Détection non-supervisée de motifs dans les partitions musicales manuscrites

Riyadh Benammar

N° d'ordre NNT : 2019LYSEI112

THÈSE de DOCTORAT DE L'UNIVERSITÉ DE LYON

opérée au sein de

INSTITUT NATIONAL DES SCIENCES APPLIQUÉES DE LYON

Ecole Doctorale N° 512

INFORMATIQUE ET MATHÉMATIQUES

Spécialité/ discipline de doctorat :

INFORMATIQUE

Soutenue publiquement le 28 Novembre 2019 par :

Riyadh BENAMMAR

Détection non-supervisée de motifs dans des partitions musicales manuscrites

Devant le jury composé de :

Rigaux,	Philippe	Professeur des Universités	CNAM- Paris	Rapporteur
Coüasnon,	Bertrand	Maître de conférences HDR	INSA-Rennes	Rapporteur
Plantevit,	Marc	Maître de conférences HDR	UCB Lyon 1	Examinateur
Vincent,	Nicole	Professeur des Universités	Université Paris Descartes	Examinatrice
Venturini,	Gilles	Professeur des Universités	Université de Tours	Examinateur
Eglin,	Véronique	Professeur	INSA-Lyon	Directrice de thèse
Largeron,	Christine	Professeur des Universités	UJM- Saint-Etienne	Co-directrice de thèse
Pardoen,	Mylène	Ingénieur de recherche	MSH Lyon-Saint-Etienne	Co-directrice de thèse

Remerciements

Il me sera très difficile de remercier tout le monde car c'est grâce à l'aide de nombreuses personnes que j'ai pu mener cette thèse à son terme. Je voudrais tout d'abord remercier grandement mes directrices de thèse, Madame Véronique EGLIN, Madame Christine LARGERON et Madame Mylène PARDOEN, pour toute leur aide.

Je suis ravi d'avoir travaillé en leur compagnie car outre leur appui scientifique, elles ont toujours été là pour me soutenir et me conseiller au cours de l'élaboration de cette thèse.

J'adresse tous mes remerciements à Monsieur Philippe RIGAUX, Professeur au Conservatoire National des Arts et Métiers, ainsi qu'à Monsieur Bertrand COÜASNON, Professeur à l'Institut National des Sciences Appliquées de Rennes, de l'honneur qu'ils m'ont fait en acceptant d'être rapporteurs de cette thèse.

Je tiens à remercier Madame Nicole VINCENT pour avoir accepté de participer à mon jury de thèse et pour sa participation scientifique ainsi que le temps qu'elle a consacré à ma recherche.

Je remercie également Monsieur Marc PLANTEVIT et Monsieur Gilles VENTURINI pour l'honneur qu'ils me font d'être dans mon jury de thèse.

Il m'est impossible d'oublier les membres de l'équipe IMAGINE du LIRIS - Lyon et l'équipe Data Intelligence du LHC - Saint-Etienne pour leur aide précieuse pour ma recherche bibliographique.

Je remercie toutes les personnes avec qui j'ai partagé mes études et notamment ces années de thèse.

Résumé

Cette thèse s'inscrit dans le contexte de la fouille de données appliquées aux partitions musicales manuscrites anciennes et vise une recherche de motifs mélodiques ou rythmiques fréquents définis comme des séquences de notes répétitives aux propriétés caractéristiques. On rencontre un grand nombre de déclinaisons possibles de motifs : les transpositions, les inversions et les motifs dits « miroirs ». Ces motifs permettent aux musicologues d'avoir un niveau d'analyse approfondi sur les œuvres d'un compositeur ou d'un style musical. Dans un contexte d'exploration de corpus de grande taille où les partitions sont juste numérisées et non transcrites, une recherche automatisée de motifs vérifiant des contraintes ciblées devient un outil indispensable à leur étude. Pour la réalisation de l'objectif de détection de motifs fréquents sans connaissance a priori, nous sommes partis d'images de partitions numérisées. Après des étapes de prétraitements sur l'image, nous avons exploité et adapté un modèle de détection et de reconnaissance de primitives musicales (tête de notes, hampes...) de la famille de réseaux de neurones à convolutions de type Region-Proposal CNN (RPN). Nous avons ensuite développé une méthode d'encodage de primitives pour générer une séquence de notes en évitant la tâche complexe de transcription complète de l'œuvre manuscrite. Cette séquence a ensuite été analysée à travers l'approche CSMA (Contraint String Mining Algorithm) que nous avons conçue pour détecter les motifs fréquents présents dans une ou plusieurs séquences avec une prise en compte de contraintes sur leur fréquence et leur taille, ainsi que la taille et le nombre de sauts autorisés (gaps) à l'intérieur des motifs. La prise en compte du gap a ensuite été étudiée pour contourner les erreurs de reconnaissance produites par le réseau RPN évitant ainsi la mise en place d'un système de post-correction des erreurs de transcription des partitions. Le travail a été finalement validé par l'étude des motifs musicaux pour des applications d'identification et de classification de compositeurs.

Mots clés

Détection et Reconnaissance de primitives musicales, extraction des motifs musicaux, Identification et classification de compositeurs.

Abstract

This thesis is part of the data mining applied to ancient handwritten music scores and aims at a search for frequent melodic or rhythmic motifs defined as repetitive note sequences with characteristic properties. There are a large number of possible variations of motifs : transpositions, inversions and so-called "mirror" motifs. These motifs allow musicologists to have a level of in-depth analysis on the works of a composer or a musical style. In a context of exploring large corpora where scores are just digitized and not transcribed, an automated search for motifs that verify targeted constraints becomes an essential tool for their study. To achieve the objective of detecting frequent motifs without prior knowledge, we started from images of digitized scores. After pre-processing steps on the image, we exploited and adapted a model for detecting and recognizing musical primitives (note-heads, stems...) from the family of Region-Proposal CNN (RPN) convolution neural networks. We then developed a primitive encoding method to generate a sequence of notes without the complex task of transcribing the entire manuscript work. This sequence was then analyzed using the CSMA (Constraint String Mining Algorithm) approach designed to detect the frequent motifs present in one or more sequences, taking into account constraints on their frequency and length, as well as the size and number of gaps allowed within the motifs. The gap was then studied to avoid recognition errors produced by the RPN network, thus avoiding the implementation of a post-correction system for transcription errors. The work was finally validated by the study of musical motifs for composers identification and classification.

Keywords

Musical primitives detection and recognition, musical motifs mining, Composers identification and classification.

Avant-propos

La thèse s'inscrit dans le cadre du projet Magazin de Musique BIS commencé en 2013 et présenté par le département de musicologie de l'université Lumière-Lyon 2 (IHRIM , UMR 5317), impliqué dans la préservation et la transmission des connaissances relatives de partitions du 18ème siècle (lecture critique de partitions, études expertes en musicologie). Le financement de ce projet a été assuré par une subvention de la région Auvergne Rhône-Alpes s'inscrivant dans le cadre de ses appels à financement de thèse d'ouverture pluridisciplinaire STIC-SHS (2015-2019).

Au cours de cette thèse, trois publications ont été acceptées dont deux dans des conférences internationales. Un article a également été soumis à Pattern Recognition Letters Journal et est en attente de notification. L'ensemble des publications est résumé dans les références suivantes :

— Riyadh Benammar, Christine Largeron, Véronique Eglin, Mylène Pardoen : "Discovering motifs with variants in music databases." in International Symposium on Intelligent Data Analysis. Springer, 2017, pp. 14–26.

— Riyadh Benammar, Christine Largeron, Véronique Eglin, Mylène Pardoen : Recherche de motifs pour l'étude critique de partitions musicales. EGC 2019 : 389-394.

— Riyadh Benammar, Véronique Eglin, et Christine Largeron. Extraction of Musical Motifs from Handwritten Music Score Images. In : VISAPP-VISIGRAPP, 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications. 2019.

— Riyadh Benammar, Christine Largeron, Véronique Eglin, Mylène Pardoen : Efficient sequential gap-based motif mining on music score images for composer identification.

In : Pattern Recognition Letters. (Soumis)

Table des matières

Table des figures

Liste des tableaux

Introduction

Dans le contexte de l'instrumentation de corpus inédits (constitués de manuscrits autographes inédits, dont l'écriture originale et la présentation ne permettent pas en l'état une exécution aisée des musiciens ni une exploitation optimale pour les musicologues), nous pouvons citer des initiatives importantes dans le domaine musicologique (pour l'édition de partitions musicales en ligne) telles que le projet Neuma (Netword-enabled & user friendly musical analysis tools - projet ANR Contint 2008). Cette initiative a visé la diffusion de corpus de partitions musicales souvent anciennes (et majoritairement inédites) par l'élaboration de cahier des charges précis sur les dispositifs d'analyse musicale automatique. Nous l'avons d'ailleurs exploité dans cette thèse à des fins d'identifications de compositeurs. Les outils actuellement disponibles (recherche mélodique, exacte ou partielle, calcul automatique des ambitus) n'existent que très partiellement même pour des sources dont on dispose d'une information très complète généralement sonore. Ainsi les outils développés dans le cadre du projet NEUMA permettent déjà les recherches ponctuelles de mélodies ou de rythmes. Un certain nombre d'autres outils ont aussi été proposés dans le projet ANR 'Musicologie des techniques de composition contemporaine' porté par l'IRCAM, notamment pour traiter des corpus de musiques polyphoniques [Donin et al., 2010]. Dans les collections de partitions numérisées, où aucune information ni sonore ni textuelle n'existe, il est nécessaire de traiter les données de bout en bout : de la partition numérisée à son interprétation par des outils dédiés.

L'étude de partitions musicales à l'aide d'outils d'analyse d'images et de fouille de données automatisés offre de nouvelles perspectives pour l'analyse musicale et l'édition critique de partitions. Dans ce contexte, la thèse est consacrée à l'étude des partitions, avec pour

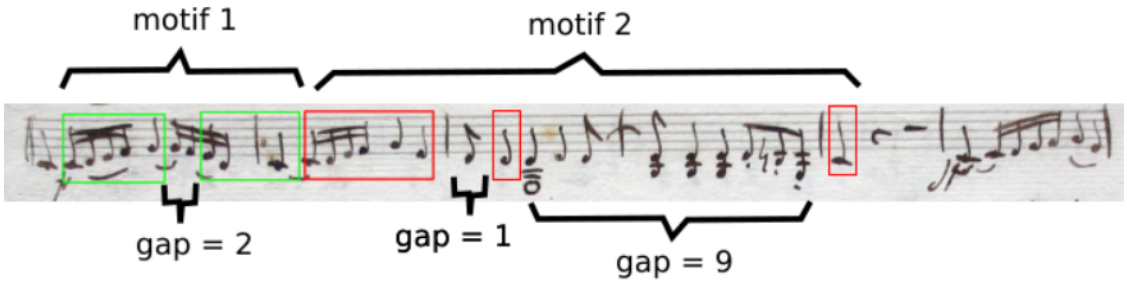


FIGURE 1 – Un motif à deux occurrences avec des sauts autorisés

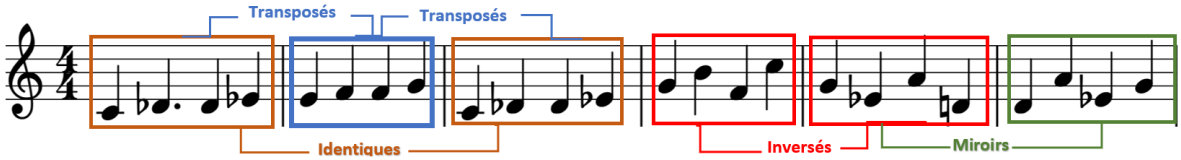


FIGURE 2 – Variantes des motifs

objectifs principaux d'aider à comprendre les origines musicales et d'élaborer des grilles de lecture, d'interpréter et de faciliter la recherche d'information musicale. Nous nous intéressons dans ce travail à l'analyse des variations et l'utilisation de motifs musicaux qui consiste en l'identification des arrangements contigus ou non contigus de notes. Dans la figure 1, on montre un exemple d'un motif non contigus (contient des *sauts* ou *gaps*) à deux occurrences tels que la première occurrence contient un *gap* de taille de 2 (notes) et la deuxième occurrence contient deux *gaps* de taille respective de 1 et 9.

La thèse par sa nature pluridisciplinaire repose sur des champs disciplinaires différents permettant de couvrir des domaines de la musicologie, de l'analyse d'images et de l'analyse des données comme illustrés à la figure 3. Le premier axe, lié au traitement des images assure le prétraitement des données en permettant de localiser et de reconnaître les symboles musicaux, de les coder et de les indexer. Le second axe est relatif à la fouille de données permettant d'aboutir à la détection de motifs et de motifs dérivés dans une partition. Dans le domaine du traitement d'images, notre objectif est la réalisation d'un système de reconnaissance de symboles musicaux. Cette étape permet de reconnaître les symboles afin de les encoder pour obtenir une transcription partielle basée sur la reconnaissance des indicateurs (formes primitives et symboles) musicaux dans le but de produire une séquence temporelle de primitives encodées. Naturellement, cette tâche est rendue plus complexe par la présence de

variations internes de l'écriture d'un symbole ou d'une primitive musicale au sein d'une même partition et des variations entre écritures de différents scripteurs. Nous avons prévu d'utiliser de manière efficace les techniques de détection et classification les plus récentes basées sur les réseaux de neurones profonds. Pour réaliser cette tâche, les RPN, pour Region Proposal Convolutional Neural Net, (dont nous détaillerons le principe dans la Section 1.3) ont prouvé leur efficacité dans la résolution de problèmes de détection et classification d'objets dans les scènes naturelles. Nous montrons leur performance dans le cadre de détection et classification des primitives de musique. Cette étape, suivie d'un processus d'encodage d'entiers représentatifs des notes, de leur position, et de certaines de leurs propriétés. On distingue trois niveaux d'information : mélodique, rythmique et harmonique. La séquence est conçue comme une succession temporelle de valeurs, relative à ces trois niveaux d'informations, mais pouvant également ne dépendre que d'un seul niveau d'encodage exclusivement mélodique ou rythmique par exemple.

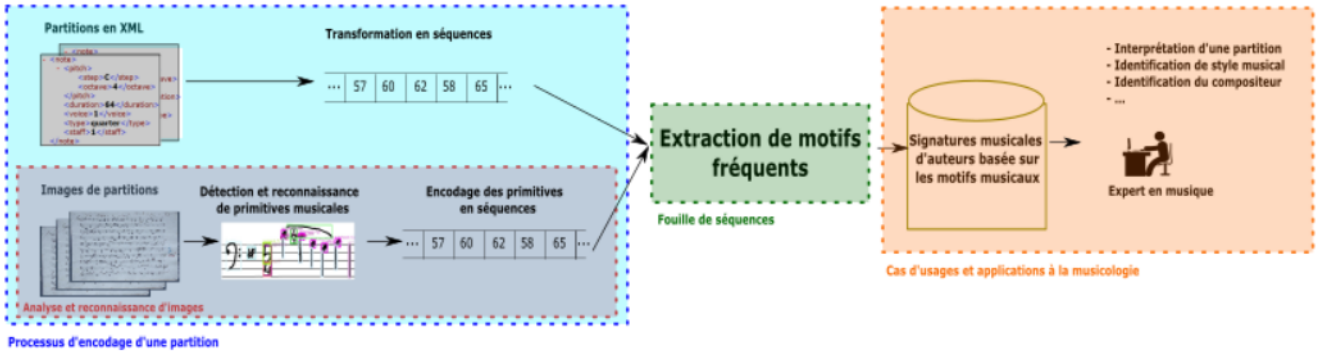


FIGURE 3 – Chaîne de traitements : de la reconnaissance d'images de partitions aux cas d'usages en musicologie

Chaque séquence de primitives transcrites est ensuite utilisée pour extraire des motifs musicaux ayant des formes particulières (identique, miroir, transposée, etc.), voir figure 2.

La partie extraction de connaissances à partir de données consiste à proposer une approche permettant d'identifier des motifs mélodiques et / ou rythmiques dans une partition transcrite afin d'en obtenir une représentation utilisable dans d'autres tâches comme par exemple le classement, supervisé ou non, de partitions ou encore la reconnaissance du compositeur. De même que les mots apparaissant dans un document textuel peuvent servir à caractériser le document et même parfois son auteur, dans cette thèse nous considérons que les motifs utilisés

par un compositeur fournissent une représentation utilisable comme une signature de l'œuvre voire du compositeur. Du fait de cette hypothèse, nous nous intéressons aux motifs mais aussi à des variantes de ces motifs introduites dans la partition par le compositeur et caractérisant son style. Dans cette étude nous considérons trois variantes possibles d'un motif : miroir, transposée et inversée qui sont illustrées sur la figure 2. Une partition étant assimilée à une séquence, c'est en faisant appel à des techniques de fouille de données (pattern mining) que nous avons naturellement cherché à identifier les motifs et pour ce faire nous avons conçu un algorithme nommé CSMA (Constrained String Mining Algorithm) permettant de traiter une ou plusieurs séquences selon que la partition comporte elle-même un ou plusieurs instruments (ou voix).

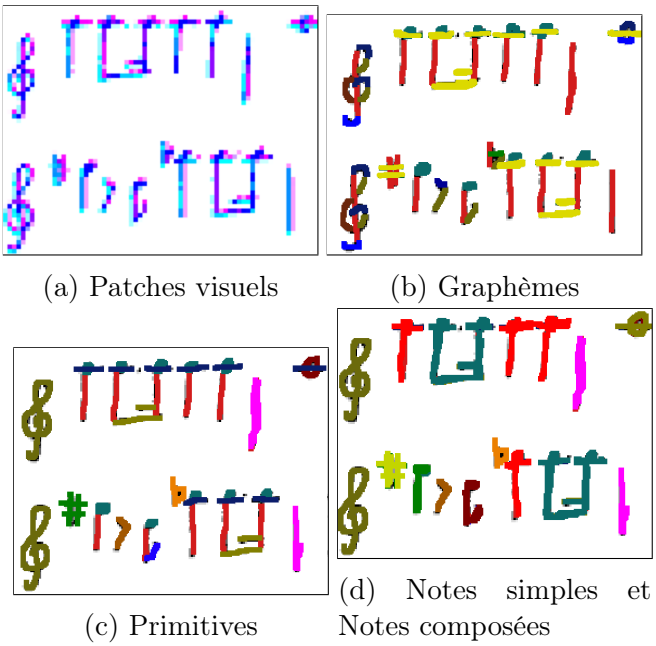
Le manuscrit est ensuite organisé comme suit : dans la section 1, on présente les approches existantes permettant le traitement automatique des images de partitions musicales. Cette partie couvre les étapes de pré-traitement d'images, détection et suppression des lignes de portée, la détection et la classification des primitives musicales, les primitives étant des fragments graphiques atomiques formant une note musicale (e.g. une noire est formé d'une tête de note pleine et une hampe). Ces primitives sont encodées en fonction de leurs positions sur la portée afin de représenter l'information musicale sous forme de séquences. Cette partie est détaillée dans la section 1.5. Ensuite, afin d'extraire les motifs fréquents de ces séquences, on présente dans la section 2 les approches proposées dans le domaine de la fouille de séquences et nous détaillons l'algorithme de fouille de séquences (**CSMA**) développé dans le cadre de ce travail. Cet algorithme se base sur des contraintes liées à la fréquence minimale des motifs, la longueur minimale et maximale des motifs, la taille des *gaps* autorisés par motif et le nombre de *gaps* autorisés par occurrence de motif. L'utilité des deux derniers critères est mise en évidence dans la section 2.5. A la fin, dans la section 3, nous présentons un cas d'utilisation des motifs musicaux dans un contexte d'identification d'auteurs. Le manuscrit se termine par une conclusion et des perspectives.

Chapitre 1

Processus d'encodage d'une partition : de la reconnaissance de primitives musicales à leur encodage

1.1 Introduction

La reconnaissance optique de symboles musicaux numérisés intégrée dans les systèmes d'OMR (Optical Music Recognition) traite de leur conversion sous un format numérique lisible. Un symbole musical peut être représenté selon différents niveaux de décomposition comme illustré à la figure 1.1. Le niveau le plus basique est celui des patches visuels (cf. figure 1.1a) où un symbole musical est caractérisé par des configurations (ou arrangements) de pixels. Le second est la représentation par graphèmes qui consistent en des formes graphiques très élémentaires telles que des traits, des inclinaisons, des cercles séparés entre eux par des points de jonctions ou de changement d'orientation [Daher et al., 2010]). Le symbole dans ce cas est constitué d'une combinaison de graphèmes (cf. figure 1.1b). Le troisième niveau de représentation est celui des primitives musicales définies comme l'entité musicale la plus basique qui constitue une note de musique, voir figure 1.1c. A la différence du graphème, la primitive présente un niveau sémantique supérieur, les primitives forment les segments nommés (tête de note, hampe, ligature, etc.). Le dernier niveau de représentation est celui



(a) Patches visuels (b) Graphèmes

(c) Primitives (d) Notes simples et Notes composées

FIGURE 1.1 – Les différents niveaux de décomposition des symboles musicaux au sein de partitions manuscrites

des notes simples et des notes composées où un symbole représente une note de musique (une noire, une croche, clef, etc.).

Cette diversité de niveaux de représentations est née de la multiplicité des approches de reconnaissance de symboles par OMR pour des partitions manuscrites aujourd'hui inaccessibles sous forme numérique [Pardoen, 2012]. L'étape de reconnaissance des symboles musicaux et de leurs compositions occupe une place centrale dans le processus de dématérialisation de ces supports papiers. On pourrait même la qualifier de goulot d'étranglement : en effet, si les techniques de transcription échouent, aucune exploitation de ces sources ne peut réellement être envisagées et les morceaux perdent leur intérêt. Bien que les techniques existantes aient permis d'obtenir d'excellents résultats pour les symboles musicaux imprimés, les symboles musicaux manuscrits restent quant à eux très mal reconnus. Il reste encore beaucoup à faire pour la recherche de solutions robustes, car elles s'accompagnent de plusieurs défis encore non résolus : la prise en compte de la dégradation de l'écriture (sur les supports anciens), le biais et la non-uniformité des symboles (écritures parfois non standards), l'inconstance de précision dans l'exécution des traits.

Notre contribution couvre une sous-partie de l'analyse de partitions manuscrites dans

un OMR complet reposant sur un schéma de transcription simplifié et des modèles de classification pour la détection et la reconnaissance de primitives musicales. Rappelons que le schéma complet d'un OMR repose sur trois grandes phases : (1) l'étape de reconnaissance de symboles musicaux ; pour ce faire, les lignes de portée doivent être enlevées et les primitives détectées. (2) l'étape de reconstitution de l'information musicale : exploitant généralement des règles graphiques et syntaxiques permettant de surmonter les erreurs de classification de la première étape et (3) la construction d'un modèle de notation musicale pour sa représentation qui peut revêtir plusieurs formes : partition MIDI, partition XML, autre format numérique audio...etc.

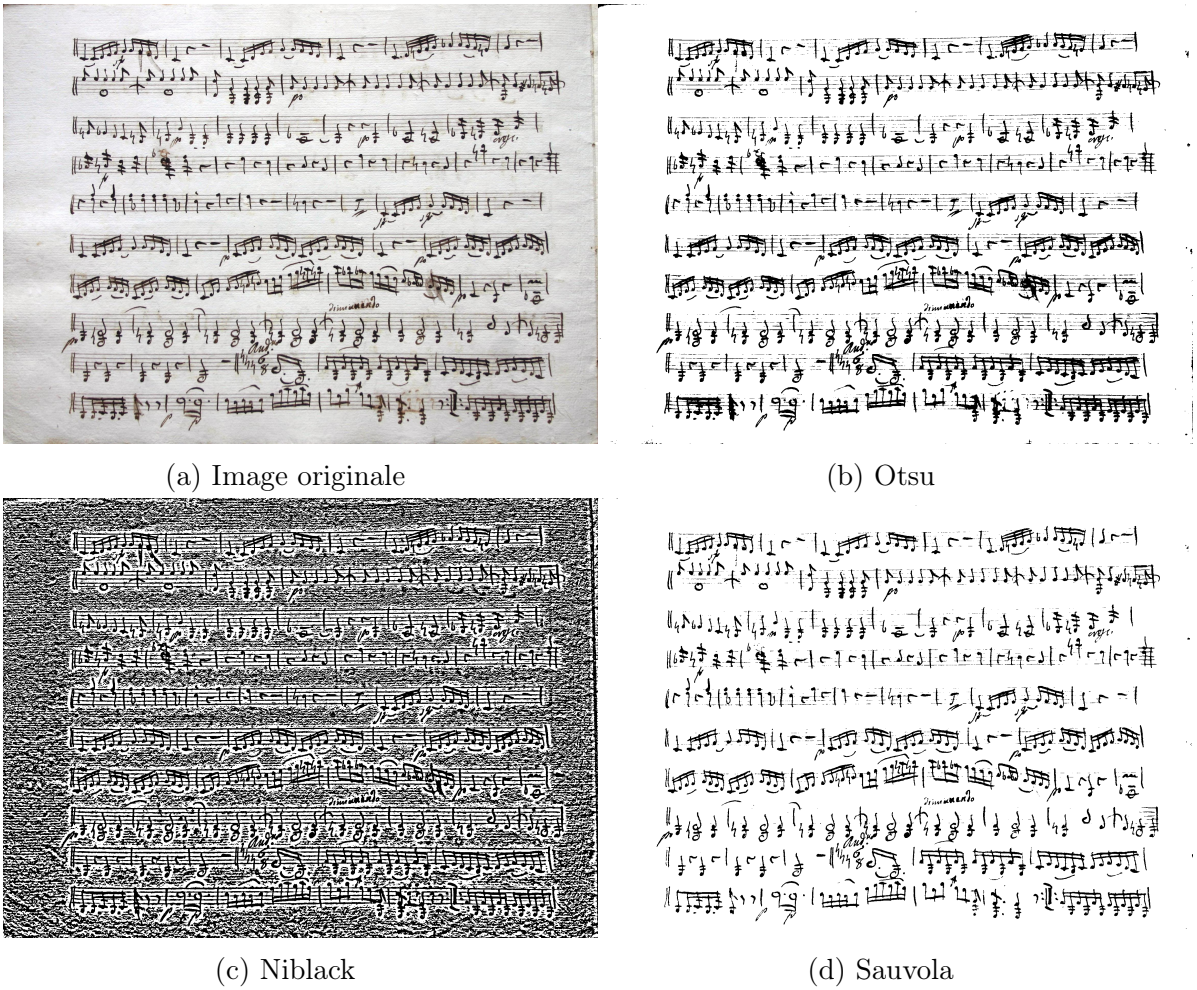
Notre schéma simplifié ne cherche pas seulement à produire une transcription complète respectant les règles de constructions sémantiques des nombreuses combinaisons mélodiques et rythmiques, mais vise également la construction d'une séquence exploitable respectant la temporalité des notes jouées et ne nécessitant pas toute l'abondance des signes musicaux fournis par le compositeur (incluant notamment les indications marginales textuelles sur la façon de jouer).

Notre schéma repose ainsi sur trois grandes étapes, à savoir le prétraitement de l'image et la détection et la suppression des lignes de portée, la détection et la classification des primitives musicales et pour finir l'encodage des symboles musicaux en vue de la transformation de la partition en une séquence textuelle.

1.2 Les étapes de préparation des images

1.2.1 Séparation fond-forme pour la détection de symboles musicaux

La phase de prétraitement d'une partition musicale numérisée en vue de la reconnaissance des symboles musicaux consiste en différentes opérations d'amélioration (voir de restauration) du rendu de la partition. Ces opérations permettent aux méthodes de détection et de reconnaissance de symboles de musique de fonctionner plus efficacement. On compte plusieurs



(a) Image originale (b) Otsu

(c) Niblack (d) Sauvola

FIGURE 1.2 – Binarisation des partitions musicales

approches de transformations dont nous ne présentons ici que les plus courantes de l'état de l'art, car elles ne constituent pas un enjeu ciblé de la thèse.

Les techniques de traitement d'image, telles que l'amélioration de l'image, l'élimination du bruit, le redressement et la binarisation, peuvent être appliquées pour améliorer la qualité de l'image et isoler les informations utiles. En fonction de la qualité des images en entrée, cette étape peut affecter le reste du processus.

La binarisation joue un rôle clé dans le processus d'analyse des documents en particulier des partitions musicales. Elle permet non seulement de réduire la complexité des tâches de reconnaissance, mais elle est également au cœur d'un grand nombre de procédures impliquant notamment des opérations morphologiques, la détection des composantes connexes ou la localisation de régions d'intérêt. De nombreuses méthodes ont été proposées pour accomplir cette tâche. Ci-dessous nous en aborderons les principales d'entre elles. Toutefois, il faut savoir

qu'il est souvent difficile d'obtenir de bons résultats de binarisation du fait du grand nombre d'irrégularités présentes dans les documents. Les partitions musicales anciennes constituent un cas d'étude intéressant du fait notamment du changement d'état du support papier, la dégradation des encres et l'usage de divers instruments d'écriture.

La méthode d'Otsu est un algorithme de binarisation bien connu basé sur le seuil global. Les pixels de l'image sont classés en deux classes, C_f et C_g, qui représentent l'avant-plan et l'arrière-plan, en fonction du seuil TOtsu. La valeur du seuil est définie pour minimiser l'écart entre deux distributions.

$$T_{Otsu} = argmax_T \frac{\sigma_B^2}{\sigma_W^2} \tag{1.1}$$

où σ_B et σ_W représentent respectivement la variance intra-classe et la variance entre classes. L'application directe de l'algorithme d'Otsu sur l'image de la partition musicale de la figure 1.2a est donnée dans la figure 1.2b. On remarque que la méthode est globalement efficace pour séparer le fond et l'information musicale. Cependant, dans certaines parties de l'image les lignes de portée sont confondues avec l'arrière plan. De plus, quand il y a une grande densité dans le cas des notes composées, les ligatures ne sont pas lisibles comme on peut le constater en bas à droite de l'image de la figure 1.2b.

Puisque la méthode d'Otsu utilise le seuil global, elle est sensible dans le cas de la variance locale. Plusieurs algorithmes de binarisation locale sont observés dans l'état de l'art. L'algorithme Niblack calcule un seuil au niveau pixels dans chaque fenêtre rectangulaire en utilisant la moyenne et la variance des valeurs de gris dans la fenêtre,

$$T_{Niblack}(x, y) = \mu(x, y) + k\sigma^2(x, y) \tag{1.2}$$

où $\mu(x, y)$ et $\sigma(x, y)$ sont respectivement la moyenne locale et l'écart-type d'une fenêtre locale centrée sur un pixel et k est une constante fixée à -0,2. La taille de la fenêtre est définie pour couvrir au moins un à deux caractères dans les images du document. Dans la figure 1.2c, on présente le résultat de l'application de Niblack sur l'image de la partition de la figure 1.2a. On constate qu'il y a beaucoup de bruits dans l'arrière plan de l'image et que

l'information musicale n'est pas très lisible.

Dans la même famille de méthodes générales de binarisation, Sauvola et Pietikainen ont proposé un nouveau seuil pour ce problème en utilisant une hypothèse sur le niveau de gris du texte et des pixels de fond :

$$T_{Sauvola}(x,y) = \mu(x,y) + \left((1 - k_{sauvola}\left(1 - \frac{\sigma^2(x,y)}{R}\right)\right) \tag{1.3}$$

où $\mu(x,y)$ et $\sigma(x,y)$ sont les mêmes que dans la méthode de Niblack et R est la dynamique de l'écart type fixée à 128. Le résultat d'utilisation de Sauvola est donnée dans la figure 1.2d.

En 2014, Lazzara et Geraud ont proposé un système de binarisation multi-échelle basée sur la méthode de Sauvola (Sauvola MS) afin de traiter les documents composés de composantes de tailles variables [Lazzara and Géraud, 2014]. Par la suite, des adaptations ad-hoc ont été apportées dans ce domaine afin de contribuer à une amélioration des techniques classiques par l'ajout notamment d'étapes supplémentaires de filtrage (filtre passe-bas de Wiener), l'utilisation du contraste local seul ou combiné au gradient local de l'image, permettant ainsi de neutraliser les effets de la dégradation des documents anciens [Su et al., 2012b].

