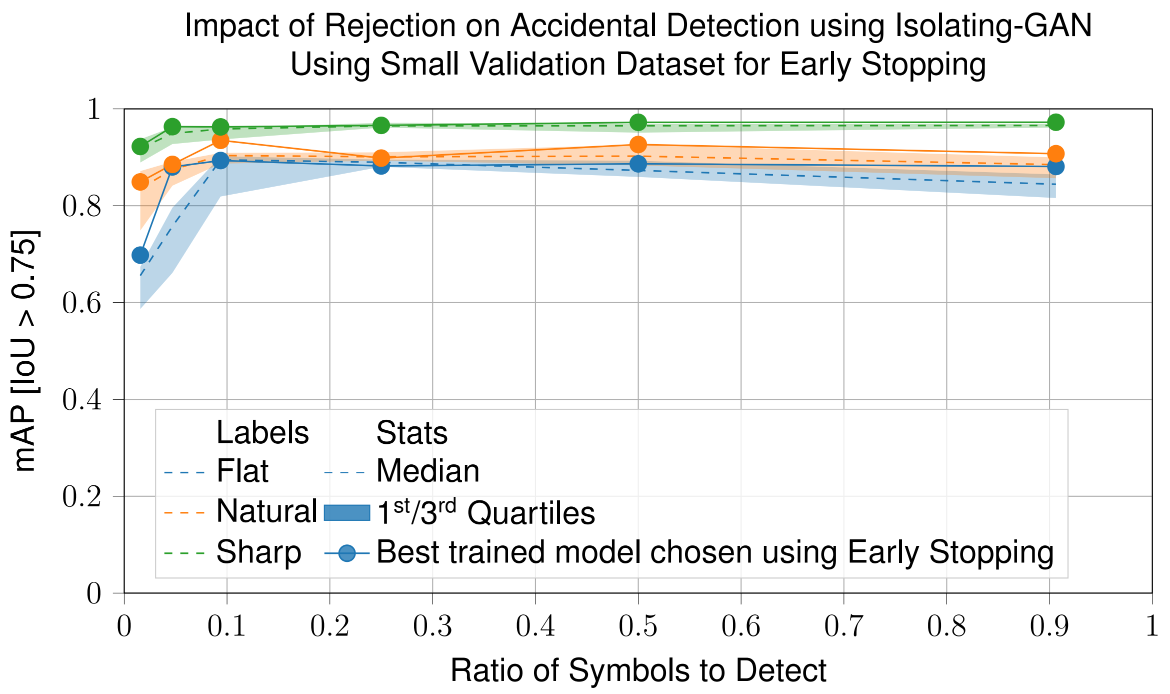


Figure A.1 – Study of the impact of rejection on the detection of accidental symbols using Isolating-GAN. We show detection results using 1%, 5%, 10%, 25%, 50% and 90% of symbol to detect out of the total amount of real data, see section A.1.1. Each experiments are repeated 10 times and we report the median, first and third quartiles of the real mAP at the best epoch found using our early stopping mechanism. We also show the mAP of the model chosen by our early stopping mechanism which gave the best results on the small validation dataset.

A.1.2 Sensibility to Generation Sizes of Isolated Symbols

Objectives One of the standing stone of our method is the use of isolated symbols to drive both the image translation done by the GAN model and symbol detection done by the SSD detector. However, isolated symbols can not be used as provided since they have to be similar to the symbols we want to detect in real historical music scores.

We also use these isolated symbols in conjunction with simple data augmentation techniques which are commonly used in deep learning experiments to introduce more variation to the data and normally improve the performance of the trained model. One basic modification to be made is the size that a symbol will have since isolated symbols and real symbols won't probably be of the same resolution. Since no ground truth exists for the real music symbols we want to detect, we can only guess the ranges of sizes for each class of symbols. Fortunately, music symbols follows strict typesetting rules, and we can often guess the possible range of sizes that a symbol class will have. On the other hand, a lot of variation in sizes can be present by using different music typefaces which will have different sizes and width/height ratio for the same symbol class. During the various experiments made while investigating the use of a GAN model for symbol detection, we noticed that the GAN model can be very sensitive to the variation in sizes that isolated symbols can take. Therefore, we propose to evaluate in this section the robustness of our Isolating-GAN in regard to isolated symbol sizes.

As usual, we base our experiment on the Isolating-GAN-ReconsLr+Neg shown in section 5.2.1.3 and reuse the same model and datasets except for one modification for the isolated symbol dataset.

Datasets We use the same dataset setup as our baseline experiment Isolating-GAN-ReconsLr+Neg except for one modification to the minimum and maximum possible width and height of isolated symbols. We show in table A.1 the detailed minimum and maximum width/heights values per class for the baseline and this new experiment. For each class, we reduce the minimum width/height and augment the maximum width/height that accidental symbols can take.