Des méthodes basées sur des approches variationnelles ont également été proposées dans ce contexte, telles que la méthode de Howe [Howe, 2011] [Howe, 2013] qui exploite l'opérateur laplacien pour estimer une probabilité locale d'étiquetage des pixels d'avant et d'arrière-plan, qui exploite l'opérateur de Canny assurant une très bonne détection des discontinuités. La tâche de binarisation est alors formulée comme un problème de minimisation d'énergie. Très récemment, cette méthode a été reprise pour transformer linéairement l'image en une surface sphérique dans laquelle les concavités correspondent au premier plan de l'image originale. Ces concavités sont estimées à l'aide de l'opérateur Hidden Point Removal [Katz et al., 2007], qui produit une probabilité pour chaque pixel d'appartenir à une concavité. Cette représentation stochastique est ensuite utilisée comme image d'entrée pour la binarisation selon Howe. Cette stratégie a été proposée par Kliger et Tal dans le cadre de la compétition sur la binarisation d'images de document proposée à la conférence internationale ICFHR 2016 [Pratikakis et al., 2016], à l'occasion de laquelle ils ont été classés premiers.

Alors que les méthodes de l'état de l'art font encore état de processus locaux ou globaux sans apprentissage (méthode d'Otsu, Sauvola, Niblack), et ont toutefois été couronnées de succès dans leur contexte, d'autres méthodes plus récentes exploitent désormais des techniques d'apprentissage automatique pour aider le système à s'auto-adapter aux variations de niveaux de gris selon la catégorie de document traité.

Les techniques d'apprentissage supervisé ont ainsi été étudiées pour résoudre les situations les plus complexes de binarisation. L'approche basée sur la classification consiste généralement à interroger chaque pixel de l'image, à en extraire une caractéristique et à utiliser un algorithme d'apprentissage supervisé pour émettre une hypothèse pour chacun d'eux. On retrouve essentiellement dans ce contexte les méthodes fondées sur le perceptron multicouche et toute une famille de méthodes autour des réseaux de neurones [Kefali et al., 2014]. Au sein de cette famille, un nombre considérable de travaux se sont tournées vers l'utilisation des réseaux neuronaux convolutifs (CNN). On mentionnera les travaux de Pastor-Pellicer et al [Pastor-Pellicer et al., 2015] qui se sont avérés très performants pour résoudre des tâches de binarisation d'images de documents ne contenant pas nécessairement d'informations textuelles [Calvo-Zaragoza et al., 2017b]. La principale limitation de ces méthodes vient de leur grande complexité calculatoire et de l'indépendance de l'étiquetage des pixels vis-à-vis du contexte ou du voisinage. L'approche de la classification au niveau pixels est néanmoins très largement poursuivie dans les travaux récents de binarisation aidé d'un apprentissage supervisé. On trouvera notamment dans [Calvo-Zaragoza and Gallego, 2019], une approche originale de binarisation utilisant les CNN et qui impliquent des architectures multicouches pour effectuer une série de transformations (convolutions) du signal d'entrée. Leur modèle est conçu pour assurer le mappage patch à patch de l'image vers sa version binarisée correspondante. Chaque pixel en sortie du réseau reçoit une valeur d'activation différente selon qu'il est ou non reconnu comme pixel de l'arrière-plan ou du premier plan.

Des modèles entièrement convolutifs ont déjà été utilisés pour la segmentation sémantique [Long et al., 2015]. Les auto-encodeurs ont été quant à eux initialement considérés pour résoudre des applications de débruitage d'image [Vincent et al., 2010], qui se modélise selon une formulation similaire à celle de la binarisation de documents (la couleur étant considé-

rée comme le bruit de l'entrée). L'utilisation de ces modèles semble tout à fait prometteuse pour la binarisation d'images de documents et de partitions musicales dégradées. Les bases Salzinnes Antiphonal Manuscript [1] et Einsiedeln Stiftsbibliothek [2] ont été utilisées dans leurs travaux produisant des taux de binarisation dépassant très largement les méthodes classiques (sans apprentissage) de l'état de l'art.

La binarisation ne permet pas de supprimer les notes des lignes de portées directement tandis que l'analyse couleur des partitions numérisées en couleurs permettent de construire des modèles de segmentation 'fond/forme/ligne de portée'. L'analyse couleur peut ainsi se charger de l'ensemble de ces étapes dans un pipeline unifié.

1.2.2 La détection et la suppression des lignes de portée

La détection et la suppression des lignes de portée sont des tâches fondamentales dans la plupart des systèmes d'analyse d'images numérisées de partitions musicales. Ce prétraitement permet d'isoler les primitives ou les symboles musicaux afin d'appliquer des méthodes de reconnaissance de symboles. Certaines approches récentes ne nécessitent plus systématiquement cette suppression (voir section 1.4.4 qui correspond à l'approche que nous avons exploitée dans cette thèse). Dans notre cas d'étude, la détection des lignes de portée est nécessaire pour l'encodage de l'information musicale et la construction de la séquence (voir section 1.5).

Bien que la détection et la suppression de lignes de portée puissent être perçues comme une tâche simple, il est souvent difficile d'obtenir des résultats corrects. Ceci est principalement dû à des déformations de l'image de la partition telles que des discontinuités, ou des dégradations du document numérisé (en particulier dans les documents anciens). Par ailleurs, les documents musicaux étant très hétérogènes, il est difficile de développer des méthodes capables de traiter tout type de partitions. Une étude complète sur les premières tentatives envisagées pour cette tâche peut être consultée dans les travaux de Dalitz et al. [Dalitz et al., 2008]

Ensuite, de nombreuses autres méthodes ont été proposées. Dos Santos Cardoso et al.,

1. https://smu.ca/academics/archives/the-salzinnes-antiphonal.html
2. https://www.kloster-einsiedeln.ch/stiftsbibliothek/

dans [Guedes et al., 2009], ont proposé une méthode qui considère les lignes de portée comme des chemins de connexion entre les deux bordures de la partition. Dans ce cas, la partition musicale est modélisée sous forme de graphe construit d'une façon itérative par des opérations d'érosions pour que la détection de portée revienne à la détection de stables dans le graphe. Cette stratégie a été améliorée et étendue pour être utilisée sur des images de partitions en niveaux de gris [Rebelo and Cardoso, 2013]. On trouve enfin dans la littérature un grand nombre d'approches géométriques et morphologiques de détection de lignes de portées. On peut citer les travaux de Dutta et al. [Dutta et al., 2010] qui ont mis au point une méthode qui considère le segment de ligne de portée comme une connexion horizontale de pixels noirs alignés verticalement de hauteur uniforme, validée en fonction des pixels voisins. Dans les travaux de Piatkowska et al. [Piątkowska et al., 2012], un algorithme d'intelligence en essaim est appliqué pour détecter les lignes de portée. Su et al. dans [Su et al., 2012a] ont estimé les propriétés de hauteur et d'espace interligne des portées afin de prédire la direction des lignes et d'émettre des hypothèses sur les portées en les ajustant ensuite. Enfin, Géraud dans [Géraud, 2014] a développé une méthode impliquant une série d'opérateurs morphologiques directement appliqués à l'image de la partition pour supprimer les lignes de portée.

Plus récemment, dans [Calvo-Zaragoza et al., 2017a], les auteurs ont utilisé les réseaux de neurones à convolution, qui ont démontré une performance exceptionnelle dans les tâches de traitement d'images. Dans cette approche, le réseau proposé, nommé *StaffNet*, consiste en deux couches de 32 filtres de taille 3×3 plus une fonction de *max-pooling* de 2×2, suivie par une couche à convolution de 64 neurones. La fonction d'activation étant la fonction *tanh*. Les patches en entrée du réseau sont de taille 21×21, centrée sur le pixel à classifier comme faisant partie de la portée ou à un symbole. Ce réseau est entraîné à l'aide d'un jeu de données contenant des paires de partitions avec et sans lignes de portée. Leur modèle a montré son efficacité sur des images binaires et en niveaux de gris. Cette approche surclasse aujourd'hui les autres approches proposées de l'état de l'art avec un taux de reconnaissance global de 99% sur les données bruitées de MUSCIMA (cf. Annexe B.2).

Une information couleur est généralement porteuse de sens : lignes de portées, notes, corrections et surcharges, informations rythmiques etc. Dans les partitions de notre collection,

il nous a semblé nécessaire de tirer profit de cette information compte tenu des différences observables et quantifiables de couleurs présentes dans les encres (lignes de portées et écritures manuscrites). Ici, une étape de binarisation entraîne mécaniquement la perte de ces informations sémantiques liées à la couleur et nécessite des traitements dédiés de suppression des lignes de portées tels que rappelés précédemment.

L'analyse colorimétrique des images de documents couleurs consiste à séparer les informations présentes sur ce document en différentes couches, présentant une unité sémantique. Ce type d'analyse permet de restituer des niveaux d'informations plus nombreux, à la différence de la binarisation qui supprime toute sémantique au-delà de la séparation fond-forme et à la différence des méthodes de séparation en lignes de portée qui nécessitent des a priori importants sinon un apprentissage coûteux sur leur aspect visuel et leur régularité. Dans le cas des partitions de musique anciennes (les partitions de la base Massonneau de notre projet), on peut constater que l'image de la partition n'est pas simplement composée de deux niveaux d'informations noir et blanc, comme cela peut apparaître visuellement sur le papier. Sur ces documents, les lignes de portée possèdent un niveau de couleur d'encre différente, ceux-ci ayant été imprimés en amont de l'écriture musicale et des autres surcharges manuscrites faites par les auteurs. Notons également que la distribution globale des couleurs, ainsi que leur contraste, peuvent perturber le calcul d'un seuil dans le cas de la binarisation (cas des partitions dont l'arrière-plan est assombri par des ombres ou des tâches). Là où la segmentation binaire peut échouer aux points de jonction entre les notes et les lignes de portées, l'analyse couleur peut également aider à réduire ces phénomènes d'inclusion.

L'analyse colorimétrique, en permettant de généraliser la séparation en couches d'informations (à autant de couches de couleurs homogènes que souhaitées), est une alternative avantageuse à un grand nombre de prétraitements des images de partitions.

La segmentation colorimétrique repose sur une analyse de distribution des couleurs dans leur espace couleur en vue de le partitionner. Les méthodes de segmentation se basent sur des hypothèses de regroupement des pixels en sous-ensembles ayant des caractéristiques couleurs proches selon une métrique donnée. L'historique des méthodes de segmentation couleur est très riche et ne fera pas l'objet d'une étude dans la thèse, cependant dans le cas des documents

on peut relever certaines tendances méthodologiques liées aux contraintes de taille des objets, leur contraste et leur organisation.

L'analyse colorimétrique des images de documents repose sur des méthodes qui peuvent être essentiellement regroupées en quatre grandes classes : la construction de régions, qui repose sur des méthodes ascendantes par croissance de région ou des méthodes structurelles de type division-fusion de régions (structure de tétra-arbre) ; les approches contour et coopération région-contour ; les approches markoviennes de la séparation en couches couleurs ; les approche mixtes (mélanges de plusieurs familles de méthodes) et enfin les méthodes de classification pixellaire qui consistent à affecter chaque pixel de l'image à la classe de la couche couleur qui le représente selon son apparence colorimétrique. Dans le cadre des images de documents couleurs, il n'existe aucune étude visant la segmentation des documents en couches couleurs sans connaissance a priori du nombre de classes. La plupart des travaux existants visent une classification supervisée (séparation de texte, séparation fond verso et recto, séparation chromatique/achromatique, classification en super pixels).

A la base, une classification de pixels couleurs repose uniquement sur l'information colorimétrique et ne tient pas compte de l'aspect spatial. Certaines techniques intègrent aussi l'information spatiale pour segmenter l'espace des couleurs, et se regroupe dans les méthodes de classification spatio-colorimétrique.

La littérature présente en général un grand nombre de méthodes de classification de pixels, le plus souvent destinées aux images couleurs naturelles. Ces techniques sont généralement considérées comme prétraitement permettant de simplifier la tâche de récupération des composantes connexes par une simple croissance de régions. Parmi les méthodes de séparation en couches couleurs sans apprentissage, on observe une utilisation assez systématique de la classification par MeanShift [Hong et al., 2007] [Zheng et al., 2009] non supervisée basée sur la densité dans l'espace des couleurs. Une variante de la méthode de classification par MeanShift a été conçue dans [Wang et al., 2006]. Il s'agit d'une méthode adaptative tenant compte du voisinage spatial. C'est cette méthode que nous avons adaptée ici, reprise des réalisations produites pour le projet d'investissement d'avenir PIXL et facilitant la détermination du nombre de classes a priori [Ho et al., 2016].

— Étape 1 : Formation des régions colorimétriques par filtrage spatio-colorimétrique basé sur le Mean-Shift La méthode Mean Shift proposée au départ par Fukunaga [Fukunaga and Hostetler, 1975] est un estimateur du gradient de densité non paramétrique exploité récemment dans le cadre de traitement d'image par Comaniciu [Comaniciu et al., 2001]. Elle a l'avantage de ne reposer sur aucun a priori de la distribution des intensités des pixels ou du nombre de couches couleurs. L'algorithme du Mean-Shift peut être appliqué globalement sur tous les pixels de l'image en utilisant uniquement le voisinage colorimétrique à chaque pixel courant définit dans un rayon Rv qui représente la principale information pour l'estimation globale de l'amplitude colorimétrique correspondant au gradient minimum d'un contour et définie dans un rayon Rc, ou enfin en utilisant un voisinage mixte spatio-colorimétrique. Nous avons constaté en élaborant ces deux types de voisinage que l'exploitation d'un voisinage mixte amorti d'une façon significative le bruit (de numérisation, de compression) et les dégradations du papier et des caractères tout en préservant les contours.

— Étape 2 : Croissance des régions sur le résultat de la première étape ; Dans une seconde étape on applique une croissance de régions, que nous avons développée directement sur le résultat de la première étape. L'objectif est de faire progressivement grossir les régions autour de leur point de départ (point germe). Il s'agit dans cette étape de parcourir les points germes marqués durant la première étape de telle manière à cibler des zones homogènes dans l'image puis à faire grossir les régions par agglomérations des pixels voisins. Cette étape de croissance utilise une distance (nous avons utilisé la distance euclidienne) dans le plan couleurs pour choisir les pixels à agglomérer (l'espace utilisé est l'espace LuV). La croissance s'arrête lorsqu'on ne peut plus ajouter de pixels sans briser l'homogénéité.

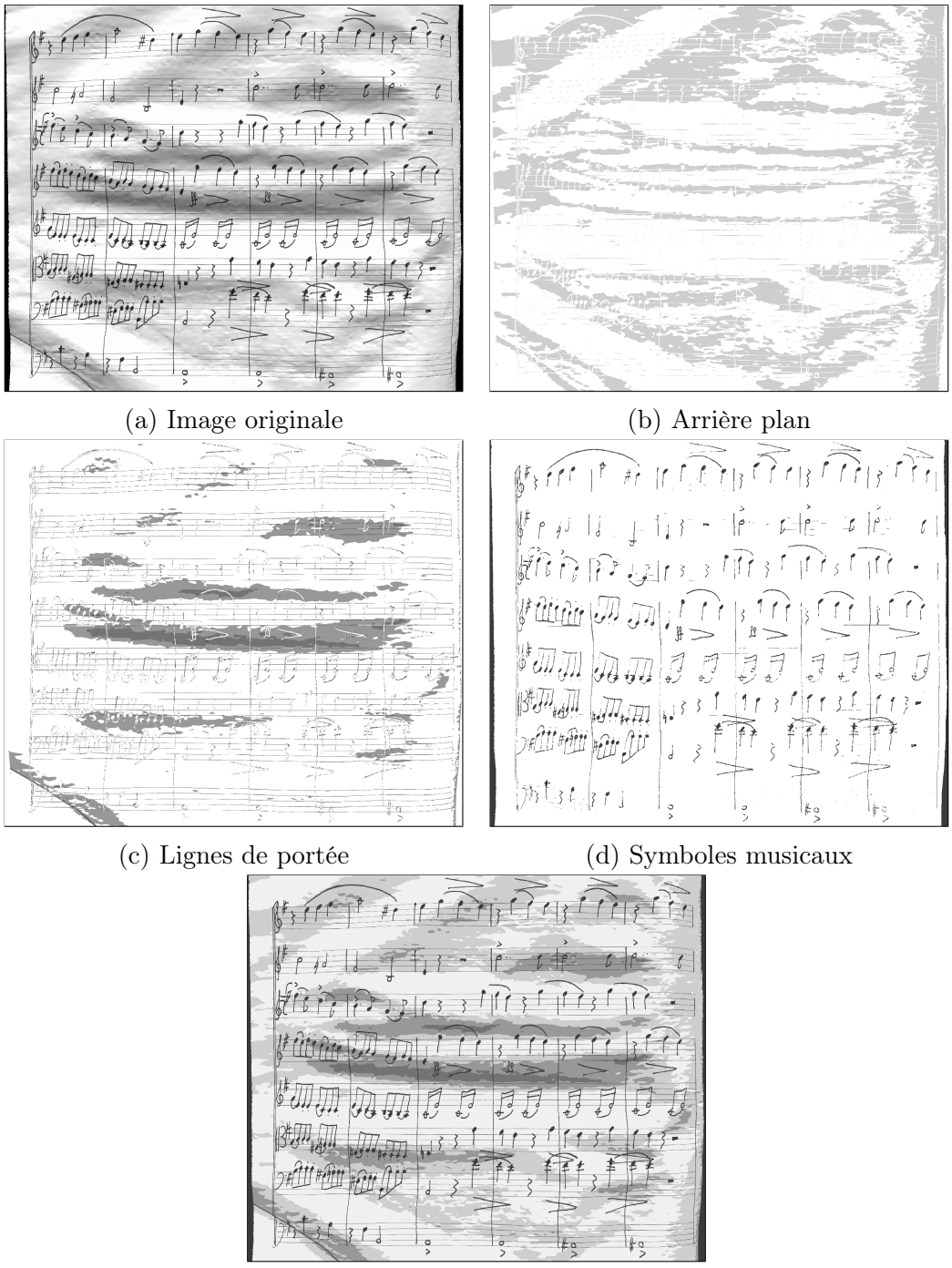
— Étape 3 : Récupération des couches couleurs : Cette dernière étape de segmentation permet de regrouper les couleurs des régions de l'étape précédente en classes couleurs définitives qui représenteront les couches couleurs disponibles dans le document. Chaque région d'une couche couleur doit avoir de fortes ressemblances avec les autres régions de la même couche, et doit être différente des régions des autres couches cou-

leurs. Pour effectuer cette tâche nous avons proposé une approche de classification basée sur un k-means à 3 ou 5 classes. Une partie d'apprentissage léger permet d'introduire cette connaissance dans le système au moment du regroupement en classes final. Cette étape d'apprentissage est réalisée une seule fois sur une image représentative. Un opérateur choisi un nombre de zones candidates significatives en indiquant le nombre k de classes souhaitées dans cette zone. Pour chaque zone une classification en k classes est réalisée produisant des centroïdes de classes qui sont stockés et qui peuvent être fusionnés (en fonction de leur proximité colorimétrique). L'ensemble des autres documents de la collection sont ensuite traités avec le même nombre de classe à partir du positionnement de ces germes. Dans le cas des partitions de musique, nous nous sommes intéressés à une classification allant de 3 classes (fond, notes, lignes de portées) à 5 classes (fond, notes, lignes de portées, bruit papier fond, surcharges manuscrites d'encre différente).

La figure 1.3 illustre un résultat de segmentation en 3 classes sur la base de cette méthodologie sur une image d'entrée en niveau de gris, prenant comme valeur Rc=5 et Rv=8 dans l'espace Luv.

La figure 1.4 illustre un résultat de segmentation en 3 classes sur la base de cette méthodologie sur une image couleur de Massonneau en considérant les mêmes paramètres.

Dans notre travail, la détection des lignes de portée est nécessaire pour le processus d'encodage. Néanmoins, la détection des lignes au pixel près ne correspond pas à notre objectif majeur. En effet, rappelons ici que la localisation des lignes de portée est nécessaire pour l'étape d'encodage (resituer les notes et les altérations par rapport à l'indice de ligne), c'est la raison pour laquelle la méthode que nous utilisons sur les partitions du projet est basée sur cette décomposition colorimétrique. L'évaluation au pixel n'a pas été détaillée ici, puisque les résultats sur lesquels nous agissons sont les centroïdes de projections verticales de la couche des lignes de portée (voir figure 1.4c) qui nous fournissent la numérotation nécessaire à l'encodage (voir section 1.5).



(a) Image originale

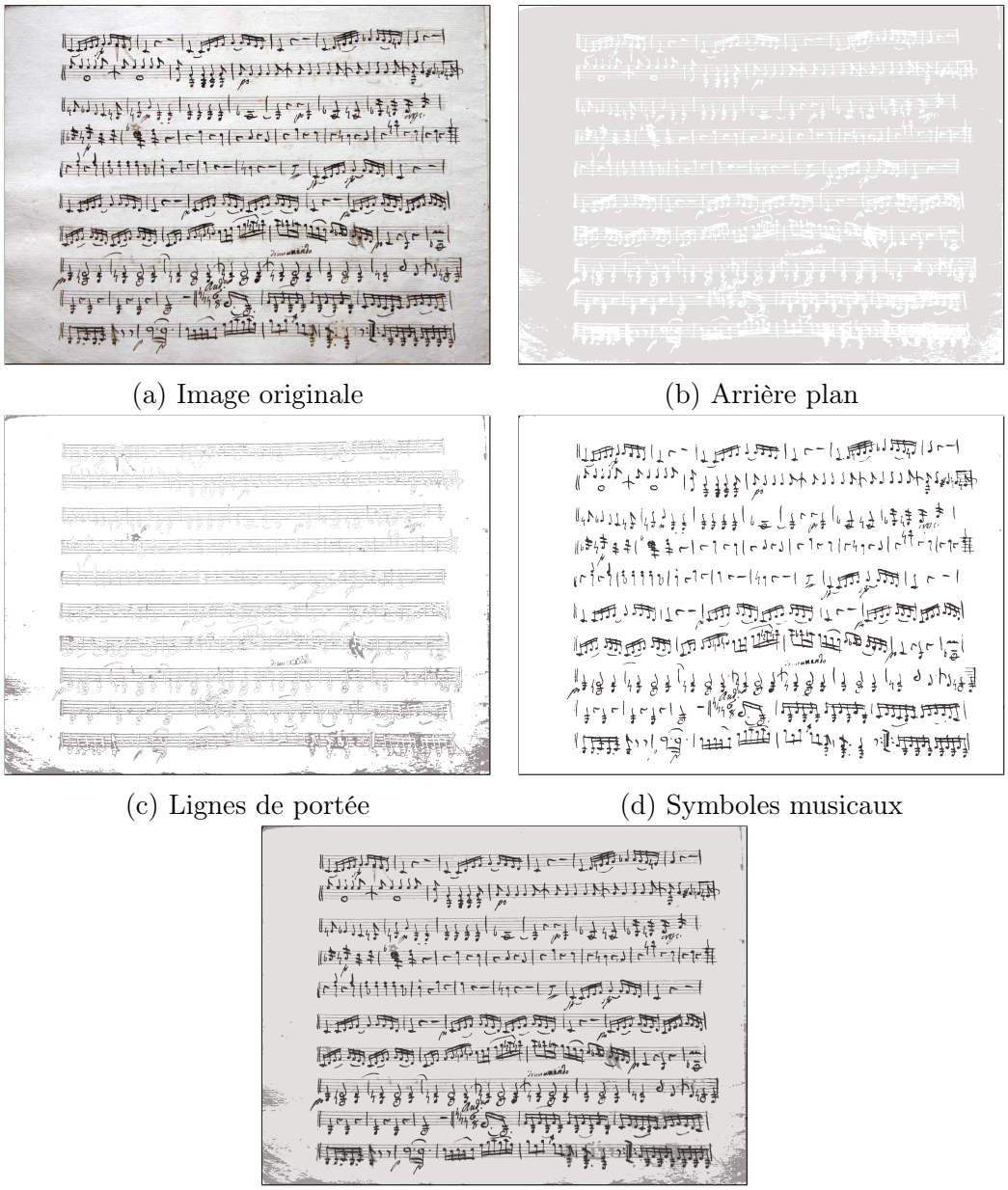
(b) Arrière plan

(c) Lignes de portée

(d) Symboles musicaux

(e) Fusion des trois couches

FIGURE 1.3 – Résultats d'une séparation en couches niveaux de gris sur une image de la compétition MUSCIMA



(a) Image originale

(b) Arrière plan

(c) Lignes de portée

(d) Symboles musicaux

(e) Fusion des trois couches

FIGURE 1.4 – Résultats d'une séparation en couches couleurs sur une image d'une partition de Massonneau

1.3 Les étapes de détection et de classification d'objets musicaux (primitives et symboles) : état de l'art

Au cours des vingt dernières années, différentes approches ont été proposées pour la reconnaissance des symboles musicaux manuscrits. Les difficultés de cette tâche résultent des problèmes liés à la qualité de la numérisation de la partition musicale et les dégradations causées par les étapes précédentes du pré-traitement en particulier après la suppression des lignes de portées. En outre, la présence des formes incomplètes de symboles, les superpositions, les irrégularités des tailles et des formes augmentent la complexité de l'opération.

Ana Rebelo, dans [Rebelo et al., 2010], a proposé une architecture générique permettant l'extraction puis la reconnaissance de symboles de musique. Les opérations d'extraction et de segmentation reposent sur des règles basées sur des opérations morphologiques. Ensuite, la phase de classification des symboles repose sur un modèle de classification. Dans ses travaux Rebelo exploite les modèles de Markov et les réseaux de neurones pour résoudre cette tâche. Ce type d'architecture est désormais assez commun, il se base sur plusieurs travaux antérieurs nécessitant des étapes préalables de segmentation (cf. section 1.3.2). Les travaux les plus récents, comme nous le verrons à la section 1.3.1 permettent désormais de s'affranchir d'une segmentation parfois très risquée car produisant de nombreuses erreurs qui conduisent à des cascades d'inexactitudes sur la série de traitements engagée pour la reconnaissance (cf. section 1.3.1).

1.3.1 Les approches de classification sans segmentation des symboles musicaux

Dans les documents de symboles musicaux dessinés à la main, la régularité et la répétabilité de styles d'écriture co-occurrents constituent une source d'informations très riche.

La première étape consiste en la localisation des informations saillantes réparties sur toute la surface de la page manuscrite. Une des approches proposées utilise une méthode de classification en considérant les patchs visuels, basées sur l'extraction de points visuels saillants, comme vecteur caractéristique [Oikonomopoulos et al., 2005].

Certaines alternatives se basent sur les caractéristiques des patchs visuels pour éviter une segmentation de symboles fastidieuse et contraignante, [Jurie and Triggs, 2005] [Fang et al., 2017]. Les patchs visuels sont des régions présentes dans une image qui sont généralement utilisées en vision par ordinateur pour éviter les comparaisons d'image pixel par pixel ou tout type de segmentation. Les comparaisons des patchs visuelles doivent être effectuées sur toutes les échelles d'une manière pyramidale. La description d'image basée sur des patchs visuels peut être utilisée pour la reconnaissance d'objet, la catégorisation d'objet et la classification complète de l'image (y compris la classification manuscrite des documents) [Fornes et al., 2011]. Généralement, les patchs visuels sont regroupés dans des histogrammes et le processus de classification est basé sur la comparaison de ces histogrammes [Coupé et al., 2011].

Dans le cas d'image de hautes résolutions, certaines approches tentent de créer des patches visuels à l'aide d'un ensemble de points-clés extraits de descripteurs (par exemple, SIFT, SURF) [Csurka et al., 2004] [Bay et al., 2008]. La limite des patchs visuels réside dans le manque d'information spatiale des positions des patchs. Ceci peut être une source d'erreur et de confusion, dans le contexte musical, entre les blanches et les bémols, les ligatures et les liens et tous les symboles qui ont une forme similaire (après rotation ou inversion par exemple).

1.3.2 Les approches de reconnaissance de symboles basée sur une segmentation supervisée de partitions

Ana Rebelo, [Rebelo et al., 2010], a montré que les modèles de réseaux de neurones sont plus performants que les HMM pour la tâche de reconnaissance de symboles musicaux testés sur leur propre jeu de données (qui consiste en un ensemble de notes isolées manuscrites et imprimées avec des déformations). C'est ainsi que ces dernières années, les réseaux de neurones convolutionnels (CNN) ont surclassé la plupart des systèmes de classification pour la reconnaissance des caractères [Chen et al., 2015]. Les chercheurs travaillant sur la reconnaissance de symboles musicaux manuscrits se sont tournés vers ces approches. En effet, les architectures à base de neurones sont capables de calculer des caractéristiques à partir des données brutes. Cependant, ils doivent être conçus selon des schémas de réseau très spécifiques, trai-

tant notamment du nombre de couches, de la disposition des couches et de l'initialisation des hyper-paramètres (taille du réseau, nombre de neurones par couche, fonctions d'activation, etc.).

Dans les deux sous-sections suivantes, nous présentons deux types d'approches de reconnaissance de symboles : la première repose sur une classification supervisée avec une étape préalable de segmentation basée sur des heuristiques morphologiques (et des descripteurs pré-calculés manuellement sur les symboles segmentés) ; la seconde approche concerne les méthodes plus récemment développées n'exploitant aucune heuristiques autour de techniques de classification supervisée reposant sur un apprentissage de représentation par réseaux de neurones. Ces derniers permettent à la fois de localiser les symboles et de les classer dans une série d'opérations exploitant la même architecture. Nous aborderons enfin en section 1.4 les méthodes permettant de traiter à la fois la détection et la reconnaissance sans segmentation préalable autour des réseaux de dernières générations les RPN.

1.3.2.1 Segmentation basée sur des descripteurs morphologiques

La taille et la forme des symboles musicaux rendent la tâche de classification très complexe (par exemple la différence entre une noire et une croche n'est qu'une ligature). Ainsi et afin de réduire la complexité et de disposer d'un petit nombre de classes, certains travaux ont tenté de réaliser une segmentation basée sur des primitives et de considérer chaque bloc de symboles comme une classe. Cette segmentation est basée sur le fait qu'une note de musique est composée de primitives (par exemple, une note de type croche est composée d'une tête de note pleine, d'une hampe et d'une ligature). Mais une segmentation superfine au niveau primitive est sujette à l'identification de graphèmes qui conduit à un travail considérable de recomposition afin de déduire les symboles musicaux. Le niveau de segmentation des symboles diffère d'une approche à une autre.

Dans [Fornés et al., 2007], la segmentation est effectuée à l'aide des filtres médians et d'un processus de détection de contours. Après une étape de détection et suppression des lignes de portées, les notes de tête pleines sont détectées en effectuant une ouverture morphologique avec un élément structurant elliptique. Ensuite, le fait que le processus de classification doit

prendre en compte les déformations et les variations dans les styles d'écritures, les auteurs ont utilisé la DTW (Dynamic Time Warping) qui permet d'aligner deux échantillons (séquences) de tailles différentes d'une façon optimale [Kruskal, 1983].

Dans [Rebelo et al., 2010], le processus de segmentation consiste à localiser et à isoler les symboles. Dans ce travail, les symboles sont répartis en quatre types différents :

1. Les symboles représentés par un segment vertical avec hauteur supérieure à un seuil : notes (par exemple, une noire), des notes avec des ligatures (par exemple, une croche) et des notes creuses (par exemple, une blanche).
2. Les symboles qui lient les notes : ligatures (par exemple deux notes liées sur une croche).
3. les symboles restants connectés aux lignes de portée : clefs, soupirs, altérations (par exemple, naturelles, bémol, dièse) et signature temporelle (par exemple 4/4).
4. Les symboles au-dessus et au-dessous des lignes de portée : notes, liens et des accents.