Table A.1 – Minimum and maximum width/height settings of isolated symbols for the Isolating-GAN-ReconsLr+Neg and Isolating-GAN-Sizes experiments. The values are in pixels and symbols are pasted in canvas of 128×128 pixels.

Experiment	Flat				Natural				Sharp			
	height		width		height		width		height		width	
	min	max	min	max	min	max	min	max	min	max	min	max
Isolating-GAN-ReconsLr+Neg (low)	79	112	25	37	98	128	16	36	72	112	27	45
Isolating-GAN-Sizes (high)	64	128	16	56	64	128	8	48	64	128	16	64

Evaluation As usual, we evaluate our method with our early stopping mechanism presented in section 5.2.2.1 by computing the mAP metric with IoU > 0.75 computed on the fully annotated training dataset at the epoch that maximized the results on the small validation dataset. We repeat the training 10 times with different random seeds and present box plots of the 10 runs.

Isolating-GAN-Sizes Experimental Results We show the results of this experiment in fig. A.2 where we can see that using a high variation in symbols sizes results in slightly worse results. For example, we can see that for the Flat class, the variation in the first and third quartile is reduced by half by using a lower variation of symbol sizes. For the Natural and Sharp class, the stability of the results are significantly better by using a lower variation of symbol sizes.

On the other hands, the maximum mAP obtain by one of the ten trained models is similar in either high or low variation of symbol sizes experimental settings. This shows that our Isolating-GAN method can still achieve find the best available detection performance even with badly adjusted size parameters.

For this experiment, we repeated each training 10 times, on 3 accidental classes and two set of size parameters for isolated symbols, totaling to 60 trainings.

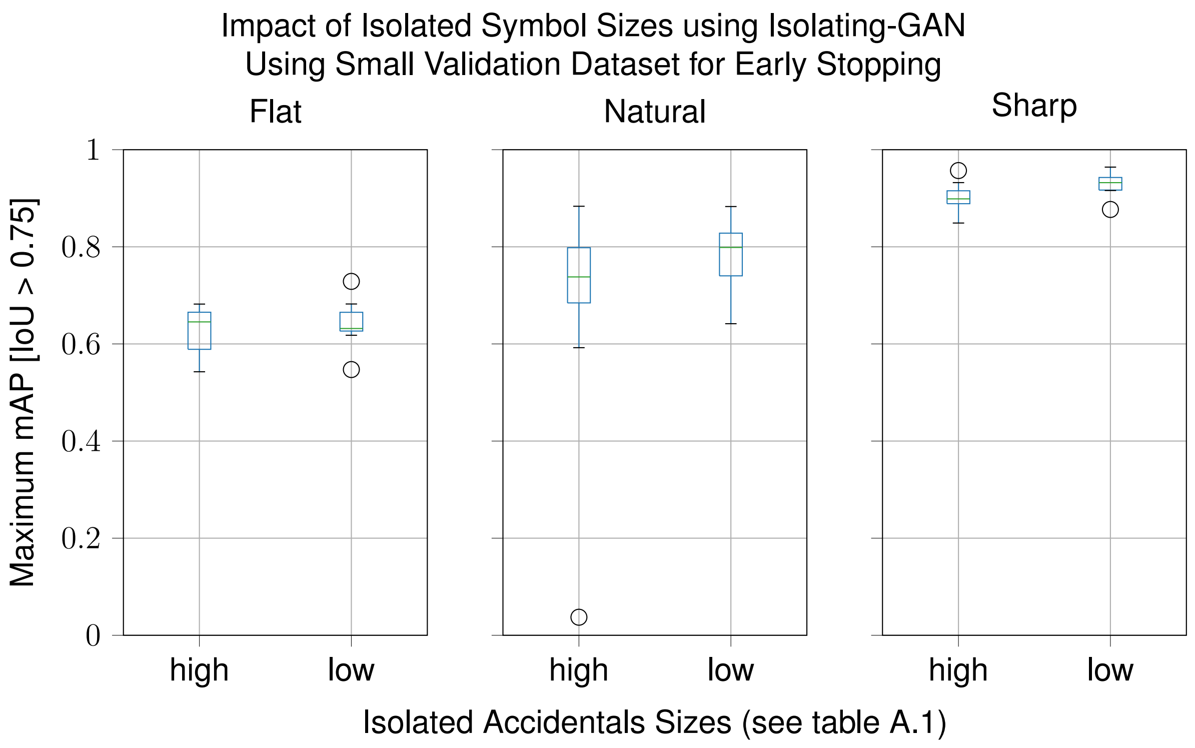


Figure A.2 – Study of the impact of isolated symbols sizes. We show detection results using a larger range of possible sizes for isolated symbols. Each experiments are repeated 10 times and we report the real mAP at the best epoch found using our early stopping mechanism.

INSA de RENNES
Service des Formations

AVIS DU JURY SUR LA REPRODUCTION DE LA THESE SOUTENUE

Titre de la thèse:
Combinaison de modèles génératifs non supervisés et d'une méthode syntaxique pour la détection de symboles musicaux avec peu de données annotées

Nom Prénom de l'auteur : CHOI KWON-YOUNG

Membres du jury :
- Monsieur PAQUET Thierry
- Madame EGLIN Véronique
- Monsieur COUASNON Bertrand
- Monsieur MOUCHERE Harold
- Madame FORNES Alicia
- Monsieur FUJINAGA Ichiro
- Monsieur RICQUEBOURG Yann
- Monsieur ZANIBBI Richard

Président du jury : Harold MOUCHERE

Date de la soutenance : 30 Juin 2021

Reproduction de la these soutenue

[✓] Thèse pouvant être reproduite en l'état

☐ Thèse pouvant être reproduite après corrections suggérées

Fait à Rennes, le 30 Juin 2021

Signature du président de jury

[illegible]

Le Directeur,

[illegible]

Abdellatif MIRAOUI

INSTITUT NATIONAL des SCIENCES APPLIQUEES - RENNES

Titre : Combinaison de modèles génératifs non supervisés et d'une méthode syntaxique pour la détection de symboles musicaux avec peu de données annotées

Mot clés : reconnaissance optique de partitions musicales, Deep Learning, détection de symboles, réseau antagoniste génératif, apprentissage non supervisé, reconnaissance de documents historiques

Résumé : Dans ces travaux, nous étudions la détection de symboles musicaux dans des partitions historiques imprimées, complexes, denses et bruitées en utilisant des modèles de détection de Deep Learning. Nous proposons une étude comparative de plusieurs modèles de détection de l'état de l'art appliqués aux symboles musicaux ainsi qu'une nouvelle architecture basée sur le Spatial Transformer pour une tâche de détection spécifique et contrainte. Bien que cette nouvelle architecture nous a permis d'explorer une approche originale à la détection, nous obtenons 94,81 % de mAP alors que la meilleure méthode de l'état de l'art obtient un mAP de 98,73 %. L'utilisation de modèles de Deep Learning nécessitant une grande quantité de données annotées, nous proposons l'Isolating-GAN, une nouvelle méthode de détection de symboles musicaux non supervisée et basée sur un réseau antagoniste génératif (GAN). En utilisant uniquement des exemples de symboles isolés, nous construisons un modèle génératif de type encodeur-décodeur capable de filtrer et d'isoler des symboles musicaux de leur contexte bruité et nous entraînons ce modèle en utilisant une fonction d'apprentissage hybride. Les symboles précédemment isolés sont par la suite détectés en utilisant un petit détecteur préentrainé avec des symboles isolés sur fond blanc. Avec cette approche, nous obtenons un mAP de 82,5 % dans le cadre d'une tâche de détection de trois types d'altérations. Nous démontrons que l'apport de l'Isolating-GAN pour filtrer et isoler les symboles avant détection permet de réduire le nombre de faux positifs de 2696 à 57. L'ensemble a été appliqué sur 1774 pages de partitions anciennes et a permis de détecter 38908 altérations.

Title: Combination of unsupervised generative models and a syntactical method for music symbol detection with few annotated data

Keywords: optical music recognition, Deep Learning, symbol detection, generative adversarial network, unsupervised learning, historical document recognition

Abstract: In this work, we study the detection of music symbols in images of complex, historical, dense, noisy and damaged printed music scores through the use of Deep Learning detection models. We propose a comparative study of multiple state-of-the-art detection models applied to music symbols as well as a new architecture based on the Spatial Transformer for a very focused music symbol detection task. Although we explored an original detection approach with this new architecture, we obtain 94.81% of mAP while the best state-of-the-art method obtains a mAP of 98.73%. Since the use of Deep Learning methods requires a huge amount of annotated data, we present the Isolating-GAN, a novel unsupervised music symbol detection method based on Generative Adversarial Network (GAN). Using only isolated music symbols and no symbol level detection annotations, we build an encoder-decoder generative model able to filter and isolate music symbols from background noise and shapes and train the model using a hybrid adversarial and image reconstruction loss. Symbols isolated by the generative model are then detected using a small detector pre-trained using isolated symbols in empty white images. With this approach, we obtain a mAP of 82.5% for a detection task with three accidental classes. We also demonstrate that using the Isolating-GAN to filter and isolate symbols before the detection operation reduces the number of false positives from 2,696 to 57. The method was applied on 1,774 pages of historical music scores and was able to detect 38,908 new accidentals.