Ici le processus de segmentation est basé sur des règles définies manuellement en fonction de la hauteur des lignes et de la hauteur des espaces entre les lignes de portée. Pour la classification des segments, les auteurs ont fait des expériences en utilisant différents modèles de classification : modèles de Markov cachés (HMM), K plus proches voisins, réseaux de neurones et machines à vecteurs de support. Les machines à vecteurs de support sont plus performantes que les autres modèles avec une précision moyenne de 96 % et un écart type de 0.6 sur tous les symboles. Ce résultat est obtenu sur un ensemble de données de partitions musicales imprimées avec quelques modifications. [Rebelo et al., 2010].

Une autre approche a été proposée dans [Wen et al., 2015] qui consiste à reconnaître les symboles musicaux avec segmentation. Premièrement, les partitions sont divisées en plusieurs blocs en fonction de la position des lignes de portée. Ensuite, les composantes connexes sont extraites. Une composante connexe peut faire référence à une seule note, à une partie de note ou à un groupe de notes. De ce fait, ils font un premier niveau de classification. Ce niveau consiste en trois réseaux de neurones combinés. Ces réseaux partagent le même nombre de neurones dans les couches cachées et la couche de sortie à l'exception de la couche d'entrée (qui correspond à la taille de l'image d'entrée : 20×20, 35×20 et 60×30). La couche de

sortie est de taille 5, ce qui correspond au nombre de classes du premier niveau : hampes, ligatures, points et têtes de notes, bruit et tous les autres symboles. Le vote majoritaire est utilisé pour prendre la décision de classification.

Pour chaque type de classe, ils ont défini un ensemble de règles en fonction de la position des symboles afin de reconstruire les notes incomplètes et de prendre des décisions pour les autres. Dans le cas des notes groupées, ils ont proposé une solution pour analyser les symboles en utilisant une fenêtre glissante. Une fenêtre d'analyse est déplacée le long des colonnes de l'image afin d'analyser les segments adjacents et de ne conserver que les notes. Généralement, la hauteur des symboles n'est pas inférieure à trois fois la taille de l'espace entre les lignes et la largeur à environ deux fois la taille de l'espace entre les lignes. La fenêtre glissante passe d'abord par les colonnes, puis par les lignes. Ils ont utilisé différentes tailles de la fenêtre glissante afin de compléter les notes. Le résultat en précision de leur approche sur un jeu de données privés est d'environ 96,73 % et le rappel atteint 91,72% [Wen et al., 2015].

Bien que les méthodes présentées ci-dessus aient montré de bonnes performances sur les données sur lesquelles elles ont été étudiées, elles ont du mal à se généraliser sur tout type de données musicales contenant un certain niveau de dégradation. Les méthodes reposant sur des détecteur/classificateur de type Region-Proposal Networks (RPN) ont contribué à l'amélioration significative des techniques de reconnaissances des données symboliques (petits graphiques, caractères et symboles musicaux). En dépassant ainsi les résultats de l'état de l'art, elles deviennent aujourd'hui les outils de référence et se généralisent (ICDAR 2019).

Dans ce contexte d'approches morphologiques, nous avons choisi de produire une méthode de base (que nous nommerons la "baseline") qui permet la mise en place d'un comparatif de méthodes de segmentation de partitions en primitives musicales (voir section suivante). Cette baseline a été conçue pour assurer la détection et la classification des primitives dans les partitions manuscrites. Elle suit un scénario classique de reconnaissance d'objets, à savoir : la détection des objets, leur caractérisation et leur classification. Cette baseline repose sur une détection d'objets à partir d'une fenêtre glissante dont on extrait des caractéristiques locales dérivées des Local Binary Patterns (LBP). Bien que dans la littérature les chercheurs utilisent plus souvent le descripteur HOG [Dalal and Triggs, 2005] ou HOG combiné avec

LBP dans les approches de reconnaissance (reconnaissance faciale de Wang dans [Wang et al., 2009]), nous avons retenu le LBP pour sa capacité à exploiter les 8 directions pour chaque pixel et à détecter les patterns locaux (comparativement au descripteur HOG qui n'utilise qu'une seule direction par pixel). Le fait que les images des partitions de MUSCIMA soient en niveaux de gris, nous ont conduit à privilégier cette famille de descripteurs pour décrire la forme de chaque primitive. Pour la partie classification nous avons choisi d'utiliser un classificateur multiclasse SVM comme proposé dans [Qiao et al., 2015]. Les résultats obtenus sur cette expérience sont donnés dans la section 1.4.4.

1.3.2.2 Segmentation basée sur des architectures neuronales

Dans une nouvelle classe de méthodes, Lee et Al., [Lee et al., 2016], ont tenté de classer le symbole de musique manuscrit du jeu de données HOMUS (cf. Annexe B.1), qui a été proposée dans [Calvo-Zaragoza and Oncina, 2014], en utilisant différentes architectures de réseaux de neurones à convolution profonds. Ils étudient l'efficacité de réseaux de neurones profonds tels que CifarNet, AlexNet et GoogleNet pour la reconnaissance de symboles musicaux.

Dans [Dorfer et al., 2017], les auteurs ont utilisé un réseau de neurones à convolution pour la segmentation des têtes de notes de musique. Le réseau prend des images de partitions en entrée et prévoit une probabilité d'appartenance des pixels à une tête de note. Le réseau consiste en une série de convolutions de taille 3×3 suivies d'une normalisation (Batch Normalization) et d'unités linéaires exponentielles (ELU : Exponential Linear Units). Ils ont également remplacé la concaténation des *features maps* par '*element-wise sum skip connections*', qui fonctionne de la même manière, mais réduit le temps de calcul du fait que le nombre des *features maps* est réduit de moitié [He et al., 2016]. La sortie du réseau est calculée comme une seule *feature map* et post-traitée avec la fonction sigmoïde, produisant une probabilité d'appartenance à une note au niveau pixels.

Plus récemment, Calvo-Zaragoza et al. ont proposé une binarisation au pixel près de documents musicaux avec des réseaux de neurones à convolution [Calvo-Zaragoza et al., 2017b]. Ils ont développé une approche basée sur un réseau de neurone à convolution pour la segmentation des partitions musicales manuscrites en lignes de portée, symboles musicaux et

régions de texte [Calvo-Zaragoza et al., 2017a].

Cette tendance s'est poursuivie en 2018 par Pacha dans [Pacha et al., 2018a]. Dans leur travail, les auteurs ont utilisé la bibliothèque d'apprentissage profond TensorFlow et ont testé un ensemble de modèles de détection d'objets (appelés **R**egion **P**roposal convolutional **N**eural net abr. RPN) sur des données musicales en prenant en compte presque tout le vocabulaire musical. Le détecteur le plus performant à ce jour demeure le Faster Region proposal CNN (Faster R-CNN) utilisant l'extracteur de caractéristiques Inception-Resnet V2 pré-entraîné sur le jeu de données COCO [Szegedy et al., 2017]. Leur modèle produit une précision moyenne de 80 % sur un ensemble de cent primitives musicales visuelles du jeu de données MUSCIMA ++ (cf. Annexe B.3) [Fornés et al., 2012] [Hajič and Pecina, 2017]. Une primitive musicale est une forme graphique d'une note musicale complète (e.g. une clef, un bémol, etc.) ou une partie d'une note musicale (e.g. tête de note, hampe, etc.). D'autres architectures RPN telles que U-Net [Ronneberger et al., 2015] ont également montré des résultats très satisfaisants sur des données similaires [Pacha et al., 2018b]. En d'autres termes, leurs approches basées sur l'apprentissage profond surclassent toutes les autres méthodes de l'état de l'art. Ce type d'architecture est détaillé dans la section 1.4

1.4 Focus sur les RPN : réseaux de neurones chargés de la détection et la classification d'objets

La détection d'objets dans les scènes naturelles est une partie intégrante dans le domaine de la vision par ordinateur. L'étape de détection d'objets participe aux tâches plus complexes d'estimation de la pose (reconnaissance d'activités), de détection de véhicules, de vidéo-surveillance, et également aujourd'hui de détection de textes. La différence entre les algorithmes de détection d'objet et les algorithmes de classification réside dans le fait que dans les algorithmes de détection, les objets sont délimités par une boite englobante permettant d'ancrer la position de l'objet et ses dimensions.

La principale raison pour laquelle on ne peut pas résoudre ce problème en construisant un réseau convolutionnel standard suivi d'une couche entièrement connectée est que la longueur

de la couche de sortie est variable - et non constante, car le nombre d'occurrences des objets d'intérêt n'est pas fixe (i.e. on peut trouver plus d'un objet dans une scène). Une approche naïve pour résoudre ce problème serait de prendre différentes régions d'intérêt de l'image et d'utiliser un CNN pour classifier la présence de l'objet dans cette région. Le problème avec cette approche est que les objets d'intérêt peuvent avoir des emplacements spatiaux différents dans l'image et des rapports d'aspect différents. Pour cette raison, il faudrait sélectionner un grand nombre de régions, ce qui pourrait faire exploser les calculs. Par conséquent, des algorithmes comme RP-CNN, YOLO etc. ont été développés pour proposer des zones candidates en contournant le problème de sélection d'un grand nombre de régions.

1.4.1 Region Proposal Convolutionnal Neural Net - Approche de base (R-CNN)

Pour contourner le problème de la sélection d'un grand nombre de régions, Ross Girshick et al. [Girshick et al., 2014], ont proposé une méthode où la recherche sélective est utilisée pour extraire un nombre réduit de 2000 régions de l'image (nommées Region Proposal). Dès lors, au lieu d'essayer de classer un grand nombre de régions, ces 2000 régions sont jugées suffisantes pour couvrir tous les objets présents dans une scène. A cet effet un algorithme de recherche sélective, décrit ci-dessous, génère ces régions.

1. Générer une sous-segmentation initiale pour la génération de nombreuses régions candidates
2. Utiliser un algorithme gourmand pour combiner récursivement des régions similaires en régions plus grandes.
3. Utiliser les régions générées pour produire les propositions finales des régions candidates.

Comme illustré dans la figure 1.5, ces 2000 propositions de régions candidates sont déformées en un carré et introduites dans un réseau neuronal convolutionnel qui produit en sortie un vecteur caractéristique de 4096 dimensions. Le CNN agit comme un extracteur de caractéristiques et la couche dense de sortie se compose des caractéristiques extraites de l'image

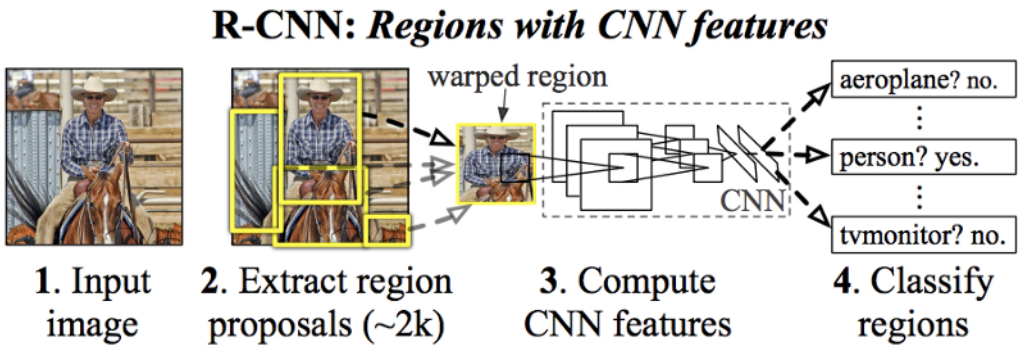


FIGURE 1.5 – Système de détection d'objets [Girshick et al., 2014]

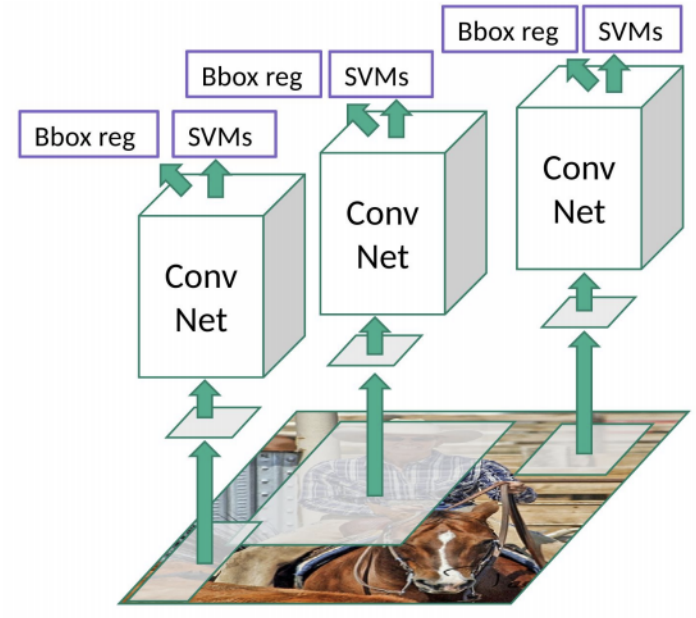


FIGURE 1.6 – Classification de régions par le R-CNN

et les caractéristiques extraites sont introduites dans un SVM pour classifier la présence de l'objet dans cette proposition de région candidate. En plus de prédire la présence d'un objet dans les propositions de région, l'algorithme prédit également quatre valeurs qui sont des valeurs de décalage pour augmenter la précision de la boîte de délimitation. Par exemple, dans le cas d'une proposition de région, l'algorithme aurait prédit la présence d'une personne dans l'exemple de la figure 1.5, même si le visage de cette personne dans cette proposition de région aurait pu être réduit de moitié. Par conséquent, les valeurs de décalage aident à ajuster le cadre de délimitation de la proposition de région.

La principale limitation du modèle R-CNN classique est que l'algorithme de recherche

sélective est un algorithme fixe qui n'utilise aucun apprentissage pour proposer les régions candidates. Cela pourrait conduire à la génération de mauvaises propositions de régions candidates.

1.4.2 Faster R-CNN

La recherche sélective est un processus lent et long qui affecte la performance du réseau. Par conséquent, Shaoqing Ren et al. [Ren et al., 2015] ont mis au point un algorithme de détection d'objet qui évite l'algorithme de recherche sélective et permet au réseau de prédire les régions d'intérêts. L'image est fournie en entrée d'un réseau convolutionnel qui fournit une carte des caractéristiques convolutives (feature map). Au lieu d'utiliser un algorithme de recherche sélective sur la carte des caractéristiques pour identifier les régions, un réseau distinct est utilisé pour les prédire. Les propositions de région prédite sont ensuite remodelées à l'aide d'une couche de regroupement de RoI (Region of Interest) qui est ensuite utilisée pour classer l'image dans la région proposée et prédire les valeurs de décalage pour les boîtes englobantes.

1.4.3 Évaluation des détecteurs d'objets

Dans la détection d'objets, l'évaluation est non triviale, car il y a deux tâches distinctes à mesurer :

— Déterminer si un objet existe dans l'image (classification)

— Déterminer l'emplacement de l'objet (localisation, une tâche de régression).

De plus, dans une scène naturelle, il y aura de nombreuses classes et leur distribution n'est pas uniforme (par exemple, il pourrait y avoir beaucoup plus de voitures que de chats dans la rue). Par conséquent, une mesure simple fondée sur l'exactitude introduira des biais. Il est également important d'évaluer le risque d'erreurs de classification. Il est donc nécessaire d'associer un "score de confiance" ou score du modèle à chaque boite englobante détectée et d'évaluer le modèle à différents niveaux de confiance.

Afin de répondre à ces besoins, la précision moyenne (AP pour Average Precision) a été introduite. Pour comprendre l'AP, il est nécessaire de comprendre la précision et le rappel

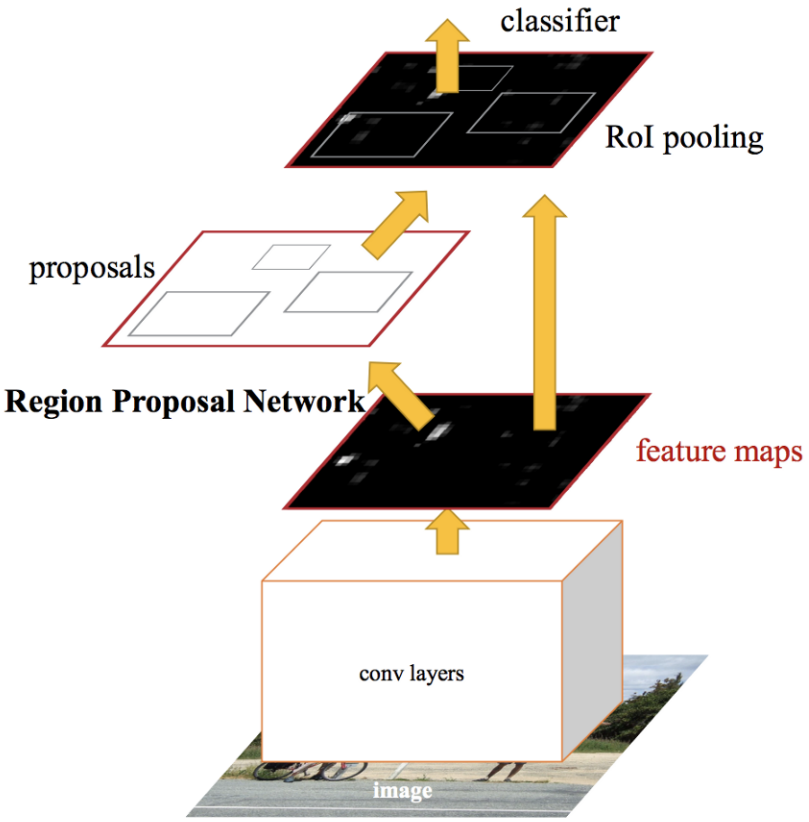


FIGURE 1.7 – Faster R-CNN

d'un classificateur. En bref, dans ce contexte, la précision mesure le "taux de faux positifs" ou le rapport entre les détections d'objets réels et le nombre total d'objets prédits par le classificateur. Ainsi, pour un score de précision proche de 1.0, il y a de fortes chances que tout ce que le classificateur prédit comme détection positive soit en fait une prédiction correcte. Le rappel mesure le "taux de faux négatifs" ou le rapport entre les détections d'objets réels et le nombre total d'objets de l'ensemble de données. Donc, avec un score de rappel proche de 1.0, presque tous les objets qui se trouvent dans votre ensemble de données seront détectés positivement par le modèle. Enfin, il est très important de noter qu'il existe une relation inverse entre la précision et le rappel et que ces mesures dépendent du seuil de score du modèle fixé (ainsi que, bien sûr, de la qualité du modèle). Par exemple, dans l'image de l'API de détection d'objets TensorFlow donnée dans la figure 1.8 [3], si nous fixons le seuil de score du modèle à 50 % pour l'objet "cerf-volant", nous obtenons 7 détections de classe positives, mais si nous fixons notre seuil de score à 90 %, il y a 4 détections positives.

3. Source : `https://github.com/tensorflow/models/tree/master/research/object_detection`

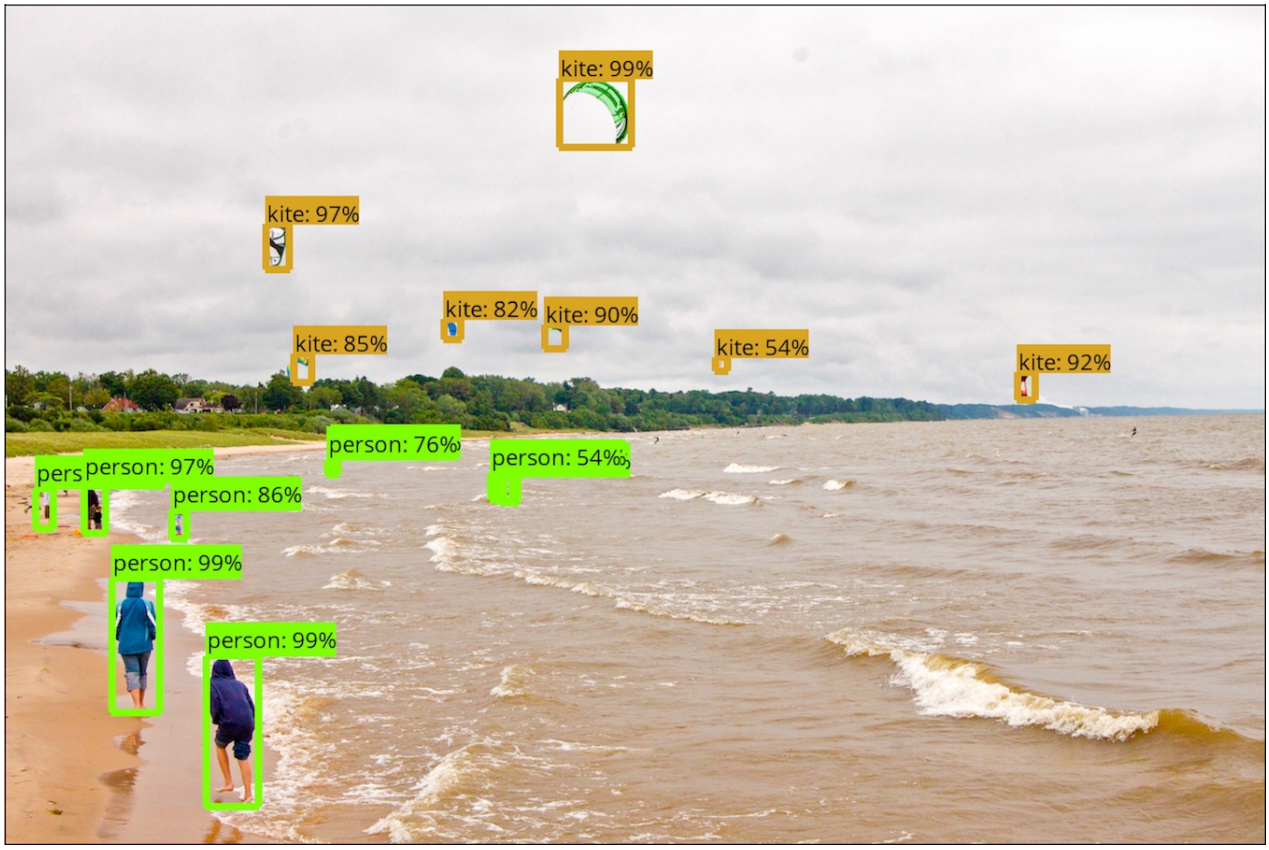


FIGURE 1.8 – Exemple de détection d'objets dans une scène naturelle

Pour calculer l'AP d'une classe spécifique (disons une "personne"), la courbe de rappel de précision est calculée à partir de la sortie de détection du modèle, en faisant varier le seuil de score du modèle qui détermine ce qui est compté comme une détection positive prévue par le modèle de la classe. Un exemple de courbe de précision-rappel peut ressembler à l'illustration donnée dans la figure 1.9.

La dernière étape du calcul de la valeur AP consiste à prendre la valeur moyenne de la précision pour toutes les valeurs du rappel. Ceci devient la valeur unique résumant la forme de la courbe de précision-rappel. Pour ce faire, la valeur AP est définie sans ambiguïté comme la précision moyenne à l'ensemble de 11 valeurs de rappel également espacées, $Rappel_i = \{0, 0.1, 0.2, ..., 1.0\}$. Ainsi,

$$AP = \frac{1}{11} \sum_{Rappel_i} Precision(Rappel_i) \tag{1.4}$$

La précision au rappel i est considérée comme la précision maximale mesurée lors d'un

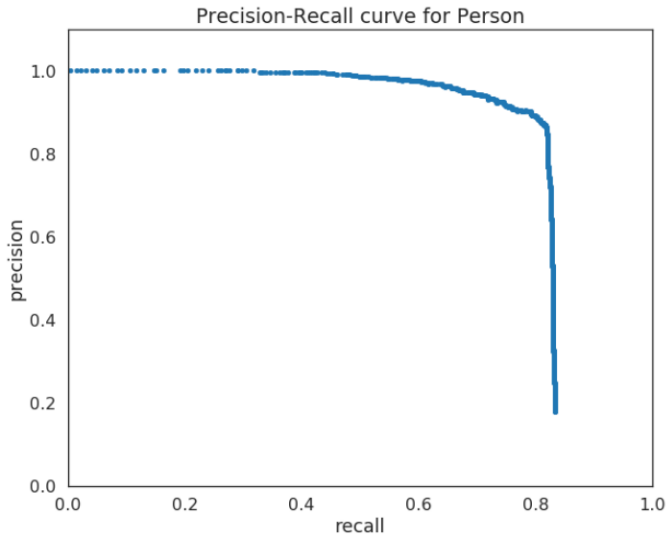


FIGURE 1.9 – Courbe Précision-Rappel pour un exemple de classificateur. Un point sur la courbe de précision-rappel est déterminé en considérant tous les objets au-dessus d'un seuil de score donné comme une prédiction positive, puis en calculant la précision résultante et le rappel pour ce seuil.

rappel dépassant $Rappel_i$. Jusqu'à présent, nous n'avons discuté que de la tâche de classification. Pour la composante de localisation (l'emplacement de l'objet a-t-il été correctement prédit ?), nous devons considérer le chevauchement entre la partie de l'image segmentée par le modèle et la partie de l'image où l'objet est réellement situé comme vraie.

1.4.3.1 Intersection over Union (IoU)

Afin d'évaluer le modèle sur la tâche de localisation d'objet, nous devons d'abord déterminer dans quelle mesure le modèle a prédit l'emplacement de l'objet. Habituellement, cela se fait en dessinant une boîte de délimitation autour de l'objet d'intérêt, mais dans certains cas, il s'agit d'un polygone à N côtés ou même d'une segmentation pixel par pixel. Dans tous ces cas, la tâche de localisation est généralement évaluée sur le seuil d'intersection par rapport à l'union des deux polygones (IoU). Pour plus de précision, dans le reste de l'article, je supposerai que le modèle prédit les encadrés de délimitation, mais presque tout ce qui est dit s'appliquera également à la segmentation par pixel ou aux polygones à N côtés.

Les détections d'objets du modèle sont déterminées comme étant vraies ou fausses en fonction du seuil de l'IoU. Ce(s) seuil(s) d'IoU pour chaque compétition varie(nt), mais dans le jeu de données COCO, par exemple, 10 seuils d'IoU différents sont considérés, de 0,5 à 0,95

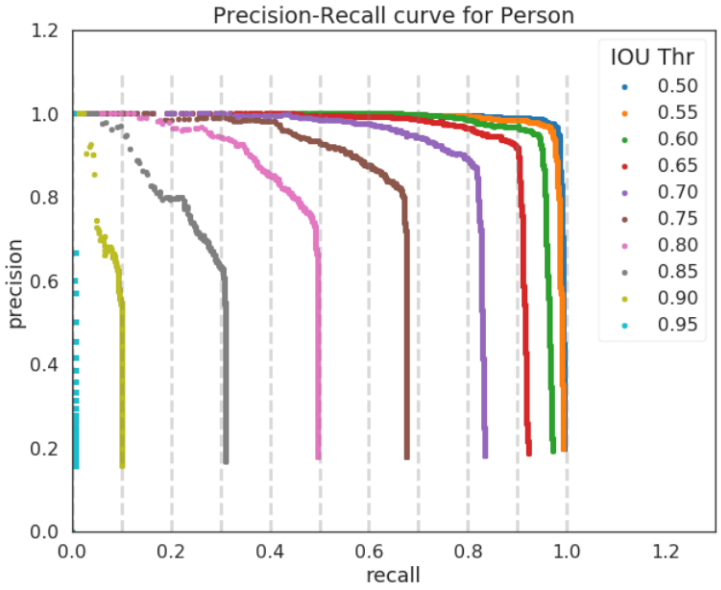


FIGURE 1.10 – Courbes de Précision-Recall calculées à différents seuils d'IoU, en fonction du défi COCO. Les lignes en pointillé correspondent à des valeurs de rappel également espacées où le PA est calculé.

par pas de 0,05. Pour un objet spécifique (par exemple, 'personne'), les courbes de rappel de précision peuvent ressembler à la courbe donnée dans la figure 1.10 lorsqu'elles sont calculées aux différents seuils de l'IoU du défi COCO.

1.4.3.2 Réunir le tout

Maintenant que nous avons défini la précision moyenne (AP) et vu comment le seuil de l'IoU l'affecte, le score moyen de précision moyenne ou score mAP est calculé en prenant l'AP moyenne sur toutes les classes et/ou sur tous les seuils de l'IoU, selon la concurrence. Par exemple :

— Le défi PASCAL VOC2007 n'a pris en compte qu'un seul seuil d'IoU : 0,5, de sorte que la moyenne du mAP a été calculée pour l'ensemble des 20 classes d'objets.

— Pour le défi COCO 2017, la moyenne du mAP a été calculée sur l'ensemble des 80 catégories d'objets et des 10 seuils de l'IoU.

1.4.4 Region-Proposal NN pour la reconnaissance des primitives

Lorsqu'on compare la détection d'objets musicaux à la détection d'objets dans des scènes naturelles ou à la reconnaissance optique de caractères, deux défis uniques méritent d'être relevés : premièrement, les partitions musicales ont souvent une densité très élevée d'objets avec plus de 1000 objets imprimés sur une même page. Deuxièmement, la position relative entre un symbole et ses lignes de portée est cruciale. Déjà une petite erreur le long de l'axe des y peut avoir un impact significatif sur la récupération de la hauteur correcte d'une note.

L'application des modèles RPN classiques, comme le R-CNN et le Faster R-CNN, sur un jeu de données différent avec un grand nombre d'objets densément compactés pourrait conduire à des performances sous-optimales. Pour cela, Pacha dans son travail [Pacha et al., 2018b] a appliqué une certaine quantité de prétraitement aux données de la base MUSCIMA++ et adapté ces détecteurs pour qu'ils fonctionnent bien pour la tâche à accomplir.

Pour l'apprentissage d'un détecteur et classificateur d'objets musicaux, ils ont utilisé le jeu de données MUSCIMA++, car il contient 140 images de haute qualité avec plus de 90000 annotations au niveau primitives effectuées par des annotateurs humains sur 105 classes différentes de symboles musicaux pour le jeu de données CVC MUSCIMA. Les images de haute résolution d'environ 3500×2000 pixels, sont binarisées et sont proposées avec et sans lignes de portée. Pour entraîner efficacement un détecteur d'objets sur de telles images, la taille de l'image doit être réduite. Ils ont proposé de recadrer les images en fonction du contexte, en coupant d'abord les images verticalement, de sorte que chaque image ne contienne qu'une seule portée, puis horizontalement pour obtenir un rapport largeur/hauteur de 2 :1 au maximum. Pour ce faire, ils ont calculé manuellement les quelques 200 tranches verticales de manière à ce que la portée et tous les objets qui lui appartiennent soient entièrement inclus, avant de couper horizontalement les images avec un chevauchement d'environ 15% par rapport aux tranches adjacentes. Fondamentalement, chaque tranche horizontale s'étend à partir du bas de la portée supérieure à la partie haute de la portée inférieure. Ce recadrage peut aussi être effectué en détectant automatiquement les lignes de portée et en appliquant ensuite les mêmes règles de découpage en tranches, ce qui permet d'obtenir des images qui, en partie se chevauchent horizontalement et verticalement. Une limitation de cette approche

est que les objets qui excèdent de manière significative la taille de ces régions n'apparaissent pas dans la vérité terrain, comme seules les annotations qui ont un point d'intersection de 0,8 ou plus entre l'objet et l'image découpée sont incluses. Pour évaluer leur approche ils ont mené plusieurs expériences pour étudier les effets de différents détecteurs, de différents extracteurs de caractéristiques, de l'enlèvement de lignes de portée, d'un nombre réduit de classes et du transfert-learning. Pour cela ils ont considéré :

- les trois méta-architectures Faster R-CNN, R-FCN (Region-based Fully Convolutional Networks [Dai et al., 2016]) et SSD (Single Shot multibox Detector [Liu et al., 2016]) comme détecteurs d'objets.
- Les quatre réseaux ResNet50, Inception-ResNet-v2, MobileNet-v1 et Inception-v2 comme extracteurs de caractéristiques, ainsi que des réseaux sur mesure directement entraînés sur les données du jeu MUSCIMA++, incluant un apprentissage complet sur l'ensemble des 105 classes, ainsi sur un nombre réduit de 71 classes (sur le jeu réduit dénommé MUSCIMA++71) et enfin de 49 classes (jeu réduit appelé MUSCIMA++ 49) ayant supprimé tous les objets composés (comme la signature de la clé et la signature temporelle) et les objets linéaires (comme les ligatures, hampes, barres de mesure et d'autres).
- La suppression pour certaines parties d'expériences de 34 classes dont les représentants apparaissent moins de 50 fois dans la vérité terrain et qui n'ont qu'une importance mineure, tels que les annotations chiffrées et textuelles (au voisinage des lignes de portée comme indication du jeu musical), à l'exception de quelques chiffres comme le 5, 6, 7 et 8 qui donnent des indications mélodiques importantes dont on ne saurait se passer.
- L'initialisation de l'entraînement des réseaux s'est faite avec des poids aléatoires (pour les réseaux entraînés uniquement sur MUSCIMA++) et avec des poids pré-entraînés au sein du jeu de données COCO pour les réseaux génériques ResNet, Inception et MobileNet.

Ils ont trouvé que le détecteur le plus performant est le Faster R-CNN utilisant l'extracteur de caractéristiques Inception-Resnet V2 pré-entraîné sur le jeu de données COCO. Ce

modèle produit une mAP de 80%. Ils ont trouvé que la suppression des lignes de portée n'est plus nécessaire avec les architectures neuronales récentes comme l'apprentissage profond. D'autres détecteurs comme le R-FCN ou le SSD donnent également de bons résultats, avec un mAP de 75% et 62% respectivement. De plus, le fait de modifier l'ensemble des classes en supprimant les classes sous-représentées de moins de 50 échantillons, a augmenté le mAP de 6% (MUSCIMA++ 71).

Dans le but de montrer la différence entre une approche basée sur un modèle de classification classique comme le SVM et un modèle basé sur les méthodes d'apprentissage profond de type RPN, on a implémenté un système de détection et de reconnaissance des primitives musicales. En effet, l'utilisation d'un SVM binaire par classe combiné avec le LBP, pour l'extraction des caractéristiques, pour la tâche de classification des primitives musicales a montré des performances générales très satisfaisantes comprises entre 90% et 100% en fonction du type de la primitive. Cependant, l'utilisation de plusieurs fenêtres glissantes (i.e. une fenêtre par classe de primitive) a montré des performances très limitées de l'ordre de 0% à environ 50% en fonction du type de la primitive. La figure 1.11 et la figure 1.12 illustrent la différence des performances entre l'approche basée sur le SVM et l'approche basée sur le RPN.

De ce fait, nous proposons d'adapter l'architecture d'apprentissage en profondeur *Faster Region Proposal CNN (Fast R-CNN)* utilisée dans [Pacha et al., 2018b] pour détecter et classifier les primitives musicales et les transformer en séquences textuelles pour extraire correctement les motifs musicaux.

Dans notre étude, au niveau de la base MUSCIMA++, nous considérons uniquement les partitions à une voix avec un seul instrument (correspondant à un total de 9 pages et 11 primitives) afin de permettre la représentation de la partition sous forme d'une séquence

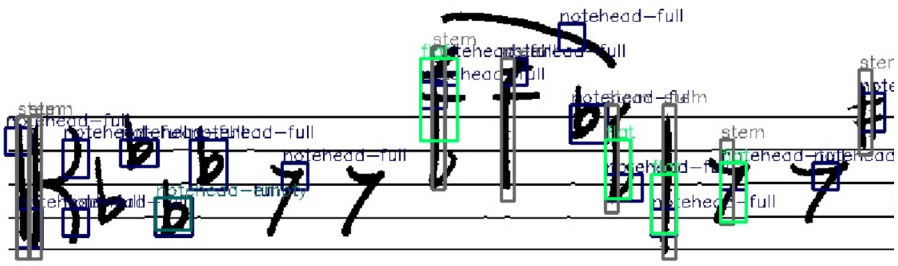


FIGURE 1.11 – Échantillon du résultat SVM

musicale unique (unidimensionnelle). Cette sélection évite de confondre les accords et la disposition des voix lors de la construction des séquences musicales. De plus, l'utilisation d'un algorithme, basé sur des projections verticales, pour la détection des lignes de portée nous a poussé à ne pas considérer certaines images de partition (celles présentant une inclinaison significative), même si cela peut être évité par une opération de redressement.

Les images de lignes de portée provenant du jeu de données MUSCIMA sont utilisées pour détecter et coder les lignes de portée pour chaque image de partition. La sortie Faster R-CNN, basée sur Inception ResNet v2 (pour l'extraction de caractéristiques) est utilisée comme entrée pour l'outil d'annotation pour coder les primitives.

Les tableaux 1.1 résume les performances en mAP (mean Average Precision) de la détection et de la reconnaissance de primitives obtenues par le Faster R-CNN (RPN) avec la même configuration que celle utilisée dans [Pacha et al., 2018b]. L'architecture a été appliquée à une sélection de primitives musicales de l'ensemble de données MUSCIMA, soigneusement choisies pour traiter de manière complète les informations mélodiques des partitions. Ces primitives sont choisies parmi les cent primitives : tête de note pleine, tête de note creuse, bémol, bécarre, dièse, barres de mesure, lignes supplémentaires (au-dessus et au-dessous de la portée) et clefs. Dans ce scénario, nous avons ignoré les primitives d'importance mineure, telles que les chiffres, les lettres, et les autres objets moins fréquents. Nous notons que la plupart des primitives sont bien détectées et reconnues avec une précision moyenne supérieure à 87 %, à l'exception des lignes supplémentaires, qui sont très confondues par rapport aux autres types de primitives (seulement 61 % de reconnaissance correcte) comme indiqué dans le tableau 1.1. Dans la figure 1.13, nous montrons deux exemples de sortie du Faster R-CNN, le premier illustre une situation dans laquelle le réseau ne parvient pas à détecter

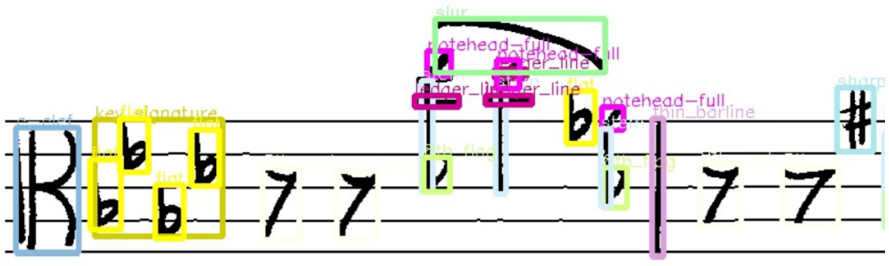


FIGURE 1.12 – Échantillon du résultat RPN

certaines primitives et le second, un exemple dans lequel il a totalement réussi. Les difficultés rencontrées par Faster R-CNN pour reconnaître centaines primitives sont dues à de nombreuses raisons : le plus souvent, cela est dû aux styles d'écriture et à la forme graphique de la primitive.

La dernière étude expérimentale présentée dans la section 1.5.2 montrera l'impact des erreurs de reconnaissance sur la qualité de la transcription. Les résultats détaillés par primitive sont donnés dans la table1.1 tels que :

— AID : Author ID

— SID : Sequence ID

— AP : Average Precision par page

— NF : Tête de note pleine

— NE : Tête de note creuse

— S : Hampe

— B : Ligature

— F : Bémol

— N : Bécarre

— f-clef : Clé de Fa

— g-clef : Clé de Sol

— c-clef : Clé d'Ut

— LL : Lignes supplémentaires

— BL : Barres de mesures

— LD : Levenstein Distance (qui sera analysé dans la section 1.5)

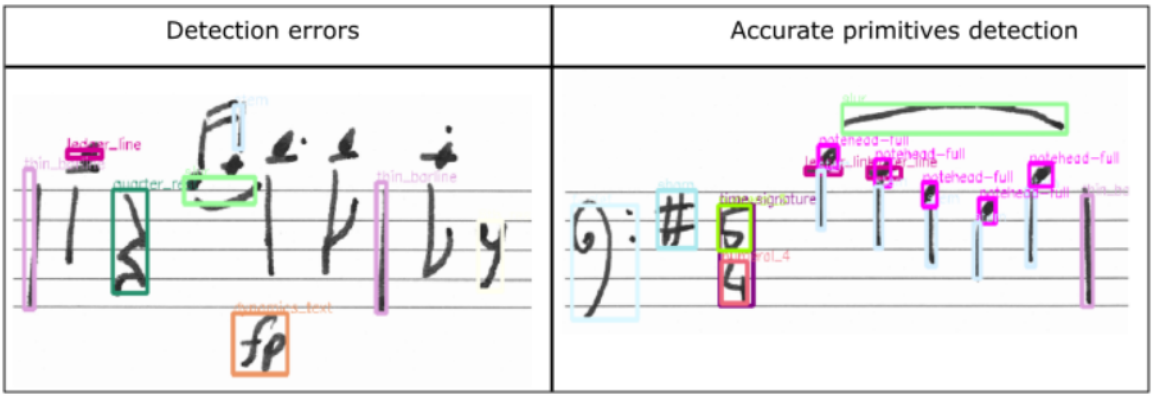


FIGURE 1.13 – Exemple de la sortie du Faster R-CNN

TABLE 1.1 – Statistiques en mAP (mean Average Precision) du Faster R-CNN sur un ensemble de teste en considérant 11 classes de primitives musicales

ID page	AID	SID	AP	NF	NE	S	B	F	#	N	f-clef	g-clef	c-clef	LL	BL	LD
1	10	1	0,906	0,993	1	0,994	0,967	1	1	1	1	-	1	0,678	0,963	0,16
2	17	1	0,913	0,993	1	0,994	1	1	1	0,889	1	-	1	0,712	1	0,21
3	24	1	0,687	0,748	0,5	0,745	0,783	0,5	0,909	0,8	1	-	1	0,424	0,885	0,55
4	38	1	0,937	1	1	0,994	1	1	1	1	1	-	1	0,883	1	0,13
5	45	1	0,919	0,993	1	0,994	0,983	1	1	0,9	1	-	1	0,78	1	0,2
Mean			0,8724	0,9454	0,9	0,9442	0,9466	0,9	0,9818	0,9178	1	-	1	0,6954	0,9696	0,25
6	6	2	0,451	0,426	0,263	0,69	0,5	0,6	0,4	0,25	-	1	-	0,208	0,725	0,71
7	13	2	0,55	0,781	0,789	0,624	0,409	0,8	1	0,5	-	1	-	0,203	0,714	0,51
8	20	2	0,858	0,991	1	0,992	0,909	1	1	1	-	1	-	0,766	1	0,2
9	34	2	0,637	0,913	0,895	0,667	0,659	0,4	0,867	0,75	-	1	-	0,338	0,796	0,44
10	48	2	0,868	0,991	1	0,984	0,884	1	1	1	-	1	-	0,868	1	0,19
Mean			0,6728	0,8204	0,7894	0,7914	0,6722	0,76	0,8534	0,7	-	1	-	0,4766	0,847	0,41
11	12	4	0,542	0,469	0,5	0,73	0,66	1	0,9	1	-	1	-	0,33	0,795	0,61
12	26	4	0,826	0,967	1	0,889	0,904	1	1	1	-	1	-	0,699	1	0,41
13	33	4	0,853	0,966	1	0,986	0,96	1	1	1	-	1	-	0,689	0,974	0,34
14	47	4	0,863	1	1	0,993	0,885	1	1	1	-	1	-	0,77	0,905	0,18
15	50	4	0,867	0,993	1	0,98	0,904	1	1	1	-	1	-	0,762	0,975	0,25
Mean			0,7902	0,879	0,9	0,915	0,862	1	0,98	1	-	1	-	0,65	0,923	0,358
16	2	6	0,839	1	1	0,992	0,88	0,976	1	0,975	-	-	1	0,634	1	0,26
17	9	6	0,84	1	1	0,953	0,84	0,977	1	1	-	-	1	0,725	1	0,24
18	30	6	0,592	0,811	0,939	0,535	0,52	0,721	0,818	0,707	-	-	1	0,275	0,839	0,48
19	44	6	0,815	0,962	1	0,977	0,846	0,977	1	0,978	-	-	1	0,5	0,954	0,38
Mean			0,771	0,943	0,984	0,864	0,771	0,912	0,954	0,915	-	-	1	0,533	0,948	0,34
20	3	7	0,93	0,981	1	0,981	0,886	-	0,917	1	1	-	-	0,909	0,941	0,08
21	10	7	0,931	0,994	1	0,968	0,93	-	1	1	1	-	-	0,682	1	0,11
22	17	7	0,924	1	1	0,987	0,831	-	1	1	1	-	-	0,727	0,941	0,15
23	24	7	0,903	0,994	1	0,942	0,797	-	1	1	1	-	-	0,818	1	0,18
24	38	7	0,93	0,987	1	0,974	0,873	-	1	1	1	-	-	0,864	1	0,1
25	46	7	0,937	0,987	1	1	0,843	-	0,846	1	1	-	-	1	1	0,06
Mean			0,925	0,991	1	0,975	0,86	-	0,9605	1	1	-	-	0,833	0,980	0,113
26	4	9	0,921	1	1	1	1	1	1	1	1	-	-	0,76	0,923	0,07
27	18	9	0,695	0,705	0,6	0,776	0,788	1	1	0,5	1	-	-	0,423	0,885	0,36
28	25	9	0,49	0,287	0,8	0,679	0,818	1	0,5	0,75	1	-	-	0,077	0,714	0,78
29	28	9	0,793	0,868	1	0,91	0,879	0	1	0,5	1	-	-	0,192	1	0,3
30	49	9	0,923	1	0,8	0,985	0,97	1	1	1	1	-	-	0,808	1	0,07
Mean			0,7644	0,772	0,84	0,87	0,891	0,8	0,9	0,75	1	-	-	0,452	0,9044	0,316
31	5	11	0,923	0,977	-	0,981	0,986	1	1	0,833	1	-	-	0,906	0,974	0,07
32	12	11	0,643	0,681	-	0,671	0,746	0,75	1	0,833	1	-	-	0,312	0,737	0,46
33	26	11	0,672	0,657	-	0,746	0,757	1	0,929	1	1	-	-	0,375	0,789	0,48
34	35	11	0,922	0,995	-	0,967	1	1	1	1	1	-	-	0,788	0,974	0,06
35	42	11	0,916	0,981	-	0,981	0,959	1	1	1	1	-	-	0,781	0,973	0,09
36	49	11	0,912	0,972	-	0,948	0,986	1	1	1	1	-	-	0,875	1	0,15
Mean			0,831	0,877	-	0,882	0,905	0,958	0,988	0,944	1	-	-	0,672	0,907	0,218
37	2	13	0,877	0,991	1	0,965	0,812	1	-	1	-	-	1	0,846	0,973	0,23
38	16	13	0,887	1	1	0,991	0,812	1	-	1	-	-	1	0,85	0,973	0,15
39	37	13	0,873	1	1	0,974	0,875	1	-	0,95	-	-	1	0,75	0,972	0,21
40	44	13	0,851	1	1	0,983	0,688	1	-	1	-	-	1	0,525	0,972	0,31
Mean			0,872	0,99775	1	0,97825	0,79675	1	-	0,9875	-	-	1	0,74275	0,9725	0,225
41	8	15	0,924	0,993	-	0,993	0,945	0,933	1	1	-	-	1	0,843	1	0,16
42	15	15	0,555	0,657	-	0,657	0,615	0,692	0	0,667	-	-	1	0,129	0,737	0,6
43	36	15	0,654	0,821	-	0,761	0,593	0,923	0,5	0,889	-	-	1	0,29	0,632	0,51
44	43	15	0,647	0,644	-	0,896	0,567	0,733	1	0,9	-	-	1	0,343	0,737	0,57
45	50	15	0,659	0,731	-	0,828	0,626	0,667	0,5	0,444	-	-	1	0,371	0,842	0,53
Mean			0,6878	0,7692	-	0,827	0,6692	0,7896	0,6	0,78	-	-	1	0,3952	0,7896	0,474

L'étape de reconnaissance de symboles de musique permet de localiser et de positionner les symboles de musique par rapport aux lignes de portées. L'approche choisie fonctionne au niveau des primitives et ceci nous aidera à développer un processus d'encodage de ces primitives afin de produire une séquence (ou pseudo-transcription). Cette séquence sera ensuite utilisée pour l'extraction de motifs. L'étape de la génération de séquences est détaillée dans la section suivante.

1.5 Génération de séquences à partir de partitions musicales numérisées

Dans la littérature, il est recommandé d'utiliser la méthode DMOS (Description et MOdification de la Segmentation) qui est une méthode générique de reconnaissance de documents structurés [Couasnon, 2001] [Coüasnon, 2006]. Cette méthode est basée sur le formalisme grammatical EPF (Enhanced Position Formalism) qui permet d'effectuer une description graphique, syntaxique et sémantique d'un type de documents. Ainsi, dans le cas des documents d'images de partitions numérisées, DMOS peut être utilisé pour décrire la structure graphique d'une partition musicale telle qu'une partition est composée de lignes de portée, notes, altérations, etc. Ainsi, la description syntaxique consiste en la vérification de l'exactitude de l'information musicale, par exemple une tête de note pleine doit impérativement être associée à une hampe. Finalement, la description sémantique permet de vérifier que le résultat obtenu est musicalement correct par exemple en s'assurant que la temporalité (ou le nombre de battements par mesure) correspond bien à la signature temporelle du solfège (Meter en Anglais). Cette méthode permet au système de détecter la plupart des erreurs ce qui mène, généralement, à proposer une méthode de correction [Couasnon, 2001].

Même si la méthode DMOS est recommandée par une bonne partie des travaux portant sur l'analyse optique des partitions musicales [Rebelo et al., 2010] [Pacha et al., 2018a], l'utilisation de cette méthode est difficile à mettre en place dans le cas des partitions manuscrites du moment où le scripteur ne respecte pas forcément la structure du solfège ce qui mène à adapter la grammaire en fonction de la qualité du manuscrit (e.g. s'il y a des inclinaisons, si le

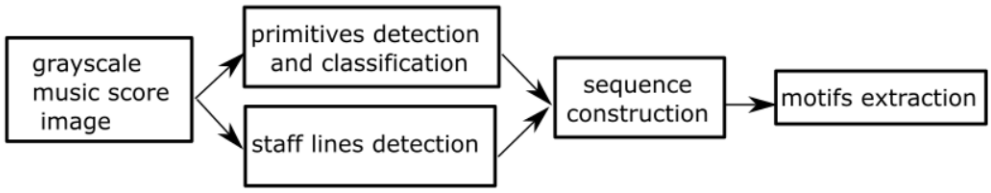


FIGURE 1.14 – Processus d'extraction de motifs musicaux à partir d'images de partitions

scripteur ne colle pas les têtes de notes des hampes, etc.). Le but de notre travail consiste en la génération d'une séquence à partir de la sortie du système de détection et reconnaissance suivant des règles simples de construction. Ceci nous conduit à développer notre méthode d'encodage.

Dans cette partie, on décrit le processus d'encodage mis en place pour la génération d'une séquence de primitives musicales à partir d'un modèle RPN (*Région-proposition Convolutional Neural Network*).

Ce processus d'encodage est utilisé pour créer une séquence pour être traitée par un algorithme de fouille de séquence, qui sera présenté dans le chapitre suivant, afin d'en extraire des motifs de tailles différentes, comme illustré à la figure 1.14.

1.5.1 Encodage de primitives et génération de séquences

Dans l'ensemble de données MUSCIMA, une image de partition relative à un auteur est présentée sous trois formes : binaire avec lignes de portée, en niveau de gris avec lignes de portée et binaires sans lignes de portée.

Bien qu'il existe une grande diversité d'approches de détection des lignes de portée, même pour les images déformées [Visani et al., 2013], nous avons opté pour l'algorithme de projection horizontale suffisamment efficace pour notre sélection d'images MUSCIMA. Il convient de noter que nous pouvons facilement utiliser un algorithme pour ajuster l'inclinaison des lignes. Cette étape permet de détecter très précisément les cinq lignes formant la portée. En conséquence, les primitives sont codées avec leurs positions (sur les lignes ou entre les lignes), telles que la première ligne en haut est codée à 0, la deuxième ligne de portée est codée à 2 et l'espace entre les deux est codé à 1, etc. La figure 1.15 illustre le mécanisme d'encodage des lignes (les nombres d'espaces ne sont pas tracés pour des raisons de lisibilité).

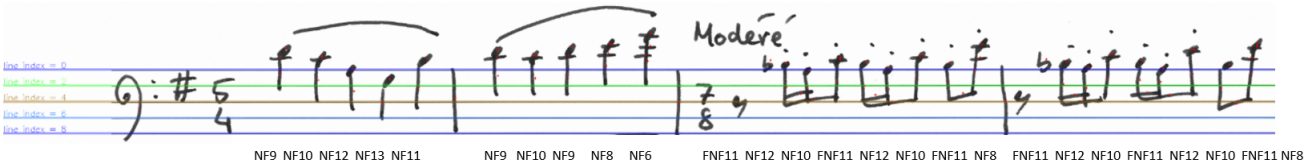


FIGURE 1.15 – Énumération des lignes de portée pour l'encodage des positions des primitives musicales

Chaque image de primitive est définie par sa classe, le point d'ancrage dans l'image originale et ses dimensions, qu'il s'agisse du jeu de données de primitives MUSCIMA ++ ou des sorties RPN. Comme expliqué précédemment, nous ne considérons qu'un sous-ensemble significatif de primitives que nous jugeons efficaces pour couvrir la plupart des informations mélodiques présentes dans une partition. Cet encodage au niveau primitives permet de créer des séquences à partir de partitions de telle sorte que les notes soient d'abord ordonnées par la position verticale de la portée associée, puis de gauche à droite en suivant le sens de lecture au niveau de la portée, voir la figure 1.15.

La première étape consiste à associer des hampes à des portées en considérant les espaces partagées maximaux entre la hampe et la portée, sinon en affectant la hampe à la portée la plus proche, comme illustré à la figure 1.16. Ensuite, les têtes de notes et les altérations sont associées aux portées suivant la hampe la plus proche (voir les liens représentés par des flèches dans la figure 1.16). Ensuite, nous stockons le numéro de ligne (ou l'espace) la plus proche du centre de la primitive. La séquence de primitives est ordonnée le long de l'axe des abscisses pour assurer la temporalité, à l'exception des notes en accord, jouées en même temps, qui sont ordonnées du haut au bas selon l'axe des ordonnées.

Dans la figure 1.16, les primitives sont codées suivant l'expression régulière :

? *(Altération) (NoteType) (Position)*

Telle que ? Pour 0 ou 1 occurrence, *Altération* peut être (F : bémol, N : bécarre, ♯ : dièse), *NoteType* peut être (NF : tête de note pleine, NE : tête de note creuse) et *Position* fait référence à l'index de la ligne de portée associée.

La position des notes, au-dessus ou au-dessous de la portée, dépend du nombre apparent de lignes supplémentaires en superposition avec la hampe de la note. Par exemple, dans la figure 1.16, la tête de note creuse sur la deuxième portée a 4 lignes supplémentaires et,

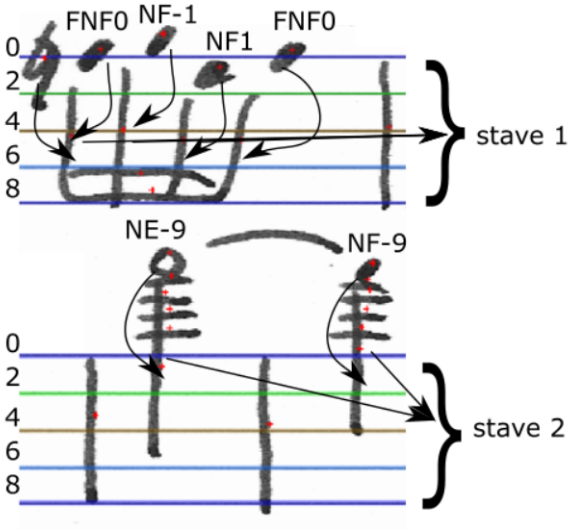


FIGURE 1.16 – Encodage des têtes de notes

suivant le même codage que celui utilisé pour les lignes de portée, cette position correspond à -9.

Afin de compléter les informations mélodiques, nous considérons la projection des altérations par rapport aux lignes de portée. Tout d'abord, les clefs doivent être identifiées et le codage des lignes de portée doit être ajusté : par exemple, pour une Clef de Fa située sur l'index de ligne 2, l'index de toutes les lignes doit être décalé de +12, en convertissant une valeur d'indice de ligne de 0 à 12. Pour plus d'explications sur la signification des clefs et leur utilisation, nous invitons le lecteur à se référer à l'annexe A ou à consulter simplement la page Web Wikipedia [4]. Ceci justifie les valeurs de la séquence dans la figure 1.15.

Ensuite, en fonction du type de la clef, nous comptons le nombre d'altérations à côté de la clé pour déterminer l'armature du solfège. Cette armature permet d'associer des altérations mineures / majeures en fonction de leur position vis-à-vis les notes de musique. À l'intérieur des mesures, chaque altération est associée aux notes qui la suivent directement. Le passage à la mesure suivante, par le biais de la barre de séparation, annule l'effet de l'altération.

De ce fait, les séquences sont générées à partir de la position des primitives dans l'image de la partition et de la présence concomitante d'altérations lors de leur existence. Il est alors simple d'associer chaque codage au niveau des primitives à un codage MIDI ainsi qu'à un codage XML (transcription) en utilisant des règles de correspondance simples entre encodages.

4. `https://en.wikipedia.org/wiki/Clef`

Par exemple, #NF0 (une noire sur la première ligne de portée avec une altération sur #) correspond à la note 78 de l'encodage MIDI. La séquence de primitives peut facilement être exprimée par le même vocabulaire qu'une séquence MIDI comme montré dans l'annexe C.

Il devient alors évident de calculer la précision de la transcription (sur le jeu de données de l'étude) en utilisant les primitives annotées MUSCIMA ++ comme vérité terrain et les primitives de sortie RPN. Ce deuxième ensemble de données révélera l'impact des erreurs de reconnaissance sur la cohérence de la transcription. Pour l'évaluation de la transcription, chaque partition est représentée par deux séquences MIDI : la séquence MUSCIMA ++ basée sur la vérité terrain qui est noté XML-GT et la séquence basée sur la sortie RPN. L'évaluation des transcriptions des séquences générées par le RPN repose sur la comparaison avec les séquences de vérité terrain MUSCIMA ++.

1.5.2 Évaluation

Le but de cette section est de montrer et de discuter l'impact des erreurs de reconnaissance des primitives sur la précision de la transcription. L'évaluation de la transcription consiste à estimer la distance de Levenshtein entre les séquences issues de la transcription MUSCIMA++ (déduites du XML de vérité terrain et notées XML-GT) et les séquences de sortie RPN. La distance consiste en un calcul pondéré des opérations de suppression, d'insertion ou de substitution d'un ou plusieurs éléments dans une séquence pour en construire une séquence donnée [Navarro, 2001]. Dans cette définition de la distance d'édition, les erreurs d'insertion, de suppression et de substitution ont le même poids. La distance finale est ensuite normalisée par rapport à la taille de la séquence de référence (dans XML-GT).

Pour montrer l'impact de reconnaissance de certains types de primitives sur la séquence générée, nous proposons aux figures 1.17 et 1.18 de visualiser le lien existant entre la qualité de reconnaissance par le Faster R-CNN (RPN) et la distance de Levenshtein établie pour chaque page du test entre la séquence XML-GT et celle issue de l'encodage. Les valeurs de ces taux de performance sont données dans les colonnes respectives NF, F, # dans la table 1.1, et la distance de Levenshtein est donnée dans la colonne LD. Sur l'axe des abscisses de la figure 1.17 on a représenté les pages des partitions musicales, et sur l'axe des ordonnées

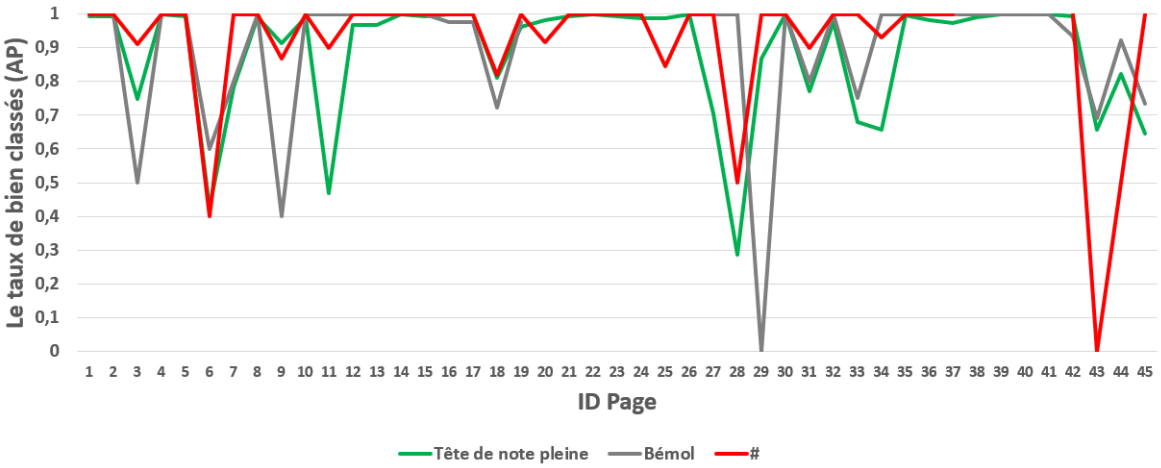


FIGURE 1.17 – Résultats de reconnaissance en précision des trois classes de primitives : tête de note pleine, bémol et # par le réseau RPN (Faster R-CNN)

FIGURE 1.18 – Distances Levenshtein par page entre la séquence XML-GT et la séquence issue de l'encodage des pages référencées par leur ID (abscisse)

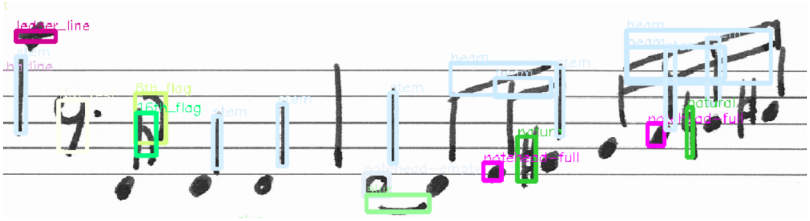


FIGURE 1.19 – Erreurs de reconnaissance par le RPN causées par le style d'écriture

figure le taux des primitives bien classées étudiées. L'axe des ordonnées de la figure 1.18 correspond aux valeurs de la distance de Levenshtein. Il faut noter que si une courbe d'une primitive dans la figure 1.17 descend à la valeur 0, ceci veut dire qu'aucune occurrence de la primitive n'a été bien trouvée dans la page analysée. A partir de ces deux graphiques, on peut remarquer la dépendance inverse de la courbe bleue (celle de la distance de Levenshtein) par rapport aux autres courbes (celles des taux de reconnaissance des primitives). Ainsi, les valeurs minimales de distance sont celles où les taux sont maximaux, et les pics supérieurs de distance (e.g. dans les cas des pages 3, 6, 28, etc.) correspondent aux situations où le RPN reconnaît mal les primitives.

Ces erreurs de détection sont causées principalement par le style d'écriture de l'auteur, comme illustré dans la figure 1.19 où les erreurs sont potentiellement dues au fait que l'auteur met trop d'espaces entre les hampes et les têtes de notes et la densité de l'information musicale dans certaines parties de l'image comme illustré dans la figure 1.20 où le réseau a trouvé des difficultés à détecter toutes les lignes supplémentaires au-dessus de la portée car elles sont très proches les unes des autres.

Une autre cause d'erreurs rencontrées concerne l'interprétation des têtes de notes vis-à-vis de leur ligne de portée (voir figure 2.7 reprise au chapitre 2 du manuscrit) lorsque la position de la note prête à confusion. On rencontre cette situation dans les pages 12, 13, 38, et 41 (cf. figure 1.17) où on peut facilement observer ce phénomène sur les variations de distance de Levenshtein (cf. figure 1.18).

Ce chapitre présente un enchaînement de traitements qui sont proposés pour préparer les données pour l'algorithme de mining qui est développé au chapitre suivant. Ces traitements comportent différentes étapes de bas niveau permettant d'une part d'extraire la couche d'information sémantique des partitions (suppression massive des lignes de portées),

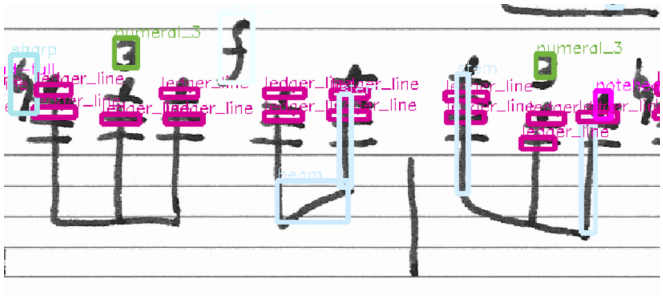


FIGURE 1.20 – Erreurs de reconnaissance par le RPN sur une partie dense de l'image

de reconnaître sur cette carte les primitives musicales principales pour ensuite produire un encodage respectant la temporalité des partitions mono-voix. La plupart des tests effectués dans ce chapitre ont été réalisés sur le jeu de données MUSCIMA pour l'analyse globale de la partition (binarisation, extraction des lignes de portées et encodage) et MUSCIMA++ pour les primitives musicales. Le scenario retenu permet dans un premier temps d'affranchir le système des difficultés d'une transcription complète (on limite la reconnaissance à l'interprétation des primitives mélodiques) ou de l'analyse de partitions plus complexes (i.e. multi-voix) qui garantit un suivi temporel immédiat (lecture de gauche à droite).

Sur ces séquences une fois produite par notre encodeur, nous proposons une méthodologie complète d'extraction de motifs musicaux fréquents ne nécessitant aucune connaissance a priori sur la forme du motif (longueur, composition, emplacement) et paramétrable de façon à n'extraire que les motifs fréquents (de fréquence minimale) et permettant de contrôler des variations internes lors de la recherche (présence de distracteurs dans la chaine, possibilité d'oubli d'un ou plusieurs caractères). Dans le chapitre suivant, nous présentons ainsi toutes nos contributions en termes de sequence mining pour la détection de motifs et nous montrerons que les motifs sont des outils performants d'analyse musicologique malgré de possibles erreurs de reconnaissance initiales.

Chapitre 2

Extraction de motifs par une approche de fouille de séquences - Application à la détection de motifs musicaux

L'extraction de connaissances à partir de données, également connue sous le nom de Knowledge Discovery in Databases (KDD), fait référence à l'extraction non triviale d'informations implicites, auparavant inconnues et potentiellement utiles à partir de données contenues dans des bases de données, comme décrit dans Fayyad et al [Fayyad et al., 1996]. La figure 2.1 illustre le processus KDD. Il est divisé en trois grandes étapes. La première étape est appelée prétraitement, qui prépare les données cibles à appliquer les techniques de fouille de données. Cette étape se divise en trois sous-étapes qui sont la sélection, le prétraitement et la transformation des données. L'étape principale du processus KDD est l'exploration de données où différents algorithmes peuvent être appliqués pour découvrir des connaissances cachées. Vient ensuite un autre processus appelé post-traitement, qui évalue le résultat de la fouille de données en fonction des besoins des utilisateurs et des connaissances du domaine. Plus précisément, le KDD peut être divisé en cinq étapes : Fayyad et al [Fayyad et al., 1996], et Dunham [Dunham, 2006] :

— Sélection : Les données nécessaires au processus de fouille de données peuvent provenir de nombreuses sources de données différentes et hétérogènes. Au cours de cette étape,

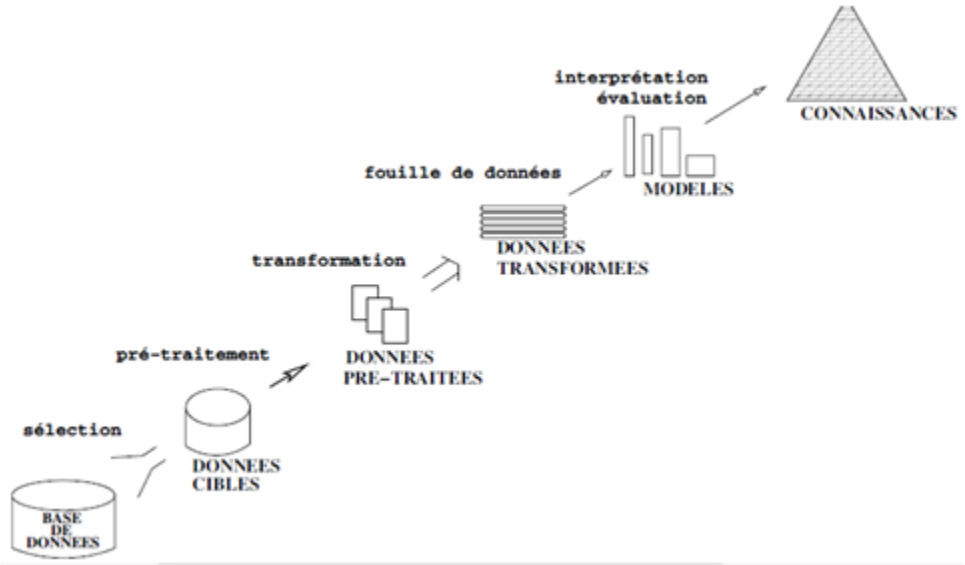


FIGURE 2.1 – Processus de fouille de données [Fayyad et al., 1996]

les données sont recueillies à partir de diverses sources telles que les bases de données, les fichiers, les sources non électroniques (rapports, livres, etc.).

— Prétraitement : Cette étape consiste à nettoyer les données collectées en éliminant les incohérences et les duplications de bruit. Ensuite, les données sont combinées dans une source commune comme la base de données, les entrepôts ou autres dépôts.

— Transformation : Les données prétraitées sont transformées dans un format traitable en fonction des formats de données en entrée pour les algorithmes de fouille de données.

— Exploration de données : L'étape de fouille de données traite du développement et de l'application de plusieurs algorithmes pour extraire l'information et les modèles dérivés du processus KDD.

— Interprétation/évaluation : Les informations découvertes avec la fouille de données doivent être validées. Dans cette étape, l'information extraite est représentée pour les analystes par le biais de techniques de visualisation pour les aider à comprendre et à interpréter les résultats.

Selon la nature des données considérées, différentes familles de méthodes peuvent être mobilisées pour la fouille et dans cette thèse, nous nous sommes orientés vers les méthodes de recherche d'ensembles fréquents (frequent iemsets mining) afin d'identifier les motifs caractéristiques d'une partition. De plus nous avons conçu un processus de pré-traitement visant

à encoder une partition sous forme de séquences pour pouvoir identifier, à l'aide du même algorithme de fouille, trois variantes d'un motif musical : miroir, inversé, translaté.

La suite de ce chapitre est organisée comme suit : on commence par présenter les travaux existants sur la fouille de séquences. Ensuite, on détaille l'algorithme CSMA proposé dans le contexte de la thèse pour la recherche de motifs dans une ou plusieurs séquences. La dernière partie du chapitre est consacrée aux différents traitements d'encodage de séquences musicales traitées par CSMA et l'extraction des variantes des motifs musicaux suivie par l'étude de l'intérêt de l'utilisation des gaps dans le contexte de fouille de séquences issues des sorties du modèle RPN.

2.1 Fouille de séquences - état de l'art

La fouille de données est un processus permettant d'extraire des connaissances cachées dans un jeu de données. Agrawal et Srikant ont introduit les approches de la fouille de séquences sur une base de données de transactions de clients, dans laquelle chaque transaction est composée de l'ID client, de la date de transaction et des produits achetés dans la transaction [Agrawal and Srikant, 1995]. Dans le domaine de la fouille de séquence, nous parlons de *motifs séquentiels fréquents*. Un motif structuré est un type particulier de motif séquentiel fréquent, où les motifs simples sont connus à priori. L'extraction de motifs structurés a des applications importantes, notamment dans l'analyse de séquences d'ADN. En règle générale, La fouille de motifs structurés est appliquée à un jeu de données de chaînes. On distingue deux types de motifs ; *motifs contigus* et *motifs non contigus*. Les motifs contigus sont des motifs où les éléments sont positionnés successivement. À l'inverse, pour les motifs non contigus, les sauts dans les positions des éléments sont autorisés. Floratou et. al ont introduit les motifs approximatifs contigus permettant des permutations des items (définis par une distance) dans les motifs [Floratou et al., 2011].

Dans le domaine de la musique, Hsu et. al ont introduit la première approche consistant à trouver des motifs fréquents dans une partition, qui est représentée par une seule séquence, en ne considérant que l'information mélodique [Hsu et al., 1998].

TABLE 2.1 – Base de données séquentielle [Egho, 2014]

ID	Transaction
1	$\{a\}, \{e\}$
2	$\{a\}, \{b, c, d\}$
3	$\{a, c\}$
4	$\{a\}$, $\{b, c, d\}$, $\{e\}$
5	$\{e\}$

Dans notre travail, on se focalise sur les données musicales, de sorte que nous essayons d'extraire les motifs séquentiels fréquents à partir de partitions, que ce soit transcrites ou issues de sortie RPN. Les données d'entrée sont une séquence de notes où chaque note contient deux types d'informations. Le premier est lié à sa valeur mélodique (position verticale sur les lignes de portée) et le second correspond à sa valeur rythmique (la durée pendant laquelle cette note doit être jouée). Dans le cas où il y a plusieurs instruments ou voix dans la partition, les données correspondront à plusieurs séquences : une par instrument ou voix.

Le reste de cette section est organisée comme suit : premièrement, nous introduisons quelques définitions utiles des notions de fouille de données. Ensuite, des approches de fouille sur une ou plusieurs séquences sont présentées.

2.1.1 Définitions

2.1.1.1 Séquence

Soit $I = \{i_1, ..., i_n\}$ un ensemble fini d'éléments (items). Une séquence S sur I est une **liste ordonnée** $< s_1 ... s_l >$, où $s_i \subseteq I$ ($1 \leq i \leq l, l \in \mathbb{N}$) est un itemset, l est appelée la taille de la séquence S, notée $|S| = l$ [Egho, 2014].

2.1.1.2 Base de données séquentielle

La base de données séquentielle $S_{DB} = \{(id_1, s_{id_1}), (id_2, s_{id_2}), ..., (id_m, s_{id_m})\}$ est définie comme un ensemble de m paires (id_i, s_{id_i}) où s_{id} est une **séquence** et id est un identifiant de séquence (exemple : voir le tableau 2.1) [Egho, 2014].

2.1.1.3 Sous-séquence

Une séquence $S =< s_1...s_l >$ est une sous-séquence d'une autre séquence $S' =< s'_1...s'_{l'} >$, notée $S \subseteq S'$, s'il existe des entiers $1 \leq i_1 < i_2 < ... < i_l \leq l'$ tels que $s_j \subseteq s'_{i_j}$, pour tout $j = 1...l$ et $l \leq l'$ [Egho, 2014].

2.1.1.4 Exemple

Dans la table 2.1, la séquence $< \{a\}, \{b\} >$ est une sous-séquence de $< \{a\}, \{b, c, d\} >$.

2.1.1.5 Chaîne (String)

Une chaîne est une séquence dans laquelle chaque itemset ne contient qu'un seul item. En d'autres termes, pour un ensemble fini d'éléments $I = \{i_1, ..., i_n\}$ la chaîne S sur I est une liste ordonnée $< s_1...s_l >$ avec $s_i \in I$.

2.1.1.6 Motif

Un motif est une sous-séquence dans une base de données transactionnelle, base de données séquentielle, base de plusieurs chaînes d'items, ou une seule chaîne d'items. Dans la documentation anglophone, on parle de *patterns* dans le cas des bases de transaction et de *motifs* dans le cas des bases de chaîne d'items et dans le cas d'analyses des séries temporelles.

2.1.1.7 Fréquence du motif

La fréquence du motif (notée *freq*) est le nombre d'occurrences du motif sur une chaîne ou un ensemble de chaînes.

Exemples

— Dans la chaîne $S =< ABABCDABD >$, $freq(AB) = 3$

2.1.1.8 Motif fréquent

On dit qu'un motif est fréquent si et seulement si sa fréquence est supérieure ou égale à un seuil (noté *minFreq*) fixé par l'utilisateur.

Exemples

— Dans la chaîne $S =< ABABCDABD >$, avec $minFreq = 3$, les motifs fréquents sont A ($freq(A) = 3$), B ($freq(B) = 3$), AB ($freq(AB) = 3$)

2.1.2 Propriété Apriori

Le processus de fouille de données consiste à trouver une caractéristique particulière des données. Dans le cas d'une base de données transactionnelle, l'objectif est généralement de rechercher des éléments fréquents ou d'extraire des règles d'association entre les éléments. Ce processus est basé sur deux étapes principales :

— Génération de candidats, des alignements de données (items ou itemsets), en calculant toutes les combinaisons possibles

— Calcul de fréquence pour chaque candidat afin de filtrer les plus rares en fonction d'un *seuil minimal*.

Pour une base de données transactionnelle donnée. En considérant un seuil minimum (appelé $minFreq$), l'objectif est de trouver l'ensemble de tous les itemsets fréquents (motifs fréquents).

Le processus de recherche consiste à tester toutes les combinaisons possibles d'items et à calculer leurs fréquences. Ensuite, l'ensemble de motifs fréquents est conservé. Cette tâche peut s'avérer très coûteuse car elle représente une complexité exponentielle pour calculer tous les motifs fréquents. Comme nous pouvons le voir dans l'exemple de la table 2.1, pour 5 éléments différents (*a, b, c, d, e*), nous avons 5^3 combinaisons possibles pour former des motifs de 3 items.

Afin de réduire cette complexité, l'antimonotonie de la fonction de fréquence (*freq*) est mise en pratique. La fonction *freq* est considérée comme fonction antimonotone car pour une séquence donnée S :

$$\forall S' \subseteq S : freq(S') \geq freq(S)$$

Cette propriété signifie que si on ajoute un item à un motif alors la fréquence du nouveau

motif ne sera pas supérieure à celle du motif original [Agrawal et al., 1994].

2.1.2.1 Algorithme Apriori

Proposé par Agrawal, Apriori est l'un des premiers algorithmes proposés pour la recherche de motifs dans une base de données transactionnelle [Agrawal et al., 1994]. Il commence par calculer des itemsets fréquents à un élément. Ensuite, il forme des motifs de taille supérieure en ajoutant un item à la fois pour chaque itemset pour former un motif plus grand. Ensuite, la fréquence de chaque nouveau motif est calculée. Les nouveaux motifs fréquents sont enregistrés. La dernière étape est répétée en tenant compte de toutes les combinaisons possibles jusqu'à ce qu'aucun nouveau motif ne soit fréquent.

2.1.3 Algorithmes de fouille de séquences

Plusieurs algorithmes ont été proposés dans l'état de l'art en fonction des données d'entrée et de la nature du motif à identifier. GSP (Generalized Sequential Patterns), proposé par [Srikant and Agrawal, 1996], est une adaptation de l'algorithme Apriori pour résoudre les problèmes de fouille de séquences. Il incorpore des contraintes de temps (l'ordre de jeu d'éléments est pris en compte), des fenêtres de temps glissantes et des taxonomies des données séquentielles. Il effectue plusieurs passages sur les données : génération de candidats et comptage de fréquence. Dans cet algorithme, les auteurs ont utilisé la notation *k-sequence* pour désigner une séquence avec k items et C_k pour l'ensemble candidat de *k-sequences*. La première étape de l'algorithme consiste à générer des candidats ($C_{k=1}$). Le processus de génération est effectué en joignant chacune des deux séquences de $C_{k=1}$. La génération de C_{k+1} se fait en joignant chacune des deux séquences s1, s2 en F_k. La jointure se fait comme suit : si la sous-séquence obtenue en enlevant le premier item de s_1 est la même que la sous-séquence obtenue en enlevant le dernier item de s_2. La séquence candidate est ensuite générée en étendant s_1 avec le dernier élément en s_2. Si le dernier élément de s_2 est un itemset alors on parle d'une jointure de type I-extension, si c'est un item alors on parle d'une S-extension.

Afin de réduire le nombre de séquences dans C_k, les séquences candidates sont organisées en arbre de hachage. Les nœuds de l'arbre de hachage contiennent soit une liste de séquences

comme nœud feuille, soit une table de hachage comme nœud intérieur. Pour trouver quelles séquences candidates sont incluses dans une séquence de données, GSP traverse l'arbre en appliquant une fonction de hachage à chaque élément de la séquence de données. Lorsqu'une feuille est atteinte, elle contient un candidat potentiel pour la séquence de données.

GSP intègre également les contraintes de la taille minimale et maximale des gaps.

D'autres approches ont été proposées en se basant sur une projection verticale des données. Les deux principaux algorithmes en utilisant le format de base de données verticale (i.e. une base de données où chaque entrée représente un item et indique la liste des séquences où l'item apparaît et la (les) position(s) où il apparaît) sont SPADE et SPAM [Zaki, 2001][Ayres et al., 2002]. Ces deux algorithmes diffèrent principalement par leur processus de génération de candidats, que nous examinerons par la suite.

L'algorithme SPAM prend en entrée une base de données de séquences SDB et le seuil de fréquence minimale. SPAM scanne d'abord une fois la base de données d'entrée pour construire la représentation verticale de la base de données et l'ensemble des éléments fréquents F1. Pour chaque élément fréquent $s \in F1$, SPAM explore récursivement les motifs candidats commençant par le préfixe s. Ensuite, il génère deux ensembles de candidates. Le premier ensemble S_n représente les éléments à joindre par une *S-extension*. La *S-extension* d'une extension séquentielle $I_1, I_2, ...I_h$ avec un élément x est défini comme $I1, I2, ...Ih, x$. Le deuxième ensemble S_i représente les éléments à ajouter par *I-extension*. La *I-extension* d'un motif séquentiel $I_1, I_2, ...I_h$ avec un élément x est défini comme $I_1, I_2, ...I_h \cup \{x\}$.

Pour chaque motif candidat généré par une extension, SPAM calcule son support (fréquence) pour déterminer s'il est fréquent. Ceci se fait en faisant une opération de jointure et en comptant le nombre de séquences où le motif apparaît. La représentation IdList utilisée par SPAM est basée sur des bitmaps pour accélérer les opérations [Ayres et al., 2002]. Si le motif est fréquent, il est alors utilisé récursivement pour générer des extensions de motifs commençant par ce motif. SPAM réduit l'espace de recherche en ne faisant pas étendre les motifs non fréquents (propriété Apriori).

L'algorithme SPAM a été repris et amélioré par l'équipe de Philippe Fournier Viger, [Fournier-Viger et al., 2014], qui a utilisé une structure de données appelée *Co-Occurrence*

Map. Cette structure de données est définie par un tableau associatif permettant de sauvegarder le lien entre chaque item et les items qui lui succèdent. Pour pouvoir l'intégrer dans SPAM et SPADE, ils ont définie pour chaque type d'extension (*I-extension* ou *S-extension*) une *Co-occurrence Map*. Cette structure de données permet de réduire l'espace de recherche des motifs candidats et donc évite de former les motifs non fréquents [Fournier-Viger et al., 2014]. Cette intégration a donnée lieu a des nouvelles versions des algorithmes SPAM et SPADE qui sont nommées respectivement CM-SPAM et CM-SPADE. L'ensemble des expériences menées par les auteurs ont montré les meilleures performances suite à l'utilisation et l'intégration de la *Co-Occurrence Map* dans les algorithmes étudiés [Fournier-Viger et al., 2014].

Maintenant, la plupart des travaux tentent d'extraire des motifs de séquence en utilisant des techniques issues du Big Data. Une des dernières études a été écrite par Fournier-Viger et al. [Fournier-Viger et al., 2017] qui rendent également disponible la plupart des algorithmes d'extraction dans le projet SPMF [1].

2.1.4 Épisodes mining

Quand les données sont présentées sous forme de séries temporelles on parle de fouille d'épisodes. Dans une base de séquences, la fréquence d'un motif correspond au nombre de séquences dans lesquelles le motif apparaît au moins une fois et plusieurs occurrences du motif dans la même séquence n'ont aucun impact sur sa fréquence. Alors que dans le cas d'un épisode dans une séquence d'événements, la fréquence représente le nombre d'occurrences du motif dans cette séquence.

Un des premiers travaux dans ce domaine a été réalisé par [Mannila et al., 1997] où la fouille d'épisodes a été utilisée pour analyser les flux d'alarmes d'un réseau de télécommunication.

Dans le cadre de l'épisode, l'entrée est une séquence d'événements. Étant donné un ensemble d'événements E, un événement est défini par une paire (e, t), où $e \in E$ et t est le moment de l'événement. Le principe de l'approche de fouille d'épisodes consiste à considérer une fenêtre glissante de largeur fixe (voir Figure 2.2), puis à considérer une sé-

1. http ://www.philippe-fournier-viger.com/spmf/

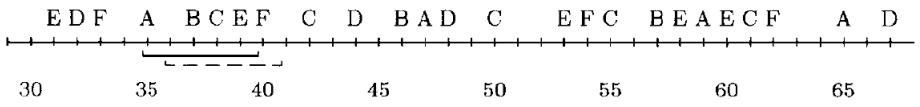


FIGURE 2.2 – Séquence d'évènements avec deux fenêtres de taille 5 [Mannila et al., 1997]

quence d'événements comme une séquence de fenêtres se chevauchant partiellement. Si nous considérons que l'événement se produit au même moment que les itemsets, l'extraction fréquente d'épisodes est identique à l'extraction fréquente d'éléments (Algorithme d'Apriori) [Laxman and Sastry, 2006].

Dans certaines applications, la taille de la fenêtre n'est pas connue à l'avance. De plus, la taille de la fenêtre glissante peut être différente pour chaque règle d'épisode (sorte de succession d'évènements). Pour faire face à cette classe d'applications, il a été proposé dans [Casas-Garriga, 2003] d'utiliser une contrainte d'écart maximum (*maxGap*) qui impose le temps écoulé maximal entre deux événements consécutifs dans les occurrences d'un épisode. Cette contrainte est similaire à la contrainte d'espace maximum gérée par les algorithmes proposés pour trouver des motifs séquentiels fréquents dans une base de séquences. Ce travail a été repris et complété par Meger et Rigotti [Méger and Rigotti, 2004] qui ont présenté un algorithme solide et complet pour explorer les règles d'épisodes sous la contrainte d'écart maximum et proposé de trouver, pour chaque règle, la taille de la fenêtre correspondant à un maximum de confiance local.

2.1.5 Fouille de chaînes

Lorsque le temps séparant les évènements successifs dans une séquence d'évènements est fixe on parle de chaîne. La recherche de motifs dans une ou plusieurs chaînes consiste à extraire les sous chaînes fréquentes. Il existe deux types de motifs : contigus et non contigus. Le motif contigu est une simple sous-chaîne telle que tous les éléments sont collés les uns des autres. Un motif non contigu est une sous-chaîne qui contient des sauts ou *gaps* ou *wildcards* en notation anglophone. Dans le domaine biologique, on extrait les motifs dits approximatifs. Les motifs approximatifs consistent en des motifs définis par une longueur, une distance et une fréquence. Parmi les algorithmes les plus célèbres, associés à la recherche des motifs

approximatifs, on peut citer FLAME et ACME [Floratou et al., 2011] [Sahli et al., 2013].

Exemple nous considérons la séquence S = <ABABCDCABDCE>, donc :

— le motif AB est contigu.

— le motif AB * C est non contigu, de sorte que le symbole * représente 0 ou plusieurs éléments possibles dans cette position. Ce symbole désigne le *gap*.

— le motif ACB est un motif approximatif du motif ABC car la distance de Hamming, qui correspond au nombre de substitutions, entre les deux motifs est égale à 2.

Il existe une intersection entre la fouille de séquences, la fouille d'épisodes et la fouille de chaînes. Si nous considérons un ensemble de chaînes en entrée et que chaque élément de l'alphabet correspond à un itemset avec une seule occurrence par chaîne, l'extraction de chaînes fréquentes non contiguës sera similaire à la fouille de séquences [Laxman and Sastry, 2006].

Sur les données musicales, certaines approches ont déjà été proposées dans la littérature. Généralement, les données musicales sont présentées sous forme séquentielle en prenant en compte les informations mélodiques et / ou rythmiques. Ensuite, un processus d'extraction est appliqué afin de trouver des motifs à l'intérieur de la partition, ou des motifs communs d'un ensemble de partitions. Dans la sous-section suivante, nous présenterons des travaux spécifiques à ce contexte.

2.1.6 Fouille de données sur des données musicales

Dans notre cadre applicatif, un motif musical correspond à un morceau de musique apparaissant à un nombre minimal de positions dans la partition. Ce motif peut être mélodique, rythmique ou les deux, selon la nature des caractéristiques de l'événement musical. Parmi les premiers travaux dans le domaine de la musique, Hsu *et al.* ont introduit une méthode pour identifier les motifs fréquents dans les partitions, en ne considérant que les informations mélodiques dans une seule séquence [Hsu et al., 1998]. Ensuite, cette méthode a été améliorée par Liu *et al.* [Liu et al., 1999], afin de trouver des motifs non triviaux (*i.e.* motifs qui n'ont pas de sous-motifs avec la même fréquence). L'algorithme de Liu est une approche de fouille de chaînes qui vise à récupérer des motifs musicaux. Cet algorithme, comme décrit

```
/* input: the music feature string S of a music object */
/* output : the set of all non-trivial repeating patterns and
their frequencies in S */
0:   begin
1:       RP = φ
2:       RP[1] = Find_RP1(S)
3:       k = 0
4:       while RP[2^k] <> φ do
5:       begin
6:           k = k + 1
7:           RP[2^k] = Find_RP2K(RP[2^(k-1)])
8:       end
9:       RP[L] = Find_Longest_RP(RP[2^k])
10:      Build_RP_Tree()
11:      Refine_RP_Tree()
12:      Generate_Non-trivial_RP(RP)
13:      return RP
14:  end
```

FIGURE 2.3 – Algorithme de fouille de motifs non triviaux dans une seule séquence

dans [Liu et al., 1999], est basé sur la structure de données tel que : un motif est défini, dans l'ordre, par trois éléments ; valeur, fréquence et positions dans la séquence.

Exemple Pour une séquence S = $< ABABCDCABDCE >$, le motif AB est défini : AB = (AB, 3, [1, 3, 8])

Les auteurs ont défini un algorithme complet pour trouver des motifs non triviaux. Cet algorithme est présenté à la figure 2.3

Cet algorithme utilise un ensemble de notations telles que :

— RP : l'ensemble des motifs fréquents

— RP [k] : l'ensemble des motifs fréquents de longueur k

L'algorithme commence par calculer l'ensemble des motifs fréquents de longueur 1. Ensuite, il essaie de joindre chaque motif de même longueur (de l'ordre de 2^k). Les autres combinaisons possibles sont calculées à l'aide d'une structure de données arborescente. La prochaine étape consiste à rechercher les motifs les plus longs et à filtrer les plus simples.

La jointure de deux motifs (concaténation) est définie comme suit :
Soit deux motifs $m_1, m_2 \in RP$ définis comme suit : $m_1 = (X, freq(X), P)$ et $m_2 = (Y, freq(Y), Q)$, la jointure de m_1 et m_2 donne $m_3 = (Z, freq(Z), R)$ tels que $Z =$ concaténation de (X, Y) et $R = \{p_i \in P, p_i + |X| \in Q\}$.

Exemple : On considère la séquence S donnée dans l'exemple 2.1.5. On a les motifs m_1 = ('AB', 3, [1, 3, 8] et m_2 = ('C', 2, [5, 7, 11]), la jointure de m_1 et m_2 donne C = ('ABC', 1, [3])

Par ailleurs, d'autres travaux étudient la représentation des partitions. Lorsque les données musicales sont au format audio, les valeurs mélodiques peuvent être extraites d'abord. Ensuite, les motifs mélodiques peuvent être extraits en utilisant, par exemple, une approche de fouille d'épisodes. Pour exploiter à la fois les informations mélodiques et rythmiques, Béatrice Fuchs suggère de transformer les données de musique entrées en un flux de données, puis d'appliquer l'extraction fréquente d'itemsets [Fuchs, 2012]. Enfin, le travail le plus proche du notre a été réalisé par Jiménez *et al.* qui a conçu un algorithme capable de trouver les motifs de musique de type transposé en faisant une correspondance exacte [Jiménez et al., 2011]. Dans cette approche, la partition est transformée en une séquence de notes et les motifs sont extraits par un algorithme de fouille de séquence, appelé *SSMiner*. Comme dans ce dernier travail, nous nous intéressons à la recherche de motifs et de leurs variantes. Pour un motif donné, nous pouvons avoir trois variantes possibles : les formes transposées, inversées et en forme miroir (cf. figure 2).

Ces algorithmes ne conviennent pas à notre tâche car nous souhaitons identifier des motifs avec des *gaps* alors qu'ils ne considèrent que des motifs contigus. De plus, ils ne traitent qu'une séquence alors que nous considérons des partitions avec un ou plusieurs instruments correspondant à un ensemble de séquences, un par instrument. Cependant, nous retenons de l'algorithme proposé par Liu et al. la structure de codage des motifs, que nous adaptons pour gérer les *gaps*.

Le tableau 2.2 contient une comparaison des quelques travaux présentés et on voit qu'aucune approche n'est adaptée pour extraire les motifs contigus et non contigus sur une ou plusieurs séquences.

Algorithme	fouille de séquence	fouille de chaines	Un séquence	motifs contigus	Repeating items	min gap	max gap	Objectif
cSPADE	OUI	OUI	NON	NON	NON	OUI	OUI	Association rule mining
CM-SPAM	OUI	OUI	NON	NON	NON	OUI	OUI	Sequence mining
FLAME	NON	OUI	NON	OUI	OUI	NON	NON	Approximative motif mining
ACME	NON	NON	OUI	OUI	OUI	NON	NON	Parallel approximative motif mining
Liu	NON	NON	OUI	OUI	OUI	NON	NON	melodic sequence mining

TABLE 2.2 – Tableau récapitulatif des approches présentées

Par conséquence, dans ce travail, nous proposons un nouvel algorithme, **CSMA**, capable d'extraire des motifs dans une ou plusieurs chaînes avec des contraintes liées à la fréquence minimale, à la taille maximale de *gaps* autorisés par motif, le nombre de *gaps* autorisés par motif et à la longueur minimale et maximale des motifs. Dans **CSMA**, les positions de motif dans la partition sont enregistrées. Ces positions sont utiles au musicologue pour analyser la partition. Ils sont également exploités pour construire de nouvelles représentations de la partition utilisées pour extraire des variantes des motifs musicaux. Cet algorithme est décrit dans la section suivante.

2.2 Extraction de motifs sur une ou plusieurs séquences : Approche CSMA

Dans cette section, nous introduisons un nouvel algorithme, appelé **CSMA** (**C**onstrained **S**tring **M**ining **A**lgorithm), pour extraire tous les motifs fréquents dans une ou plusieurs chaînes. CSMA effectue une recherche de motifs en fonction de contraintes liées à la fréquence, aux écarts entre les motifs, à la longueur minimale et maximale des motifs. Il utilise la même structure que l'algorithme de Liu *et al.* Mais avec des modifications pour prendre en compte les *gaps* [Liu et al., 1999].

Un motif m_i est défini par trois éléments $m_i = (X, freq(X), P_i = [(p_{i1}, len_{i1}), \ldots, (p_{in}, len_{in})])$ tel que X correspond à la valeur du motif (liste ordonnée d'éléments), $freq(X)$ correspond à sa fréquence et P_i à ses positions et à ses longueurs. Dans l'ensemble des positions, appelé P_i, la j^{me} position du i^{me} motif est notée p_{ij} et len_{ij} sa longueur à cette position. Afin de simplifier les notations, nous omettons le premier index i lorsque nous considérons un seul motif. De plus, nous illustrons notre propos avec l'exemple suivant.

Exemple 2.2.1. Étant donné la séquence $S = <ABABCDCABDCE>$:

— Le motif m correspondant à A est défini par m = (A, fréq (A) = 3, P = [(1,1), (3,1), (8,1)]) ; $p_1 = 1$, $len_1 = 1$, $p_2 = 3$, $len_2 = 1$, $p_3 = 8$ et $len_3 = 1$.

— Le motif m correspondant à $C * D$, où * désigne un saut ou un *gap*, est défini par m = (C * D, freq (C * D) = 2, P = [(5,2), (7,4)]) ; $p_1 = 5$, $len_1 = 2$, $p_2 = 7$ et $len_2 = 4$. Ici, puisque $len_2 = 4$, le *gap* entre C et D à partir de la deuxième position de C est de 2.

Le pseudo-code de **CSMA** est donné dans l'algorithme 1. Cet algorithme prend en entrée une séquence S, un seuil minimum de fréquence $minFreq$, une longueur maximale des *gaps* autorisés à l'intérieur des motifs $maxGap$, le nombre de gaps maximum autorisés par motif $\#Gaps$, une longueur du motif minimale $minLength$ et une longueur du motif maximale $maxLength$.

La table 2.3 montre un exemple du processus de la séquence $S = <ABABCDCABDCE>$ avec $minFreq = 2$, $maxGap = 1$, $minLength = 1$ et $maxLength = 4$. Les éléments de S et leurs positions d'index sont illustrés à la figure 2.4.

Algorithm 1: Constrained String Mining Algorithm (CSMA).

Input : Sequence S, minFreq,maxGap, #Gaps, minLength and maxLength
Output: $\mathcal{F}$: L'enseemble des motifs fréquents

```
1  begin
2      K = 1 ;
3      F_1 = ∅ ;
4      COMPUTE(S, minFreq, F_1) ;
5      while F_K ≠ ∅ do
6          K = K + 1;
7          for m_i = (X, freq(X), P_i) ∈ F_{K-1} do
8              frequentPosition = p_{iminFreq} + len_{iminFreq} ;
9              C = GEN_CAND(frequentPosition, F_1);
10             for m_j = (Y, freq(Y), P_j) ∈ C do
11                 m_l = JOIN(m_i, m_j, maxGap, #Gaps, maxLength);
12                 if freq(Z) ≥ minFreq then
13                     F_K = F_K ∪ {m_l};
14                 end
15             end
16         end
17     end
18     F = ⋃_{k≤K} F_k
19     FILTER(F, minLength) ;
20     return F ;
21 end
```

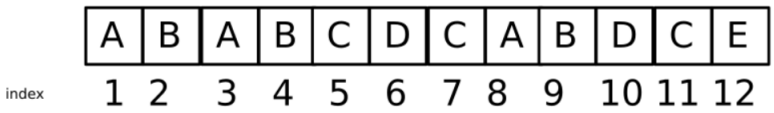


FIGURE 2.4 – La séquence s=<ABABCDCABDCE>

La première étape de CSMA (ligne 4, algorithme 1) consiste à calculer l'ensemble $\mathcal{F}_1$ contenant les motifs fréquents de longueur un. Pour cela, la fonction **COMPUTE** commence par énumérer tous les éléments possibles à partir de S et calcule leurs fréquences. Les éléments dont la fréquence est supérieure ou égale à $minFreq$ sont ajoutés à $\mathcal{F}_1$.

Dans l'exemple de la table 2.3, l'ensemble des éléments est $I = \{A, B, C, D, E\}$ et $\mathcal{F}_1 = \{m_1 = (A, 3, P_1 = [(1, 1), (3, 1), (8, 1)]), m_2 = (B, 3, P_2 = [(2, 1), (4, 1), (9, 1)]), m_3 = (C, 3, P_3 = [(5, 1), (7, 1), (11, 1)]), m_4 = (D, 2, P_4 = [(6, 1), (10, 1)])\}$; le motif contenant E n'appartient pas à $\mathcal{F}_1$ car il n'apparaît pas au moins $minFreq$ fois.

Pour obtenir l'ensemble $\mathcal{F}_K$ contenant les motifs de longueur égale à ($K = 2$), une opération de jonction est considérée (ligne 7) entre chaque élément m_i de $\mathcal{F}_{K-1}$ et chaque élément appartenant à $\mathcal{F}_1$. Afin d'élaguer l'espace de recherche, nous calculons la position sur laquelle le motif m_i est considéré comme fréquent (ligne 8). Cette position, appelée $frequentPosition$, correspond à la somme de l'index de $m_i \in \mathcal{F}_{K-1}$ à la position $minFreq$ et à la longueur de m_i pour la même position.

Exemple 2.2.2. Dans l'exemple 2.2.1, avec $minFreq = 2$, si on considère le motif $m_1 = (A, 3, [(1, 1), (3, 1), (8, 1)])$, sachant que sa deuxième position correspond à $p_{12} = 3$ et sa longueur à cette position est $len_{12} = 1$, la $frequentePosition$ de m_1 est égal à $p_{12} + len_{12} = 3 + 1 = 4$.

Ensuite, les motifs candidats sont générés à l'aide de la fonction **GEN_CANDIDATES**. Étant donné la position $frequentPosition$ et l'ensemble de tous les motifs fréquents de longueur un déjà extraits, cette fonction calcule un ensemble $\mathcal{C} \subseteq \mathcal{F}_1$ de motifs candidats pouvant être joints à m_i à partir de $frequentPosition$. Notre stratégie d'élagage est basée sur le fait qu'un motif $m_j \in \mathcal{F}_1$ ne peut pas être candidat pour $m_i \in \mathcal{F}_{K-1}$ s'il n'apparaît pas après $frequentPosition$ puisque le résultat de la jointure de m_i et m_j ne peut pas être fréquent. L'implémentation de la fonction **GEN_CANDIDATES** est détaillée dans Algorithme 2.

K	$\mathcal{F}_{K-1}$	*frequentPosition*	$\mathcal{C}$	**nouveau motif**	**accepté**	**violations**
-	$\emptyset$	-	-	-		
1	$A, 3, [(1,1),(3,1),(8,1)]$	-	-	-	✓	
1	$B, 3, [(2,1),(4,1),(9,1)]$	-	-	-	✓	
1	$C, 3, [(5,1),(7,1),(11,1)]$	-	-	-	✓	
1	$D, 2, [(6,1),(10,1)]$	-	-	-	✓	
2	$A, 3, [(1,1),(3,1),(8,1)]$	4	A	$A*A, 1, [(1,3),\underline{(3,5)}]$	X	*maxGap*, *minFreq*
2	$A, 3, [(1,1),(3,1),(8,1)]$	4	B	$A*B, 3, [(1,2),(3,2),(8,2)]$	✓	
2	$A, 3, [(1,1),(3,1),(8,1)]$	4	C	$A*C, 1, [(3,3)]$	X	*minFreq*
2	$A, 3, [(1,1),(3,1),(8,1)]$	4	D	$A*D, 1, [\underline{(3,4)},(8,3)]$	X	*maxGap*, *minFreq*
2	$B, 3, [(2,1),(4,1),(9,1)]$	5	A	$B*A, 1, [(2,2),\underline{(4,5)}]$	X	*maxGap, maxLength*, *minFreq*
2	$B, 3, [(2,1),(4,1),(9,1)]$	5	B	$B*B, 1, [(2,3),\underline{(4,6)}]$	X	*maxGap, maxLength*, *minFreq*
2	$B, 3, [(2,1),(4,1),(9,1)]$	5	C	$B*C, 2, [(4,2),(9,3)]$	✓	
2	$B, 3, [(2,1),(4,1),(9,1)]$	5	D	$B*D, 2, [(4,3),(9,2)]$	✓	
2	$C, 3, [(5,1),(7,1),(11,1)]$	8	A	$C*A, 1, [(7,2)]$	X	*minFreq*
2	$C, 3, [(5,1),(7,1),(11,1)]$	8	B	$C*B, 1, [(7,3)]$	X	*minFreq*
2	$C, 3, [(5,1),(7,1),(11,1)]$	8	C	$C*C, 1, [(5,3),\underline{(7,5)}]$	X	*maxGap, maxLength*, *minFreq*
2	$C, 3, [(5,1),(7,1),(11,1)]$	8	D	$C*D, 1, [(5,2),\underline{(7,4)}]$	X	*maxGap*, *minFreq*
2	$D, 2, [(6,1),(10,1)]$	11	C	$D*C, 2, [(6,2),(10,2)]$	✓	
3	$A*B, 3, [(1,2),(3,2),(8,2)]$	5	A	$A*B*A, 1, [(1,3),\underline{(3,5)}]$	X	*maxGap, maxLength*, *minFreq*
3	$A*B, 3, [(1,2),(3,2),(8,2)]$	5	B	$A*B*B, 1, [(1,4),\underline{(3,7)}]$	X	*maxGap, maxLength*, *minFreq*
3	$A*B, 3, [(1,2),(3,2),(8,2)]$	5	C	$A*B*C, 2, [(3,3),(8,4)]$	✓	
3	$A*B, 3, [(1,2),(3,2),(8,2)]$	5	D	$A*B*D, 2, [(3,4),(8,3)]$	✓	
3	$B*C, 2, [(4,2),(9,3)]$	12	-	-		
3	$B*D, 2, [(4,3),(9,2)]$	11	C	$B*D*C, 2, [(4,4),(9,3)]$	✓	
3	$D*C, 2, [(6,2),(10,2)]$	12	-	-		
4	$A*B*C, 2, [(3,3),(8,4)]$	12	-	-		
4	$A*B*D, 2, [(3,4),(8,3)]$	11	C	$A*B*D*C, 2, [\underline{(3,5)},(8,4)]$	X	*maxLength*, *minFreq*
4	$B*D*C, 2, [(4,4),(9,3)]$	12	-	-		

TABLE 2.3 – Extraction des motifs de la séquence $S =< ABABCDCABDCE >$ avec $minFreq = 2$, $maxGap = 1$, $\#gaps = 1$, et $maxLength = 4$

Algorithm 2: *GEN_CANDIDATES*

Input : une position *position*, un ensemble de motifs $\mathcal{M}$
Output: L'ensemble de motifs candidats $\mathcal{C}$

```
1  begin
2      C = ∅ ;
3      for m = (X, freq(X), P = [⋃_{i≤freq(X)} (p_i, len_i)]) ∈ M do
4          i = 1 ;
5          estCandidat = Faux ;
6          while ( i ≤ freq(X) ∧ estCandidat = Faux) do
7              if p_i ≥ position then
8                  estCandidat = Vrai;
9              end
10             i = i + 1;
11         end
12         if estCandidat = Vrai then
13             C = C ∪ {m} ;
14         end
15     end
16     return C ;
17 end
```

Cette fonction retourne l'ensemble de motifs ayant au moins une occurrence à partir de *position*. Pour chaque motif $m \in \mathcal{M}$, nous recherchons des positions venant après *position*. Si nous trouvons au moins une position, nous ajoutons le motif $m \in \mathcal{M}$ à l'ensemble de motifs candidats $\mathcal{C}$.

Par exemple, dans l'exemple du tableau 2.3, illustré à la figure 2.5, où les couleurs indiquent les positions de chaque motif dans la séquence S, pour le motif $m_4(D)$, Les motifs A, B et D n'apparaissent pas après son $frequentPosition$, ce qui équivaut à 11, DA, DB et DD ne pouvait pas être fréquent et le seul candidat pour ce motif est C.

Une fois la sélection des motifs candidats terminée, un ensemble $\mathcal{C} \subseteq \mathcal{F}_1$ est calculé. Ensuite, l'opération de jonction est effectuée pour le motif sélectionné m_i avec chaque élément $m_j \in \mathcal{C}$. La jointure (concaténation) de deux motifs est définie comme suit :
Soit deux motifs $m_1 \in \mathcal{F}_{K-1}$ et $m_2 \in \mathcal{F}_1$ définis comme $m_1 = (X, freq(X), P_1 = [\bigcup_{i \leq freq(X)} (p_{1i}, len_{1i})])$ et $m_2 = (Y, freq(Y), P_2 = [\bigcup_{j \leq freq(Y)} (p_{2j}, len_{2j})])$, la jointure de m_1 et m_2 donne $m_3 \in \mathcal{F}_K$ qui est défini par $m_3 = (Z, freq(Z), P_3)$ tels que Z est la concaténation de (X, Y) et P_3 est un ensemble de positions p_{3k} et de longueurs

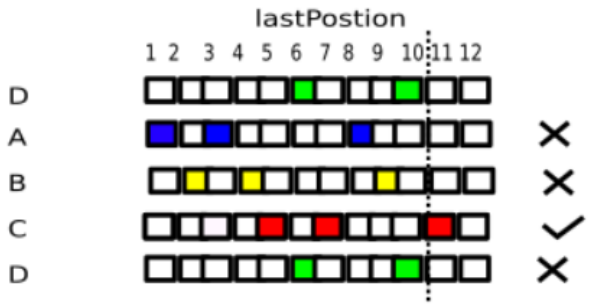


FIGURE 2.5 – Sélection des motifs candidats basée sur la valeur de *frequentPosition* du motif D

len_{3k}. Une position $p_{3k} \in P_3$ est égale à p_{1i} si et seulement si $\exists j \leq freq(Y)$ telle que les trois conditions soient vérifiées :

$$\begin{cases} 0 \leq p_{2j} - (p_{1i} + len_{1i}) \leq maxGap & (1) \\ i = \arg\min_{l \leq freq(X)}(p_{2j} - (p_{1l} + len_{1l})) & (2) \\ p_{2j} + len_{2j} - p_{1i} \leq maxLength & (3) \end{cases}$$

La première condition garantit la contrainte associée à $maxGap$. Elle permet de calculer toutes les valeurs de *gaps* possibles pour un p_{2j} donné en tenant compte de tous les p_{1i}. Seules les valeurs d'écart positives inférieures ou égales à $maxGap$ sont conservées. Ainsi, si la valeur de $maxGap$ est strictement positive alors le $\#gaps$ est décrémenté de 1 s'il est strictement positif sinon s'il est nul alors l'opération de jointure n'est plus possible. Ensuite, avec la deuxième condition, l'indice i de p_{1i} qui a une valeur minimale pour le *gap* est calculée. Enfin, dans la troisième condition, la longueur du motif est calculée afin de vérifier s'il respecte la contrainte $maxLength$.

Plus précisément, les positions p_{1i} de m_1 et p_{2j} de m_2 vérifiant les conditions précédentes permettent de définir la position p_{3k} correspondant à p_{1i} pour m_3 et la longueur len_{3k} est égale à $p_{2j} + len_{2j} - p_{1i}$. La fréquence de m_3 est égale au nombre de positions dans P_3.

En résumé, la valeur du *gap* est calculée entre chaque position i de m_1 à partir d'une position fixe j de m_2. La valeur du *gap* est définie par $gap = p_{2j} - (p_{1i} + len_{1i})$. Ensuite, nous conservons i avec une valeur d'écart positif minimum qui est inférieure ou égale à $maxGap$; cela correspond aux conditions (1) et (2). La longueur de la séquence $len = (p_{2j} + len_{2j}) - p_{1i}$

est calculée et si elle ne dépasse pas $maxLength$, la position (p_{1i}, len) est ajouté à P_3. On peut remarquer que la fréquence de chaque nouveau motif est inférieure ou égale à ses sous-motifs. Cela signifie que l'opération de jonction est anti-monotone (propriété Apriori), ce qui permet de ne pas explorer tout l'espace de recherche.

Exemple 2.2.3. Dans l'exemple de la table 2.3, considérons l'opération de jointure des motifs m_1 = (A * B, 3, [(1,2), (3,2), (8,2)] et m_2 = (B, 3, [(2,1), (4,1), (9,1)]). Pour chaque p_{2j} de m_2 on cherche la position p_{1i} de m_1 en vérifiant les conditions (1), (2) et (3).

Ainsi, pour $(p_{21}, len_{21}) = (2, 1)$, nous calculons différentes valeurs possibles du *gap* :

— $(p_{11}, len_{11}) = (1, 2)$: gap = 2- (1 + 2) = -1

— $(p_{12}, len_{12}) = (3, 2)$: gap = 2- (3 + 2) = -3

— $(p_{13}, len_{13}) = (8, 2)$: gap = 2- (8 + 2) = -8

Aucune des valeurs de *gap* n'est supérieure à zéro. Il n'existe donc aucun moyen possible de joindre m_1 et m_2 pour la position p_{21} puisque la condition (1) n'est pas vérifiée.

Pour $(p_{22}, len_{22}) = (4, 1)$, nous calculons différentes valeurs possibles de *gap* :

— $(p_{11}, len_{11}) = (1, 2)$: gap = 4- (1 + 2) = 1

— $(p_{12}, len_{12}) = (3, 2)$: gap = 4- (3 + 2) = -1

— $(p_{13}, len_{13}) = (8, 2)$: gap = 4- (8 + 2) = -6

Seule la position p_{11} satisfait à la première condition en considérant le paramètre contrainte $maxGap = 1$ et $\#gaps > 0$. Selon la deuxième condition, nous conservons la position $i = 1$. Maintenant, vérifions la troisième condition pour calculer la longueur du motif $len = (p_{22} + len_{22}) - p_{11} = (4 + 1) - 1 = 4$. Comme len est inférieur à $maxLength$, il est possible de joindre m_1 avec m_2 pour p_{22}.

De la même manière, pour $(p_{23}, len_{23}) = (9, 1)$, nous calculons différentes valeurs d'intervalle possibles :

— $(p_{11}, len_{11}) = (1, 2)$: gap = 9-(1+2) = 6

— $(p_{12}, len_{12}) = (3, 2)$: gap = 9-(3+2) = 4

— $(p_{13}, len_{13}) = (8, 2)$: gap = 9-(8+2) = -1

La valeur positive minimale est 4 et correspond à p_{12}. En raison de la contrainte $maxGap$ (1), cette position n'est pas conservée. Il est donc impossible de joindre m_1 avec m_2 pour p_{22}.

À la fin, il n'y a qu'un moyen possible de joindre m_1 et m_2 en vérifiant les trois conditions qui correspondent respectivement à p_{11} et à p_{22}. Comme *minFreq* doit être supérieur ou égal à 2, le motif de résultat $A * B * B$ n'a pas pu être conservé car sa fréquence est de 1.

Une fois que $\mathcal{F}_2$ est obtenu, les autres ensembles $\mathcal{F}_K$ de longueur $K > 2$ sont calculés et la boucle *while* s'arrête lorsqu'aucun nouveau motif n'est généré. En conclusion, dans cet exemple, nous obtenons le résultat suivant : $\mathcal{F}_1 = \{A, B, C, D\}$, $\mathcal{F}_2 = \{A * B,\ B * C, B * D, D * C\}$, $\mathcal{F}_3 = \{A * B * C, A * B * D, B * D * C\}$ et $\mathcal{F}_4 = \emptyset$.

À l'étape suivante (ligne 19 de l'algorithme1), tous les motifs fréquents d'ordre $k \leq K$ sont placés dans $\mathcal{F}$. Ensuite, les motifs qui ne respectent pas la contrainte *minLength* sont supprimés de $\mathcal{F}$. La fonction **FILTER** analyse chaque motif $m = (X, freq(X), P = [\bigcup_{i \leq freq(X)} (p_i, len_i))) \in \mathcal{F}$ et s'il trouve une position pour laquelle len_i est inférieur à *minLength*, il le supprime de l'ensemble P. Au final, la valeur $freq(X)$ est mise à jour et si elle est inférieure à *minFreq*, le motif est supprimé de $\mathcal{F}$. Dans l'exemple, comme *minLength* est égal à 1, $\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2 \cup \mathcal{F}_3$. Dans le tableau 2.3, la colonne *violation* indique les contraintes violées qui échouent à l'opération de jointure.

La version par défaut de CSMA recherche des motifs dans une seule chaîne. Or, pour un morceau de musique donné, nous pouvons être intéressés par l'identification de motifs pour différents instruments, notamment dans plusieurs séquences, une par instrument. Pour cela on propose une extension pour CSMA pour extraire des motifs dans un ensemble de chaînes. L'adaptation est effectuée dans l'opération de jointure en ajoutant aux premières conditions une nouvelle condition selon laquelle " p_{1i} et p_{2j} devraient appartenir à la même chaîne". Dans la suite, pour faire la différence entre les deux versions, nous avons appelé la première CSMA1 et la version adaptée CSMA2. L'évaluation de ces deux versions de CSMA est détaillée dans la section 2.4.

2.3 Extraction des motifs musicaux et de leurs variantes

2.3.1 Définition des variantes des motifs musicaux

Les caractéristiques basées sur des symboles musicaux sont généralement représentées par des valeurs mélodiques et rythmiques. L'information mélodique correspond à une lettre alphabétique, suivie de la valeur d'octave pouvant être encodée au format MIDI. Par exemple, une note C (Do) sur l'octave 5 correspond à 72 en encodage MIDI. L'information rythmique correspond à la durée de la note. L'objectif de notre travail est d'identifier des motifs musicaux mélodiques, rythmiques ou les deux (séquence de hauteur (*pitch* en notation Anglaise)).

Notre but dans cette section est de présenter les règles développées pour la recherche des variantes de motifs mélodiques : transposée, inversée ou miroir, comme illustré sur la figure 2.6. Un motif transposé est une sous séquence mélodique avec la même variation tonale qu'une autre sous séquence dans la partition, par exemple dans la figure 2.6, m_2 est une forme transposée de m_1. Un motif inversé est une sous séquence caractérisée par la variation tonale inversée d'une sous séquence dans la partition, comme par exemple m_4 et m_5. Enfin, un motif miroir correspond à un positionnement symétrique de notes par rapport à une note centrale, comme par exemple les motifs m_5 et m_6. Ces formes sont utiles pour caractériser un compositeur ou une partition. Elles peuvent être utilisées comme signature pour d'autres tâches de fouille de données telles que la classification supervisée ou non supervisée ou l'identification du compositeur.

2.3.2 Détection des variantes

Nous proposons une méthode, basée sur un prétraitement de la séquence mélodique initiale S, capable de détecter les variantes mélodiques. À cette fin, nous considérons la séquence d'intervalles entre les notes consécutives de S. De cette manière, trois nouvelles séquences sont construites. La première, notée V, correspond à la variation mélodique entre deux notes. Par exemple, dans la figure 2.6, la variation entre la note dans la première position (60) et la note dans la seconde (61) est égale à 1 (61-60). La deuxième séquence, notée $-V$, est obtenue en prenant l'opposé de chaque valeur dans la séquence V, de sorte que chaque valeur positive

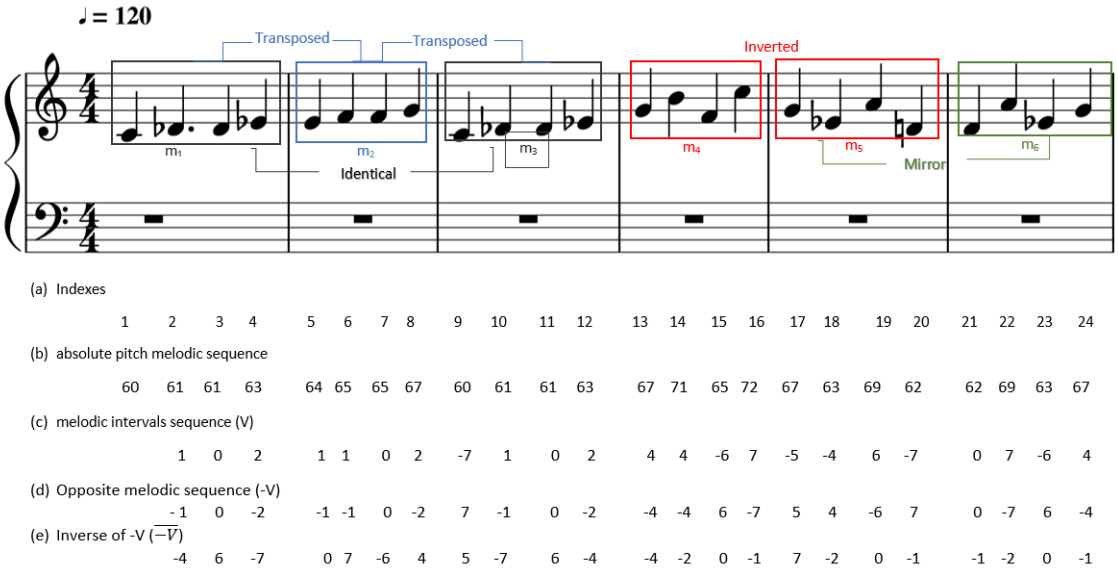


FIGURE 2.6 – Motifs musicaux : motifs identiques, motifs transposés, motifs inversés et motifs miroirs

dans V devient négative et inversement. Enfin, la dernière séquence, appelée inverse de $-V$ et notée $\overline{-V}$, est obtenue en prenant la séquence $-V$ dans l'ordre inverse en commençant par son dernier élément.

Une fois que les séquences primaires et secondaires ont été définies, CSMA peut être appliqué pour extraire les variantes de motifs musicaux, comme expliqué ci-dessous.

Pour détecter les motifs transposés et inversés, $-V$ est placé après V (le dernier élément de V est suivi du premier élément de $-V$), ce qui donne une séquence $< V, -V >$ de taille $2l_v$ sur lequel CSMA est appliqué. Ensuite, si CSMA génère un nouveau motif ayant deux positions $(i, len_i), (j, len_j)$ tels que *si* $i \leq l_v \wedge j \leq l_v \wedge len_i = len_j \wedge$ *il n'y a pas d'occurrence de motif identique dans ces positions*, alors nous avons une *forme transposée* aux positions i et j. Par exemple, comme on peut le voir à la figure 2.6, en considérant la séquence (c) suivie par la séquence (d), le motif de valeur <1 0 2> a trois positions qui correspondent à 1, 5 et 9. Comme les positions 1 et 9 correspondent aux motifs identiques m_1 et m_3, il reste le motif à la position 5 qui correspond à m_2. Nous concluons donc que m_2 est un motif transposé de m_1 et m_3.

Quand CSMA extrait un motif ayant deux positions i et j, tel que si $i \leq l_v, j > l_v$ et $S_{i-1} = S_{j-l_v}$, la sous-séquence $S_1 =< S_{j-l_v}, ..., S_{j-l_v+len_j} >$ est une *forme inversée* de la sous-

séquence $S_2 =< S_{i-1}, ..., S_{i-1+len_i} >$. Par exemple, lorsque nous concaténons les séquences (c) et (d) de la figure 2.6, le motif de valeur <4 -6 7> est présent à la position 13, ce qui est inférieur à 23 (l_v) et à la position 40 qui est supérieure à 23. De plus, $S_{i-1} = S_{13-1} = 67$ et $S_{j-l_v} = S_{40-23} = S_{17} = 67$. Par conséquence, le motif m_5, qui commence à la position 17, est une forme inversée du motif m_4, qui commence à la position 13.

Pour détecter la forme miroir, $\overline{-V}$ est placé après V de sorte que le dernier élément de V soit suivi du premier élément de $\overline{-V}$. Cela crée à nouveau une nouvelle séquence $< V, \overline{-V} >$ de taille $2l_v$. Si CSMA trouve un motif $m = (X, freq(X), P)$ avec $(i, len_i), (j, len_j) \in P$ tel que $(i \leq l_v) \wedge (j > l_v) \wedge (V_{i+len_i/2} = 0) \wedge (len_i = len_j)$ puis la sous-séquence de la position i à $i + len_i$ dans la séquence mélodique S est un *miroir*.

Par exemple, dans la figure 2.6 (c) (e), le motif de valeur <-4 6 -7 0 7 -6 4> apparaît à deux positions $i = 17$ et $j = 24$ dans $< V, \overline{-V} >$, la première étant inférieur à l_v et la deuxième est supérieure à l_v. Comme la variation de valeur au milieu est égale à 0, le motif <67 63 69 62 62 69 63 67> de S à partir de l'index $i = 17$ à l'index $i + len_i = 24$, comprenant m_5 et m_6, est un motif en forme miroir.

2.4 Évaluation du processus d'extraction de motifs dans le cas de séquences synthétiques

Afin de générer des données pouvant être interprétées comme des données musicales, nous considérons qu'une note est un item ou itemset dans le cas des accords, une partition est une séquence. La stratégie que nous avons utilisée pour contrôler les motifs appartenant à la séquence consiste à générer un jeu de données sans motif, puis à construire des motifs et à les introduire dans le jeu de données. Le générateur SPMF a été utilisé pour créer des séquences avec un grand vocabulaire (nombre maximum d'éléments distincts) et de petits ensembles d'items (nombre d'items par élément) avec les paramètres suivants. Le nombre de séquences, le nombre maximal d'éléments distincts, le nombre d'éléments par jeu et le nombre d'éléments par séquence ont été, respectivement, fixés à 500, 1000, 1 et 10.

Ainsi, avec ces paramètres, dans l'ensemble de données généré, appelé *D500I1KT10*, la

probabilité d'avoir des motifs est très faible, estimée à 5×10^{-5}.

Ensuite, un ensemble de motifs contigus a été généré, ainsi qu'un ensemble de *gaps* insérés dans les motifs pour former des motifs non contigus. Les deux types de motifs ont été insérés indépendamment dans D500I1KT10, menant à deux bases de données contenant chacune dix jeux de données : les jeux de données dans Database1 contiennent des séquences avec des motifs contigus, tandis que ceux de Database2 contiennent des séquences avec des motifs non contigus.

Ces ensembles de données permettent d'évaluer CSMA2, la version de notre algorithme capable d'extraire les motifs dans un ensemble de chaînes. Pour cette raison, CSMA2 a été comparé à CM-SPADE, proposé par Philippe Fournier-Viger [Fournier-Viger et al., 2014]. Comme prévu, CSMA2 et CM-SPADE ont trouvé les mêmes motifs avec exactement les mêmes fréquences. La première conclusion est que CSMA est capable d'extraire des motifs contigus et non contigus à partir de bases de données composées de différentes séquences, mais l'un de ses avantages est de fournir les positions des motifs détectés.

La deuxième partie de l'évaluation concerne CSMA1, la version de notre algorithme qui recherche des motifs à partir d'une seule chaîne.

Pour cette expérience, chaque jeu de données a été transformé en une chaîne en concaténant toutes les chaînes. Ainsi, nous obtenons deux autres bases construites à partir de Database1 et de Database2, appelées respectivement chaîne1, avec des motifs contigus, et chaîne2, avec des motifs non contigus. Les résultats sont évalués en fonction du nombre de motifs distincts identifiés et de la fréquence (nombre d'occurrences correspondant à ces motifs). Les valeurs moyennes μ et les écarts types σ calculés sur les 10 jeux de données de chaque base de données sont rapportés comme résultats finaux. CSMA1 est comparé à l'algorithme de Liu. Pour CSMA, *minFreq*, *minLength*, *maxLength* ont été réglés respectivement sur 2, 2, 200 avec *maxGap* égal à 0 pour chaîne1 et à 3 pour chaîne2 avec *nbrGaps* fixé à 1.

Les résultats sont présentés dans le tableau 2.4.

En ce qui concerne les ensembles de données contenant une séquence sans *gaps* (chaîne1), les résultats montrent que les algorithmes CSMA1 et Liu trouvent approximativement le même nombre de motifs distincts ayant presque les mêmes fréquences. La différence vient

TABLE 2.4 – Résultat de la fouille de séquences (une séquence à la fois)

Algorithmes	chaîne1 ($\mu(\sigma)$)		chaîne2 ($\mu(\sigma)$)	
	Nombre de motifs distincts	fréquences	Nombre de motifs distincts	fréquences
CSMA1	1100.6 (41.45)	7893.2(1210.14)	6964.8 (2669.22)	25514.8 (10035,26)
Liu[Liu et al., 1999]	1003.4(38.94)	7088.4 (869.72)	1073.1 (21)	6891.4 (914)

du fait que l'algorithme de Liu identifie uniquement des motifs non triviaux (*i.e.* motifs qui n'ont pas de sous-motifs avec la même fréquence) alors que CSMA1 détecte tous les motifs vérifiant les contraintes. Pour les jeux de données avec des *gaps* (chaîne2), nous observons les mêmes phénomènes mais la différence est plus importante. Cela s'explique par le fait que les motifs longs sont cassés en motifs plus petits, en raison des *gaps*, mais ils restent fréquents. Cependant, puisque CSMA1 autorise des *gaps* de 3, le nombre de motifs détectés par CSMA est supérieur à celui de l'algorithme Liu. Ainsi, CSMA est capable d'extraire des motifs contigus et non contigus d'une chaîne unique et il peut prendre en compte les *gaps*, ce qui n'est pas le cas de l'algorithme de Liu.

En conclusion, CSMA peut trouver, avec ses deux versions, des motifs contigus et non contigus à partir d'une seule chaîne ou d'un ensemble de chaînes.

2.5 Évaluation du processus d'extraction de motifs dans le cas des séquences musicales

Le but de cette section est de montrer et de discuter le rôle de l'algorithme de fouille de séquences dans sa capacité à extraire des motifs corrects dans le cas de séquences construites à partir de la sortie du RPN (Faster R-CNN) sans post-traitement ni correction des erreurs de classification en introduisant uniquement les gaps dans les séquences traitées comme illustré dans la figure 2.7.

Pour étudier l'utilité des gaps pour l'extraction des motifs en fonction des taux d'erreur de reconnaissance des primitives, nous avons considéré l'ensemble de données MUSCIMA ++ et nous avons construit des transcriptions pour chaque image de partition de musique.

Nous avons opté pour deux stratégies pour évaluer la qualité du processus d'extraction de motifs : la première consiste à analyser les séquences MIDI de la vérité terrain initialement

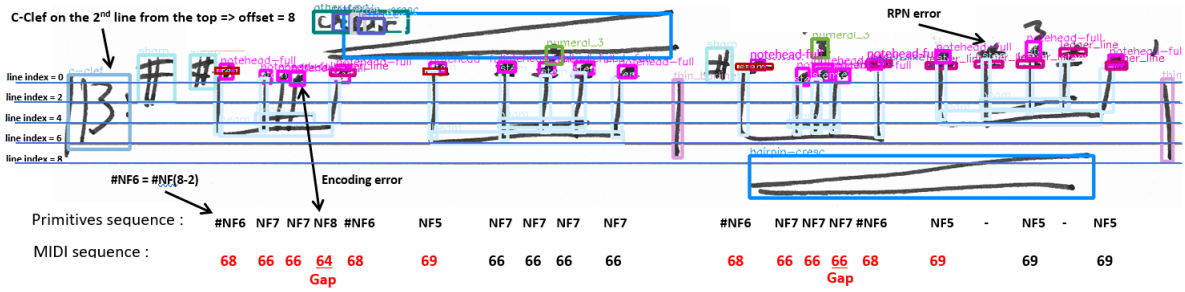


FIGURE 2.7 – Intérêt du gap dans le contexte de recherche de motifs dans des séquences contenant des erreurs du RPN

altérées d'une certaine quantité d'erreurs aléatoires (suppression ou substitution de caractères aléatoires dans les séquences) et d'en extraire les motifs selon des valeurs différentes de gaps. Cette évaluation est présentée dans la section 2.5.1. La deuxième approche est utilisée comme test de la séquence produite par le processus d'encodage (génération de séquence) après la classification par RPN (incluant donc les erreurs réelles de reconnaissance). À partir de la séquence résultante, nous appliquons l'algorithme d'extraction de motifs avec et sans utilisation de gaps afin de montrer l'amélioration substantielle de détection de motifs sans recourir à des traitements lourds de post-correction d'erreurs de reconnaissance (et d'encodage). Cette partie expérimentale est présentée dans la section 2.5.2.

2.5.1 Évaluation du processus de fouille de données avec gaps sur les séquences contenant des erreurs contrôlées

L'ensemble des partitions transcrites de la vérité terrain MUSCIMA++ est noté XML-GT. Pour cette première expérience, nous introduisons aléatoirement des erreurs dans les séquences associées de ces transcriptions avec des taux d'erreur croissants variables (allant de 0% à 50%) représentés dans l'axe X du graphique de la figure 2.8.

Cette expérience est réalisée avec deux configurations :

— Gap size = 1 et nombre de gaps autorisés par occurrence de motif (#gaps) = 1

— Gap size = 1 et nombre de gaps autorisés par occurrence de motif (#gaps) = 2

L'évaluation de l'extraction de motifs est basée sur l'estimation du nombre de motifs communs entre les séquences modifiées et les données XML-GT de vérité de terrain.

On peut noter qu'avec un taux d'erreur de 0, nous devrions trouver les mêmes motifs

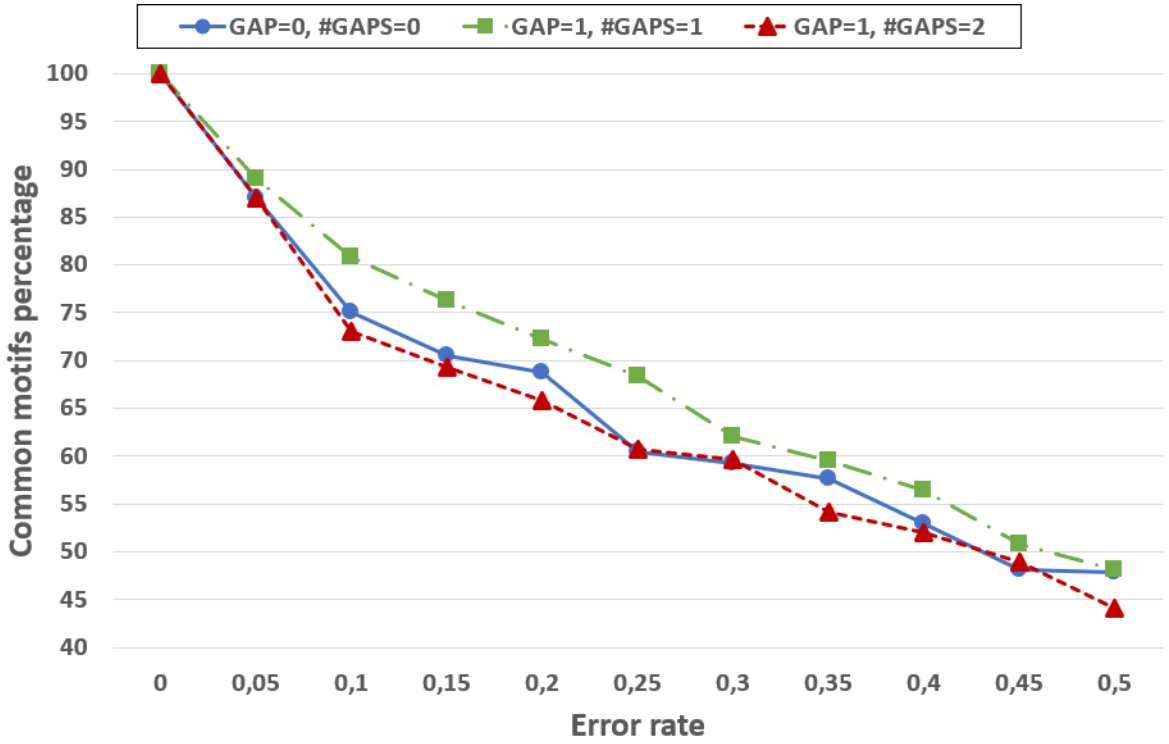


FIGURE 2.8 – Pourcentage de motifs communs en fonction du taux d'erreur simulé

puisqu'aucune altération n'a été introduite et que le pourcentage de motifs communs devrait diminuer lorsque le nombre d'erreurs introduites dans les séquences modifiées augmente.

Pour montrer les avantages de l'utilisation des gaps, nous avons d'abord calculé les pourcentages de motifs communs entre les séquences modifiées et les séquences XML-GT en utilisant une taille de gap égale à 0, puis nous avons modifié la taille et le nombre des gaps et calculé les performances pour chaque configuration.

Le pourcentage de motifs XML communs (dénotés CM-Xml, voir figure 2.9) est estimé comme l'intersection entre les motifs corrects Xml (considérés comme référence, et dénotés XML-GT(seqID, E=0)) et les séquences XML originales (avec seqID$\in \{1, 2, ..., n\}$) avec l'ensemble des motifs extraits de la séquence modifiée avec une erreur croissante (dénotés e_j) dans les différents seqID$\in \{1, 2, ..., n\}$. En pratique on aura $n = 9$ car 9 partitions de tests. CM-Xml est alors défini par l'équation (2.1).

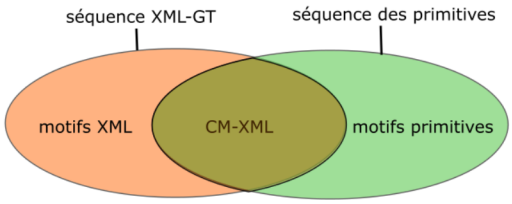


FIGURE 2.9 – Évaluation de la performance des motifs communs

$$\left[\text{CM-Xml}(seqID_i)_{i\in\{1,..,n\}}\right]_j = \frac{|\text{XML-GT(seqID,E=0)} \cap \text{ XML-GT(seqID, E=}e_j\text{)}| \times 100}{|\text{XML-GT(seqID, E=0)}|}$$

$$, j \in \{0, .., 0.5\} \text{ avec un pas de } 0.05 \tag{2.1}$$

L'erreur cumulée moyenne pour tout taux e_j, notée $ACM.Xml_j$ est définie par (2.2).

$$ACM.Xml_j = \frac{\sum_{i=1}^{n} \text{CM-Xml}(seqID_i)}{\#Scores} \tag{2.2}$$

Avec #Scores, le nombre de partitions utilisées dans la vérité terrain (dans ces expériences #Scores = n partitions (n=9).

Le résultat de ces différentes estimations est illustré dans la Fig. 2.8 qui permet de comparer nos deux configurations avec gap (courbe rouge et verte) et sans gap (courbe en bleue).

Dans la Fig. 2.8, lorsque le taux d'erreur est fixé à 0, nous avons logiquement un 100% de motifs communs ($ACM.XML_0 = 100$). Puis, plus le taux d'erreur augmente, plus les pourcentages diminuent avec des tendances différentes selon les configurations. Nous remarquons que l'utilisation d'un gap de 1 (en vert) réduit l'impact des erreurs introduites comme on peut le voir dans nos expériences, la courbe correspondante est supérieure à celle de $maxGap$=0 (en bleu). Pour des erreurs comprises entre 0,1 et 0,25 on observe une amélioration moyenne d'environ 10% sur les partitions de tests. Par exemple, lorsque le taux d'erreur est d'environ 0,2, CSMA est capable de trouver 68% des motifs XML communs lorsque $maxGap$ =0 alors qu'il peut trouver environ 73% lorsque $maxGap$ = 1 et #GAP=1 produisant dans ce cas une augmentation de performances de 5%. On peut enfin remarquer (sur la courbe rouge), que le fait d'autoriser plus d'un gap à l'intérieur des motifs peut, à l'inverse, diminuer les

TABLE 2.5 – Évaluation des performances des données réelles : ID page : ID de page MUSCIMA ; mAP : Précision moyenne du Faster R-CNN ; ALD : distance moyenne de Levenshtein entre la séquence de référence XML-GT et la séquence de sortie Faster R-CNN.

ID page	**mAP**	**ALD**	**ACM.Xml**	
			maxGap=0. #gaps = 0	**maxGap=1, #gaps = 1**
1	0,87	0,25	63,2	68,6
2	0,67	0,41	**49,2**	**54,8**
4	0,79	0,36	54,6	61,2
6	0,77	0,34	55,5	61,3
7	0,93	0,11	75,3	81,5
9	0,76	0,32	60,8	66,2
11	0,83	0,22	71,3	76,6
13	0,87	0,23	64,7	70,5
15	0,69	0,47	**44,8**	**50,6**
Moyenne			60	65.7

performances. Ceci est dû au fait qu'avec de nombreux gaps nous pouvons former plus de motifs et qu'un même item peut être présent dans plusieurs motifs. Ainsi, un item erroné peut avoir un impact sur plus d'un motif.

2.5.2 Expérience sur les séquences réelles de sortie Faster R-CNN (RPN)

Nous avons vu dans la sous-section précédente l'intérêt de l'utilisation des gaps dans le processus d'extraction des motifs. Maintenant, nous montrons comment les gaps améliorent les performances sur les données réelles. Pour cette expérience, nous avons utilisé le réseau neuronal convolutionnel Faster Region-proposal Convolutional Neural Network décrit dans la section 1.4.4.

Pour faire le lien avec l'expérience précédente, nous avons mis dans le tableau 2.5 les résultats obtenus en utilisant les mêmes configurations sur les pages MUSCIMA étudiées en fonction des performances du modèles RPN (données dans la colonne mAP) et les distances Levenshtein moyennes par page qui sont données dans la colonne ALD.

Dans les colonnes 4 et 5 du tableau 2.5, nous calculons le nombre moyen de motifs communs entre les séquences RPN et la séquence XML-GT associée (vérité de terrain) respectivement sans gaps et avec gaps (maxGap et #gaps fixés à 1).

On observe le même phénomène que dans le cas de séquences altérées aléatoirement. Par exemple, pour une distance moyenne de Levenshtein de 0,41 dans le cas de la page 2, ce qui équivaut à un taux d'erreur de 0,41, où la précision moyenne est de 0,67 (cf. tableau 2.5) l'utilisation du gap permet de trouver jusqu'à 54.8% de motifs communs plutôt que 49.2% avec $maxGap = 0$. La même observation vaut pour les autres pages avec une augmentation moyenne de 5,7% de détection correcte avec séquence obtenue après reconnaissance RPN.

Les taux initiaux de bonne détection reposant sur le Faster R-CNN ont affiché des taux avoisinant les 90%. Si les résultats de détection de motifs sont d'environ 60%, cela provient du fait que le réseau RPN n'a pas été en mesure de trouver certaines occurrences de primitives. Les erreurs les plus fréquentes concernent les lignes supplémentaires, celles qui ne sont pas situées sur les lignes de portées mais dans les marges et l'impact d'un défaut de reconnaissance à ce niveau est important au niveau de l'encodage des notes qui leurs sont associées.

Les meilleurs scores de détection et de reconnaissance du Faster R-CNN (score moyen de 93% dans le papier de Pacha [Pacha et al., 2018a]) sont influencés par les classes les moins fréquentes comme les clefs, les souffles, les chiffres, ou les classes aux caractéristiques morphologiques simples comme les hampes et les barres de mesure. Au sein du sous-ensemble que nous avons considéré dans notre étude, le RPN a rencontré des difficultés à détecter certaines primitives comme les lignes supplémentaires (marginales), les altérations et les têtes de notes. Cela explique que sur les partitions que nous avons choisies de traiter (partitions mono-voix présentant des notes situées en dehors du cadre des lignes de portées), nous obtenons un score de détection de motifs qui ne peut excéder les 82% dans le cas d'autorisation d'un seul gap.

2.5.2.1 Conclusion

Dans cette partie, nous avons présenté un nouvel algorithme d'extraction de motifs, appelé CSMA, capable de trouver des motifs contigus et non contigus. Cet algorithme incorpore des contraintes liées à la fréquence, à la taille et le nombre de gaps et à la longueur du motif. Avec ses deux versions, CSMA peut trouver des motifs d'une ou plusieurs chaînes. CSMA a été employé ici pour permettre l'extraction de motifs fréquents dans les séquences encodées

des partitions de la base MUSCIMA++. Pour l'heure, la base de motifs produite, n'a encore subi aucun filtrage spécifique pour répondre au cahier des charges de notre musicologue. Cependant, un logiciel de validation et de sélection sous contrainte a été implémenté pour faciliter le travail de l'expert musicologue dans la sélection des motifs principaux (filtrage selon la taille du motif, filtrage selon l'emplacement du motif dans la partition (début – milieu- fin). Il s'agit là d'une partie d'exploitation de notre système de fouille qui sera réalisé au fil de l'eau par l'experte musicologue du projet. Ce que nous avons montré à travers les expérimentations du chapitre, c'est la capacité du système à contourner les erreurs de transcriptions et d'encodage lorsque celles-ci sont ponctuelles (gap à 1 ou gap à 2) et à retrouver indépendamment des erreurs ponctuelles produites lors des étapes de préparation de la séquence une base de motifs de très bonne qualité (atteignant plus de 5 % de bonne détection au final).

Le chapitre suivant présente des cas d'usages de la base de motifs produite pour l'identification et la classification de compositeurs.

Chapitre 3

Exploration de CSMA pour l'identification de compositeurs

3.1 Introduction

Dans cette section, nous nous proposons d'évaluer l'intérêt des motifs détectés à travers deux tâches, d'une part l'identification de compositeurs et d'autre part le classement supervisé de partitions. Pour ces deux tâches, nous supposons que les motifs musicaux permettent de caractériser les œuvres musicales. Dans les deux cas les motifs seront donc utilisés pour représenter les partitions mais dans le cas d'identification de compositeur il s'agit de décider si une partition a été écrite ou non par un compositeur donné alors que dans le cas du classement supervisé, il s'agit d'associer une partition donnée à un des auteurs possibles parmi plusieurs.

Dans ce qui suit, nous allons commencer par présenter, en détail dans la section 3.2, le système d'identification d'auteurs. Ensuite, du fait que le pré-traitement est commun, nous allons présenter dans la section 3.3, avec moins de détail, le système de classification d'auteurs.

3.2 Identification d'auteurs

Le problème d'identification d'auteur peut être défini de la façon suivante : étant donné un corpus composé de documents, on dispose pour chaque problème p d'un ou plusieurs documents A_p du corpus qui ont été rédigés par un même auteur et d'un document p_u dont l'auteur est inconnu. L'objectif est de déterminer si p_u a été écrit ou non par le même auteur que les documents de A_p.

Pour mettre en place un système d'identification d'auteurs le corpus des documents est réparti en deux ensembles ; un ensemble 1 et un ensemble 2.

Le processus d'identification d'auteurs consiste en deux étapes principales ; la construction de l'index à partir de l'ensemble 1 et l'attribution des identités aux œuvres inconnues de l'ensemble 2. L'index doit contenir le plus d'informations discriminantes permettant de caractériser chaque auteur. La décision d'identification est basée sur le calcul de similarité entre une œuvre inconnue et les œuvres connues de l'auteur présumé.

Dans le cadre de ce travail, on part de l'hypothèse que les variantes des motifs musicaux représentent une signature caractérisant les œuvres musicales. Dans ce chapitre nous présentons les résultats d'expériences effectuées sur un système d'identification d'auteurs (compositeurs) en fonction de la nature des motifs considérés.

3.2.1 Indexation des données

Dans ce travail on cherche à utiliser les motifs musicaux comme caractéristiques. L'ensemble des motifs noté $\mathcal{M} = \{m_1, m_2, \ldots, m_j \ldots, m_M\}$ correspond à tous les motifs trouvés dans toutes les œuvres des auteurs dans l'ensemble 1 et constitue l'index . L'ensemble des auteurs présents dans la base est noté $A = \{a_1, a_2, \ldots, a_i, \ldots, a_N\}$. L'ensemble des partitions associées à un auteur a_i est noté $P_i = \{p_i^1, p_i^2, \ldots, p_i^k, \ldots, p_i^K\}$. Ainsi, l'ensemble de toutes les partitions est noté $P = \cup_i P_i$. La partition p_i^k de l'auteur a_i est représentée par un vecteur dont chaque composante w_{ij}^k correspond au poids du motif $m_j \in \mathcal{M}$ dans cette partition.

Le poids w_{ij}^k est défini suivant le modèle *TF-IDF*. Ce poids est calculé suivant la formule (3.1) :

$$w_{ij}^{k} = freq(m_{ij}^{k}) \times ln \frac{\sum_{i=1}^{N} |P_i|}{|\{p_{i'}^{k'} \in P : freq(m_{i'j}^{k'}) > 0\}|} \tag{3.1}$$

Où $freq(m_{ij}^{k})$ correspond à la fréquence du motif j dans la partition k de l'auteur i.

Pour le calcul des similarités entre deux œuvres musicales $p_{i_1}^{k_1}$ et $p_{i_2}^{k_2}$, on utilise la formule du cosinus qui est définie dans l'équation (3.2).

$$\cos(p_{i_1}^{k_1}, p_{i_2}^{k_2}) = \frac{\sum_{j=1}^{M} w_{i_1 j}^{k_1} w_{i_2 j}^{k_2}}{\sqrt{\sum_{j=1}^{M} (w_{i_1 j}^{k_1})^2 \times \sum_{j=1}^{M} (w_{i_2 j}^{k_2})^2}} \tag{3.2}$$

3.2.2 Processus d'identification d'auteurs

Différentes approches peuvent utilisées pour traiter le problème d'identification d'auteur. Dans ce manuscrit nous allons analyser les performances de l'algorithme du comptage de dissimilarité (**DCM**) car cette méthode a obtenu de bons résultats dans le contexte des données textuelles lors de la compétition Clef 2015 [Frery et al., 2015].

3.2.2.1 Dissimilarity Counter Method

Étant donné un problème d'identification d'auteur $p = (p_u, a_i)$, Cette méthode consiste à assigner le document p_u au même auteur que les documents de P_i (l'ensemble des partitions de l'auteur $a_i \in A$) si la plupart d'entre eux sont plus proches de p_u qu'ils ne le sont des autres documents de P_i. Plus précisément la plus petite similarité de chaque document $p_i^k \in P_i$ aux autres documents de $P_i - \{p_i^k\}$ est calculée puis comparée à la similarité de p_i^k à p_u. Si la première est supérieure à la seconde, un compteur est incrémenté.

Formellement, la valeur *count* associée à l'auteur i et la partition p_u est définie par l'équation (3.3)

$$count(p_u)_i = |\{p_i^k \in P_i : min\{\cos(p_i^k, p_i^{k'}), k \neq k'\} > \cos(p_i^k, p_u)\}| \tag{3.3}$$

Ensuite, afin d'attribuer l'auteur de l'œuvre inconnue, un seuil σ doit être fixé tel que si le count dépasse ce seuil la partition est considérée comme trop éloignée par rapport à l'auteur i, sinon la partition fait partie des œuvres de cet auteur.

Le calcul de la valeur du *count* et la décision sur l'appartenance d'une partition p_u à l'auteur i sont réalisés suivant l'algorithme 3 :

Algorithm 3: $Count(p^u)_i$

Input : $\mathcal{P}_i$: Set of music scores features of author a_i, p_u : Unknown music score author, δ : Count threshold
Output: $True$ if $p_u \in \mathcal{P}_i$, $False$ otherwise

```
begin
    count = 0
    for p_i^k ∈ P_i do
        min = 1
        for p_i^k' ∈ P_i − {p_i^k} do
            sim = cos(p_i^k, p_i^k')
            if min > sim then
                min = sim
            end
        end
        if cos(p_i^k, p_u) < min then
            count = count + 1
        end
    end
    if count > δ then
        return False
    else
        return True
    end
end
```

3.2.2.2 Exemple :

On considère trois auteurs avec trois partitions. On considère les motifs avec une fréquence minimale de 2. Chaque motif est associé à une couleur. La répartition des motifs est présentée dans la Table 3.1.

TABLE 3.1 – Exemple de séquences contenant des motifs

Auteurs	Ensemble	Partitions	Séquences
P_1	Train	p_1^1	60-63-59-72-63-60-60-57-68-67-66-70-75-65-63-59-72-63-60-74-63-54-57-68-67-63-59-72-63-60
		p_1^2	63-59-72-63-60-75-74-73-70-69-63-75-74-73-70-57-68-67-63-59-72-63-60-74-73-70
	Ensemble 2	p_1^3	63-59-72-63-60-75-74-73-70-69-63-75-74-73-70-57-68-67-63-59-72-63-60
P_2	Train	p_2^1	57-68-67-69-75-78-73-69-66-69-57-68-67-68-65-61-57-68-67-63-59-59-57-66-63-57-68-67
		p_2^2	63-59-72-63-60-72-77-57-68-67-61-65-64-57-68-67-65-69-63-59-72-63-60
	Ensemble 2	p_2^3	57-68-67-73-69-66-69-69-63-75-57-68-67-69-75-78-73-71-70-68-63-60-62
P_3	Train	p_3^1	66-68-67-69-75-78-73-69-66-69-57-69-75-78-73-69-66-63-68-67-69-75-78-73-69-66-58-68-67
		p_3^2	69-75-78-73-69-66-77-57-68-67-61-65-68-70-75-72-60-53-64-57-68-67-65-69-75-78-73-69-66-68-70-75-72-60-53
	Ensemble 2	p_3^3	68-70-75-72-60-53-66-69-69-68-70-75-72-60-53-78-73-71-70-68-63-60-62

Pour plus de lisibilité on renomme les motifs comme suit :

— motif 1 = 63-59-72-63-60

— motif 2 = 57-68-67

— motif 3 = 74-73-70

— motif 4 = 69-75-78-73-69-66

— motif 5 = 68-70-75-72-60-53

Pour la phase de construction de l'index on prend en compte un sous-ensemble de partitions composé de $\{p_1^1, p_1^2, p_2^1, p_2^2, p_3^1, p_3^2\}$ tandis que $\{p_1^3, p_2^3, p_3^3\}$ est utilisé pour la validation.

Les motifs générés ont une fréquence minimale de 2. Le tableau suivant contient la fréquence de chaque motif pour chaque auteur et pour chaque partition :

TABLE 3.2 – Index de la base de données - TF

Auteur		**Partition**	**Motif 1**	**Motif 2**	**Motif 3**	**Motif 4**	**Motif 5**
P_1	**ensemble 1**	p_1^1	3	2	0	0	0
		p_1^2	2	0	3	0	0
	Ensemble 2	p_1^3	2	0	2	0	0
P_2	**ensemble 1**	p_2^1	0	4	0	0	0
		p_2^2	2	2	0	0	0
	Ensemble 2	p_2^3	0	2	0	0	0
P_3	**ensemble 1**	p_3^1	0	0	0	3	0
		p_3^2	0	2	0	2	2
	Ensemble 2	p_3^3	0	0	0	0	2

La table 3.3 contient le nombre d'occurrences de chaque motif, noté DF, par rapport à chaque partition.

Le poids de chaque motif w_{ij}^k, calculé suivant l'équation (3.1)., est donné dans la table 3.4 tel que les lignes grisées correspondent aux données de l'ensemble 2.

Dans la Table 3.5, les cases avec fond vert correspondent aux similarités valides (c.à.d supérieur au minimum des similarités associées à la partition en ligne), et les rouges correspondent aux similarités invalides. Le *count* correspond ici au comptage des cases rouges par auteur. L'association d'une œuvre à un auteur donné se base sur la valeur maximale du

TABLE 3.3 – Nombre de partitions de l'ensemble 1 contenant chaque motif

Motifs	Motif 1	Motif 2	Motif 3	Motif 4	Motif 5
DF	3	4	1	2	1

TABLE 3.4 – Motifs *TF-IDF*

Auteur	Partition	Motif 1	Motif 2	Motif 3	Motif 4	Motif 5
P_1	p_1^1	0.903	0.352	0	0	0
	p_1^2	0.602	0	2.334	0	0
	p_1^3	0.602	0	1.556	0	0
P_2	p_2^1	0	0.704	0	0	0
	p_2^2	0.602	0.352	0	0	0
	p_2^3	0	0.352	0	0	0
P_3	p_3^1	0	0	0	1.431	0
	p_3^2	0	0.352	0	0.954	1.556
	p_3^3	0	0	0	0	1.556

TABLE 3.5 – valeurs du *count* normalisé entre les partitions

	p_1^1	p_1^2	p_2^1	p_2^2	p_3^1	p_3^2	p_1^3	p_2^3	p_3^3
p_1^1	1	0,232					0,336	0,363	0
p_1^2	0,232	1					0,993	0	0
p_2^1			1	0,504			0	1	0
p_2^2			0,504	1			0,311	0,504	0
p_3^1					1	0,513	0	0	0
p_3^2					0,513	1	0	0,189	0,837

count.

Ce comptage simple est suffisant dans le cas d'une base de données équilibrée. Par contre, dans le cas d'une base non équilibrée on envisage un comptage normalisé pour chaque auteur ce qui revient à diviser le *count* associé à un auteur par le nombre total des œuvres de cet auteur.

De cet exemple, on remarque que pour un seuil de 30%, les partitions p_1^3 et p_2^3 sont bien identifiées comme œuvres de l'auteur 1 et l'auteur 2 respectivement. Tandis que la partition p_3^3 n'est pas identifiée comme une œuvre de l'auteur 3.

3.2.3 Protocole expérimental

3.2.3.1 Base de données

Dans la littérature on trouve très peu de bases de données musicales en format symbolique. Certains sites web donnent accès à des œuvres transcrites. Une des références les plus célèbres est MuseScore.com qui propose un très grand nombre de partitions au format XML. Cependant, les transcriptions proposées dépendent des éditeurs qui ne sont pas forcément

des professionnels, et donc ces transcriptions ne correspondent pas nécessairement à l'œuvre originale. En d'autre lieu, on trouve la base NEUMA qui propose un ensemble de partitions transcrites au format XML [1]. Cette base met à disposition les œuvres de dix auteurs (compositeurs) avec un nombre variable de partitions par compositeur. Nous avons pris la décision de travailler avec cette base pour faire de l'identification d'auteurs.

Cependant, les données de cette base ne sont pas équilibrées par rapport aux œuvres par auteur. Ainsi, par exemple nous avons 402 partitions pour Bach mais seulement une pour Mozart. De ce fait, nous proposons la répartition des données en un ensemble 1 et en un ensemble 2 de deux façons : répartition équilibrée et répartition non équilibrée.

De plus, comme l'ensemble 2 contient des partitions écrites par des compositeurs (Berlioz, Brumel, Cherubini, Mozart) qui n'apparaissent pas dans l'ensemble 1 et qui ne sont pas utilisés pour la construction de l'index, nous résolvons la tâche d'identification du compositeur dans deux contextes. Dans le premier paramétrage, nous avons considéré 114 problèmes positifs, pour lesquels la solution attendue est le même compositeur, et 456 problèmes négatifs (c'est-à-dire différents compositeurs) concernant les partitions écrites par les 5 premiers compositeurs comme indiqué dans le tableau 3.6. Dans le second paramétrage, nous ajoutons des problèmes négatifs concernant aussi les autres compositeurs qui n'ont pas servi à la recherche des motifs de l'index (Berlioz, Brumel, Cherubini, et Mozart) comme indiqué dans la table 3.6, soit dans ce cas un total de 114 problèmes positifs et 476 problèmes négatifs contre 456 problèmes négatifs dans le premier paramétrage.

La répartition non-équilibrée des données est détaillée dans la table 3.6. La répartition équilibrée est la même que celle proposée dans la table 3.6 avec un nombre de partions réduit à 20 pour Bach dans l'ensemble 1 et un nombre de partitions égale à 382 pour le même auteur dans l'ensemble 2, comme indiqué entre parenthèses dans la table 3.6. De ce fait, le nombre total d'exemples positifs dans cette configuration est 394 et le nombre total d'exemples négatifs dans le premier paramétrage est 1576 contre 1596 dans le second paramétrage.

1. `http://neuma.huma-num.fr/home/corpus/composers/`

TABLE 3.6 – Distribution des données non-équilibrée

Auteur	**Ensemble 1**	**Ensemble 2**	**Problèmes positifs**	**Problèmes négatifs**	
				Paramétrage 1	Paramétrage 2
Bach	300 *(20)*	102 *(382)*	102 *(382)*	12 *(12)*	16 *(16)*
Corelli	2 *(2)*	2 *(2)*	2 *(2)*	112 *(392)*	116 *(396)*
Haydn	4 *(4)*	4 *(4)*	4 *(4)*	110 *(390)*	114 *(394)*
Monteverdi	6 *(6)*	4 *(4)*	4 *(4)*	110 *(390)*	114 *(394)*
Palestrina	4 *(4)*	2 *(2)*	2 *(2)*	112 *(392)*	116 *(396)*
Berlioz	0	1			
Brumel	0	1			
Cherubini	0	1			
Mozart	0	1			

3.2.3.2 Indexation

Nous avons généré les motifs musicaux des partitions musicales de l'ensemble 1 dans le cas des données équilibrées et non-équilibrées. Pour couvrir le maximum des motifs communs entre les partitions nous avons choisi d'utiliser CSMA avec des valeurs de contraintes minimales par rapport à la fréquence et la taille minimale des motifs qui sont fixées à 2. Aussi, nous avons considéré les motifs triviaux et non triviaux. Par contre, les gaps n'ont pas été autorisés.

La répartition des motifs par auteur, dans le cas des données non-équilibrées, est donnée dans la table 3.7 :

— IM : les motifs identiques mélodiques

— IR : les motifs identiques rythmiques

— IB : les motifs identiques à la fois mélodiques et rythmiques

— Variations : les variantes transposés, inversés et miroirs des motifs

— $|m \in A_i|$: correspond au nombre de motifs présents dans les œuvres de l'auteur A_i

— mPr : correspond au pourcentage des motifs propres à l'auteur A_i. Les motifs propres sont les motifs qui n'apparaissent que dans les œuvres de l'auteur A_i.

On remarque qu'on trouve plus de motifs identiques mélodiques et/ou rythmiques que les motifs variantes. Le nombre de motifs trouvés dans les partitions de Bach est supérieur par rapport aux autres car on a plus de partitions de Bach, dans le cas des données non équilibrées.

On constate aussi qu'on trouve plus de variantes de nature transposée et beaucoup moins

TABLE 3.7 – Nombre de motifs discriminants par auteur et le pourcentage des motifs propres (mPr : Pourcentage des motifs propres, IM : Motifs identiques mélodiques, IR : Motifs identiques rythmiques, IB : Motifs identiques mélodiques et rythmiques)

Auteur (A_i)	IM		IR		IB		Variations					
							transposés		inversés		miroirs	
	$\|m \in A_i\|$	mPr	$\|m \in A_i\|$	mPr	$\|m \in A_i\|$	mPr	$\|m \in A_i\|$	mPr	$\|m \in A_i\|$	mPr	$\|m \in A_i\|$	mPr
Bach	223348	98%	232993	97%	217619	99%	1962	74%	253	76%	205	82%
Corelli	425	40%	1088	99%	335	100%	97	37%	16	25%	1	0%
Haydn	7957	80%	30045	89%	7489	89%	780	70%	74	50%	58	50%
Monteverdi	4034	52%	8789	59%	3472	69%	519	45%	39	18%	35	28%
Palestrina	1883	50%	7988	79%	1422	68%	437	56%	28	1%	20	45%

de motifs en forme miroir. On remarque aussi que Corelli a les plus bas pourcentages de variantes de motifs propres que les autres auteurs.

Dans le but de réduire le nombre de motifs pour la tâche d'identification d'auteurs, dans le cas des données non équilibrées, nous avons choisi d'appliquer une opération de sélection des motifs discriminants sur les motifs identiques mélodiques et/ou rythmiques. Ce traitement est détaillé dans la section suivante.

3.2.3.3 Sélection de descripteurs discriminants

Le but de la sélection de motifs consiste à garder ceux qui sont fréquemment utilisés par les auteurs d'une façon indépendante. Ces motifs spécifiques correspondent à des motifs qui sont plus souvent utilisés dans les œuvres d'un auteur donné et moins utilisés par les autres auteurs.

La sélection des motifs discriminants est basée sur le calcul du χ^2. Le χ^2 est une mesure statistique permettant d'analyser le caractère indépendant d'un motif par rapport aux auteurs. Le χ^2 est calculé pour chaque motif m_j et les partitions P_i de l'auteur i suivant l'équation (3.4).

$$\chi^2(m_j, P_i) = \frac{N \times (AD - CB)^2}{(A + C)(B + D)(A + B)(C + D)} \tag{3.4}$$

Tel que :

— N : Le nombre total de partitions dans la base

— $A = |\{p_i^k \in P_i : m_j \in p_i^k\}|$, le nombre de partitions de l'auteur i contenant le motif m_j

— $B = |\{p_j^k \notin P_i : m_j \in p_j^k\}|$, le nombre de partitions des autres auteurs contenant le

TABLE 3.8 – Table de contingence

	$motif1 \in p_j^k$	$motif1 \notin p_j^k$
$p_j^k \in P_1$	A=2	C=0
$p_j^k \notin P_1$	B=1	D=3

TABLE 3.9 – Valeurs critiques de la distribution χ^2 avec un degré de liberté

p	valeur critique du χ^2
0.1	2.71
0.05	3.84
0.01	6.63
0.005	7.88
0.001	10.83

motif m_j

— $C = |\{p_i^k \in P_i : m_j \notin p_i^k\}|$, le nombre de partitions de l'auteur i ne contenant pas le motif m_j

— $D = |\{p_j^k \notin P_i : m_j \notin p_j^k\}|$, le nombre de partitions des autres auteurs ne contenant pas le motif m_j

La valeur du χ^2 permet de déterminer le poids discriminant du motif par rapport à un auteur donné. En d'autres termes, plus la valeur du χ^2 est élevée plus le motif associé est discriminant pour l'auteur en question.

Ainsi, la table de contingence associée au motif 1 par rapport à l'auteur 1 est donnée dans la Table 3.8. Donc, la valeur du $\chi^2(m_1, P_1)$ est égale à 3.

On déduit le degré de liberté de la table de contingence qui est égale à 1 (i.e. (nombre de colonnes -1) $\times$ (nombre de lignes -1)). Ce degré de liberté nous aide à prendre une décision sur la valeur du χ^2 sur laquelle on confirme ou on infirme l'indépendance d'un motif par rapport à un auteur donné. Pour cela, nous nous basons sur la table 3.9 qui représente une partie de la table des valeurs théoriques du χ^2 en fonction du degré de liberté. Par exemple, si les deux événements sont indépendants, alors $P(\chi^2 > 6.63) < 0.01$. Donc pour $\chi^2 > 6.63$ l'hypothèse d'indépendance peut être rejetée avec 99% de confiance [Manning et al., 2008].

Dans notre cas, nous avons choisi la valeur critique du χ^2 à 10.83. En d'autres termes les motifs choisis sont dépendants des auteurs associés avec un degré de confiance de 99.9%. Ceci implique que nous conservons tous les motifs dont la valeur du χ^2 est supérieure à 10.83. La

TABLE 3.10 – Nombre de motifs discriminants par auteur et le pourcentage des motifs propres (mPr : Pourcentage des motifs propres, IM : Motifs identiques mélodiques, IR : Motifs identiques rythmiques, IB : Motifs identiques mélodiques et rythmiques)

Auteur (A_i)	IM		IR		IB	
	$\|m \in A_i\|$	mPr	$\|m \in A_i\|$	mPr	$\|m \in A_i\|$	mPr
Bach	654	0%	1115	1%	302	0%
Corelli	323	52%	1088	99%	335	100%
Haydn	7396	86%	29023	92%	7216	93%
Monteverdi	3170	66%	7259	71%	3018	79%
Palestrina	1488	63%	7550	84%	1284	76%

table 3.10 représente les nombres des motifs mélodiques et/ou rythmiques sélectionnés par auteur.

On remarque que les motifs mélodiques discriminants de Bach sont aussi utilisés par les autres auteurs. Cependant, ils sont marqués par leurs fréquences d'apparitions dans les œuvres de Bach (i.e. le poids de ces motifs est élevé dans les œuvres de Bach). On remarque aussi que dans le cas de Corelli et Haydn, leurs motifs discriminants de nature IR (identiques rythmiques) et IB (identiques mélodiques et rythmiques) sont très peu utilisés par les autres auteurs. Mais, tandis qu'on a une grande diminution du nombre de motifs par auteurs, on remarque que dans le cas de Haydn et Monteverdi le nombre de motifs discriminants reste trop élevé par rapport aux autres.

Après la sélection des motifs discriminants des auteurs, on teste les performances du système d'identification d'auteur. L'ensemble des expériences effectuées et la méthode d'évaluation sont détaillés dans la section suivante.

3.2.4 Évaluation de l'identification de compositeurs

Nous avons évalué le système d'identification d'auteurs suivant différentes configurations selon que les données sont ou non équilibrées dans l'ensemble 1 et selon que l'on considère ou non de nouveaux compositeurs lors de l'identification (respectivement paramétrage 1 ou 2). Les résultats obtenus par configuration sont détaillés dans les sections suivantes comme indiqué dans la table 3.11.

Dans ces sections nous présentons les résultats obtenus en considérant les motifs identiques

	Paramétrage 1	Paramétrage 2
Données non équilibrées	section 3.2.4.1	section 3.2.4.2
Données équilibrées	section 3.2.4.3	section 3.2.4.4

TABLE 3.11 – Références des résultats d'évaluation

TABLE 3.12 – Mesures d'évaluation

Résultat attendu	Résultat obtenu	
	$p_u \in P_i$	$p_u \notin P_i$
$p_u \in P_i$	Vrai positif (TP)	Faut négatif (FN)
$p_u \notin P_i$	Faut positif (FP)	Vrai négatif (TN)

mélodiques et nous avons testé le système d'identification d'auteurs en considérant un seul type de motif par expérience. Nous avons déterminé le seuil du *count* d'une façon empirique par expérience. Ceci repose sur l'évaluation des performances du système en changeant les valeurs du *count*.

Enfin, dans la section 3.2.4.5, lorsque nous considérons en plus des motifs mélodiques, les motifs rythmiques et les variantes des motifs, en appliquant le vote majoritaire, nous avons combiné les sorties de chaque système conçu pour un type de motif pour obtenir une seule sortie.

Pour évaluer le processus d'identification d'auteurs, nous nous sommes basés sur les quatre indicateurs décrits dans la table 3.12.

Ces indicateurs permettent de calculer les mesures suivantes par rapport à chaque auteur :

1. Taux de bien classé : $Accuracy = \frac{TP+TN}{TP+FP+FN+TN}$
2. Précision : $Precision = \frac{TP}{TP+FP}$
3. Rappel : $Recall = \frac{TP}{TP+FN}$
4. F1-Score $= 2 \times \frac{Rappel \times Precision}{Rappel+Precision}$

Afin d'avoir une évaluation globale des performances, nous avions le choix d'utiliser deux types d'analyses : macro- et micro-évaluation [Yang et al., 1999]. La première consiste à calculer les indicateurs pour chaque auteur, puis à en faire la moyenne pour avoir un indicateur global. La seconde consiste à calculer les valeurs TP, TN, FP et FN par auteur puis à faire la somme sur l'ensemble des auteurs pour calculer ensuite les indicateurs globaux.

Cette section présente l'ensemble des résultats obtenus sur les différents jeux de données

TABLE 3.13 – Résultats du processus d'identification d'auteurs – Paramétrage 1

Author	TP	TN	FP	FN	Accuracy	Precision	Recall	F1-score
Bach	102	0	12	0	0,895	0,895	1	0,945
Corelli	2	111	1	0	0,991	0,667	1	0,8
Haydn	1	109	1	3	0,964	0,5	0,25	0,334
Monteverdi	4	99	11	0	0,903	0,267	1	0,421
Palestrina	2	100	12	0	0,894	0,142	1	0,25
					Macro-evaluation			
					0,929	0,494	0,85	0,601
					Micro-evaluation			
					0,929	0,75	0,973	0,829

(équilibrés et non équilibrés) avec les deux paramétrages (paramétrage 1 et paramétrage 2) en utilisant que les motifs mélodiques.

3.2.4.1 Expérience sans prise en compte des auteurs moins fréquents (Paramétrage 1) sur des données non équilibrées

Les performances du système d'identification d'auteurs dans le cas des données non équilibrées sont présentées dans la table 3.13.

On remarque, au niveau de la micro-évaluation et de la macro-évaluation, qu'en général les résultats sont acceptables. En effet, pour les deux types d'évaluation, on obtient un taux d'identification à 0,929. Par contre, la précision est faible surtout en macro-évaluation, et ceci vient de la confusion du système sur l'appartenance de certaines partitions aux auteurs notamment pour Monteverdi ou Palestrina où le taux de faux positifs est plus élevé. Ceci peut s'expliquer par le pourcentage de motifs discriminants spécifiques à ces auteurs qui est plus faible, comme indiqué dans le tableau 3.10, respectivement égal à 66% et 63% .

3.2.4.2 Expérience avec prise en compte des auteurs moins fréquents (Paramétrage 2) sur des données non équilibrées

Cette expérience diffère de la précédente par le fait qu'elle intègre des auteurs dont le processus d'identification n'a pas été considéré lors de la recherche de motifs et donc de la construction de l'index (Berlioz, Brumel, Cherubini, Mozart) (paramétrage 2). Les performances obtenues sont données dans la Table 3.14.

TABLE 3.14 – Résultats du processus d'identification d'auteurs sur des données non équilibrées – Paramétrage 2

Author	TP	TN	FP	FN	Accuracy	Precision	Recall	F1-score
Bach	102	0	16	0	0,864	0,864	1	0,864
Corelli	2	112	4	0	0,966	0,333	1	0,495
Haydn	1	109	5	3	0,932	0,166	0,25	0,281
Monteverdi	4	101	13	0	0,889	0,235	1	0,371
Palestrina	2	100	16	0	0,864	0,111	1	0,196
					Macro-évaluation			
					0,903	0,341	0,85	0,441
					Micro-évaluation			
					0,903	0,672	0,973	0,77

TABLE 3.15 – Résultats du processus d'identification d'auteurs sur des données équilibrées – Paramétrage 1

Author	TP	TN	FP	FN	Accuracy	Precision	Recall	F1-score
Bach	382	0	12	0	0,969	0,969	1	0,984
Corelli	1	305	87	1	0,776	0,011	0,5	0,223
Haydn	2	282	108	2	0,720	0,018	0,5	0,035
Monteverdi	3	300	90	1	0,769	0,031	0,75	0,061
Palestrina	1	202	190	1	0,515	0,005	0,5	0,010
					Macro-évaluation			
					0,75	0,207	0,65	0,222
					Micro-évaluation			
					0,75	0,444	0,987	0,612

On remarque qu'il y a une légère perte de performances et que les partitions des auteurs non connus dans la phase de construction de l'index ont créé plus de confusion pour le système. En conséquence, les occurrences des auteurs ont été en grande partie mal attribuées aux autres auteurs ce qui augmente les faux positifs et donc diminue la précision.

3.2.4.3 Expérience sans prise en compte des auteurs moins fréquents (Paramétrage 1) sur des données équilibrées

Dans cette expérience, on considère un échantillon équilibré pour la recherche de motifs ; ce qui revient à réduire le nombre de partitions de Bach considérées lors de cette étape à 20 au lieu de 300 et donc à augmenter le nombre de problèmes positifs pour ce compositeur lors de l'identification à 382 au lieu de 102. Les résultats obtenus sont donnés dans la table 3.15

On remarque qu'il y avait une diminution des performances. En effet, la réduction du

TABLE 3.16 – Résultats du processus d'identification d'auteurs sur des données équilibrées – paramétrage 2

Author	TP	TN	FP	FN	Accuracy	Precision	Recall	F1-score
Bach	382	0	16	0	0,959	0,959	1	0,979
Corelli	1	306	90	1	0,771	0,010	0,5	0,021
Haydn	2	284	110	2	0,716	0,017	0,5	0,034
Monteverdi	3	300	94	1	0,761	0,030	0,75	0,059
Palestrina	1	204	192	1	0,516	0,005	0,5	0,010
					Macro-évaluation			
					0,745	0,204	0,65	0,220
					Micro-évaluation			
					0,745	0,436	0,987	0,605

nombre d'œuvres de Bach a influencé la sélection des motifs discriminants de cet auteur et ceux des autres de telle sorte que ces motifs sont utilisés par tout le monde mais avec des poids supérieurs pour Bach. De ce fait, les performances pour les auteurs ont été perturbées.

3.2.4.4 Expérience avec prise en compte des auteurs moins fréquents (Paramétrage 2) sur des données équilibrées

Dans cette expérience, nous avons effectué la même analyse que celle effectuée sur le jeu de données non équilibrées et en considérant les auteurs non connus par le système (ne faisant pas partie des auteurs considérés lors de la construction de l'index). Le résultat obtenu est donné dans la table 3.16.

On constate les mêmes observations que dans le cas de la répartition équilibrée des données. Ceci montre les limites du système d'identification d'auteurs dans le cas où il n'y a pas assez de données pour identifier les motifs caractéristiques des compositeurs. Plus l'index est construit à partir d'un jeu de grande dimension, meilleures seront les performances lors de l'identification.

TABLE 3.17 – Résultats du processus d'identification d'auteurs

Type des motifs	COUNT	TP	TN	FP	FN	Accuracy	Precision	Recall	F1-score
IM	0.01	111	419	37	3	0,929	0,75	0,973	0,829
IR	0.06	108	428	28	6	0,940	0,794	0,947	0,864
IB	0.01	111	430	26	3	0,949	0,810	0.973	0,884
Variations	0.01	109	291	165	5	0,701	0,397	0,956	0,561
IM+IR+IB+Variations		103	463	3	11	**0,975**	**0,971**	**0,903**	**0,935**

3.2.4.5 Expérience avec considération des variantes de motifs sur des données non équilibrées sans considération des auteurs inconnus (Paramétrage 1)

Contrairement aux expériences précédentes, dans celle-ci nous considérons les motifs mais aussi leurs variantes. Les micro-évaluations obtenues sont données dans la table 3.17 où la colonne COUNT correspond au seuil du count sur lequel le système est plus performant en fonction du type du motif considéré. La dernière ligne correspond à la performance du système (notée **IM+IR+IB+Variations** tels que IM pour mélodiques identiques, IR pour rythmiques identiques, IB pour identiques à la fois mélodiques et rythmiques et Variations pour les variantes transposées, inversées et miroirs des motifs mélodiques) en se basant sur la décision collective. En d'autres terme, le système **IM+IR+IB+Variations** attribue une œuvre à un auteur A_i donné si tous les systèmes IM, IR, IB et Variations ont attribué cette œuvre à l'auteur A_i.

On remarque que le modèle combiné surclasse tous les autres modèles avec un taux de bien identifiés de 97.5%. Cette performance est fortement liée aux performances des systèmes IM, IR et IB. On remarque que le système IB est plus efficace que IM et IR par le fait qu'il a moins de FP et FN. Les résultats montrent aussi que le système basé sur les variantes de motifs est le moins performant par rapport aux autres. Ceci revient au nombre réduit des variantes dans les partitions musicales. Même si ce modèle a bien contribué dans l'amélioration des décisions du modèle **IM+IR+IB+Variations**.

TABLE 3.18 – Résultats Modèles de classification

	Précision	Rappel	F-score
Modèle de classification			
KNN (k=1)	0,784	0.962	0.655
Naive Bayes	0.963	0.937	0.947
Random Forest	0.717	0.959	0.586

3.3 Classification des œuvres musicales par auteur

Nous avons vu que le système d'identification d'auteurs peut attribuer plusieurs auteurs potentiels à une œuvre inconnue. Dans le cas d'un système de classification, on utilise un modèle de classification qui associe une et une seule classe pour une œuvre inconnue. Plusieurs étapes sont communes entres les deux types de traitements. On résume les étapes de construction du système de classification par les points suivants :

1. Extraction des motifs avec des contraintes minimales pour CSMA (minFreq = 2, minLen = 2)
2. Construction de l'index (document -> motifs) avec calcul des poids basé sur *TF-IDF*
3. Calcul du χ^2 pour chaque motif
4. Sélection des motifs les plus discriminants par auteur en fonction des valeurs du χ^2
5. Mise à jour de l'index
6. Mise en œuvre d'une méthode de classification pour déterminer l'auteur d'une partition donnée

Les données utilisées sont les données de NEUMA (en version non équilibrée). Pour cela nous avons combiné l'ensemble 1 et l'ensemble 2 pour construire un seul ensemble. La validation de cet ensemble de données se fait par la méthode de validation croisée avec un échantillonnage de 10% des données (*10-folds cross-validation*).

Pour le choix du modèle de classification nous avons testé plusieurs méthodes dans l'api Weka telles que le k-plus proches voisins (K= 1), Naive Bayes, et Random forest.

Les résultats obtenus sont résumés dans la table 3.18 où nous avons calculé les taux de reconnaissance moyens par modèle de classification.

La méthode Naive Bayes donne le meilleur résultat par rapport aux autres avec un taux de précision égale à 96.3%. Cependant, pour certains auteurs le système trouve des difficultés pour bien identifier leurs œuvres. La cause principale revient au fait qu'ils n'ont pas assez d'exemples dans l'ensemble 1 et que leurs motifs discriminants sont aussi utilisés par d'autres auteurs.

3.4 Conclusion

Dans cette partie nous sommes parvenus à mettre en pratique l'utilisation des motifs musicaux avec les variantes transposées, inversées et miroirs dans un système d'identification de compositeurs. Ceci prouve que les motifs musicaux font partie de la signature d'un style ou d'un compositeur.

En effet, nous nous sommes inspirés des méthodes de recherche d'information dans le calcul des poids des motifs et la construction de l'index (en considérant les motifs musicaux comme des termes). Ensuite, l'utilisation du χ^2 nous a permis de réduire l'ensemble des motifs en sélectionnant les motifs discriminants par auteur. A la fin, la méthode Dissimilarité Count, qui a déjà prouvé son efficacité sur les données textuelles, a encore montré ses bonnes performances dans le cas des données musicales.

Enfin, le modèle de données utilisé était facilement adaptable pour le faire fonctionner dans un modèle de classification d'auteurs. Nous avons trouvé que le modèle Naive Bayes était le plus performant pour cette tâche.

Conclusion générale et perspectives

3.5 Conclusion

La plupart des travaux en relation avec l'instrumentation d'œuvres musicales et plus généralement l'édition critique de partitions se réfère à des outils de reconnaissance de motifs harmoniques et se concentre sur des informations issues de flux sonores synchronisés [Donin et al., 2010] [Echeveste et al., 2011]. Dans le cas des corpus que nous étudions, ces données ne sont pas disponibles et il a donc été nécessaire de baser l'analyse sur des données non temporelles et non sonores mais qui sont néanmoins articulées entre elles : il s'agit des symboles musicaux issus des images. L'objectif scientifique de la thèse s'inscrit dans ce contexte avec l'objectif ciblé de découvrir les motifs musicaux fréquents. Transcrire directement les partitions aurait pu constituer un défi supplémentaire, sachant que leurs transcriptions induisent des coûts humains importants, notamment dans le domaine symphonique, celui qui est porté par la musicologue de notre projet Mylène Pardoen. On y rencontre notamment un grand nombre d'inédits ou des éditions souvent séparées et aucun matériel accessible pour permettre leur exécution et leur diffusion (comme les œuvres inédites de Louis Massoneau, 1766-1848, ou encore la collection du compositeur autrichien Sigismond Ritter von Neukomm, 1778-1858). Des connaissances très spécifiques sont requises pour la lecture de ces œuvres ainsi que leur publication. Les analyser passe ainsi par l'élaboration de grilles de lecture qui s'intéressent de près aux motifs musicaux, leurs fréquences, leurs caractéristiques et leur apparition dans la partition. Il va sans dire que se lancer dans un travail d'OMR aurait engagé une démarche moins subjective mais également sujette au critique sachant que l'interprétation faite par l'expert musicologue est une dimension que le système de trans-

cription ne peut pas intégrer. Rechercher les motifs en partant de la source image afin de mettre à disposition une signature de la composition constitue en ce sens un outil d'analyse experte pour faciliter certaines études (identification d'auteurs, classification par genre musical, étude des influences musicales, etc.). Nous avons proposé ici un dispositif d'analyse sans transcription de la partition s'intéressant à toute la chaîne de transformation : de la préparation des données sources images permettant de produire des séquences de primitives à la construction de bases de motifs dont on maîtrise certains paramètres structurels : la forme (miroir, translatée, inversée, etc.), la taille minimale, la fréquence minimale d'apparition, la taille et le nombre de gaps.

La thèse a également eu pour ambition de consolider les liens entre les sciences humaines et sociales et l'informatique, cela a pu se faire grâce à la collaboration inédite entre Mylène Pardoen, musicologue du projet, et les deux équipes spécialistes des données textes et images.

Dans ce travail, nous avons débuté par une étude générale impliquant les différentes étapes de pré-traitement, de détection et reconnaissance des primitives dans les partitions musicales manuscrites. Cette étude nous a conduit à adapter un modèle à base de réseaux de neurones profonds de type Region-Proposal Network (RPN) qui ont prouvé leur performance dans des tâches de reconnaissance d'objets dans les scènes naturelles. Nous avons conçu un pipeline complet exploitant la sortie du classifieur RPN pour construire une séquence de primitives couvrant l'information mélodique présente dans une partition.

La contribution majeure de la thèse et son originalité repose sur la conception d'un algorithme de fouille de séquence : nommé CSMA, qui a été mis au point pour extraire les motifs sans connaissances préalable de leurs positions ou leurs valeurs (i.e. de façon totalement non supervisée) sur une ou plusieurs séquences adjacentes de la partition. CSMA ne prend en compte que des contraintes liées à la fréquence minimale des motifs, la taille minimale et maximale des motifs, la taille maximales des gaps (sauts) et le nombre de gaps autorisés par occurrence de motif.

Dans ce travail, nous avons démontré une utilisation prometteuse des gaps (avec un gap par occurrence) pour l'extraction de motifs. En effet, le gap s'est avéré être une propriété intéressante permettant un gain de détection pouvant aller jusqu'à 13 % sur des séquences

bruitées (données synthétiques) et 7 % sur les données réelles de MUSCIMA++ (dont certains motifs clés ont été initialement mal encodés du fait des limites de reconnaissance des primitives du réseau de neurones dédié). La moyenne se situe autour de 5,7 % d'amélioration. L'utilisation du gap épargne ainsi la mise en place d'un dispositif supplémentaire de correction d'erreurs dans la création de la séquence (erreurs de reconnaissance et/ou erreur d'encodage). Pour valider les scénarios de reconnaissance et d'extraction de motifs, nous avons bâti l'ensemble de nos expériences sur la base MUSCIMA++. Rappelons également que l'algorithme CSMA a été conçu pour faciliter l'intégration de règles spécifiques pour la détection des variantes des motifs. Ces variantes peuvent être : des motifs transposés, des motifs inversés, et des motifs dits miroir. Ces motifs et leurs variantes ont été utilisés comme signatures des œuvres musicales dans un système d'identification de compositeurs (base NEUMA). Le système a démontré de bonnes performances de l'ordre de 97% de taux d'identification sur une base de cinq compositeurs de la base.

Ce travail mené de bout en bout permet aux musicologues de disposer d'outil d'aide à l'analyse et des recommandations pour répondre à leurs besoins.

3.6 Perspectives

Touchant à différents domaines de l'analyse d'images et de données ainsi que la musicologie, les travaux entrepris dans la thèse ouvrent des perspectives nombreuses, disciplinaires d'une part mais également à la croisée des domaines. Les perspectives sont à ce titre de diverses natures et touchent à la fois les contributions techniques, scientifiques et d'usage, à court, moyen et long terme.

Perspectives à court terme : Généraliser les expérimentations et la validation par l'expert Parmi les expériences à entreprendre rapidement, nous proposons d'adapter le RPN à un usage généralisé sur des données réelles (celles du projet) en ré-exploitant le réseau de neurones utilisé pour la détection et la reconnaissance des primitives au cas réels par transfert d'apprentissage. Cette partie pourra être validée en utilisant un jeu de données

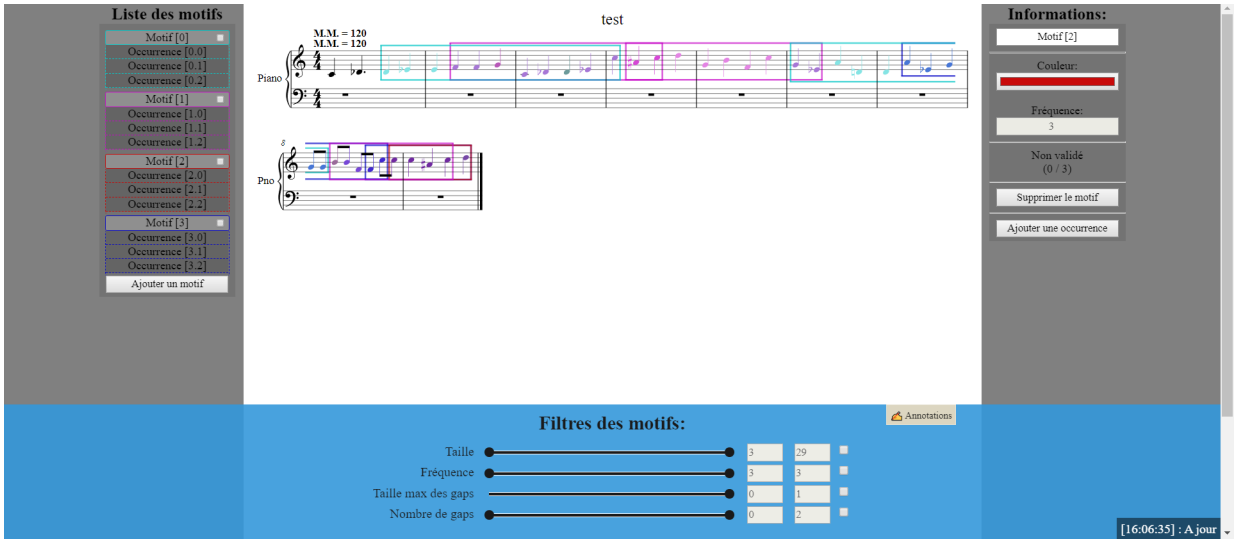


FIGURE 3.1 – IHM de validation des motifs

différent (comme DeepScores [2]) pour lequel on dispose d'une vérité terrain exploitable. Ces données pourront également être complétée par des données annotées de la base du projet, à partir de la création d'un jeu de données annotés (de plus petite taille) utilisant l'outil Muscimarker [3] ou labelImg [4] dédié à ces tâches d'annotations manuelles.

Nous envisageons également dans ces perspectives à court terme d'intégrer davantage de primitives dans le scénario de reconnaissance et d'encodage, notamment les primitives rythmiques et les variantes de motifs (motifs miroirs, inverses, monnayés). Naturellement cette tâche pourra se faire sur la collection du projet dont on sait que certaines partitions disposent de variantes effectivement présentes.

Afin d'aider la vérification experte de l'approche appliquée au projet, nous comptons également déployer l'interface de validation et de filtrage des motifs réalisées durant le projet pour l'experte musicologue du projet (voir figure 3.1). Cet outil de vérification est également un dispositif de visualisation des motifs qui reprojète dans l'espace de la partition en image les boîtes englobantes des motifs similaires (ou la coloration des notes directement par des code couleur distincts). A l'aide de cette interface l'utilisateur expert est en mesure de valider la détection et le bon alignement des motifs identiques (ou similaires). Cette information pourra ensuite être reprise et stockée pour une réflexion à plus long terme visant à apprendre des

2. https://tuggeluk.github.io/deepscores/
3. https://github.com/hajicj/MUSCIMarker
4. https://github.com/tzutalin/labelImg

erreurs d'associations et améliorer les performances du système (afin d'écarter les situations de découvertes de motifs similaires qui n'en sont pas et de créer une base de motifs négatifs que le système pourra ainsi apprendre à écarter (en post-traitement).

Perspectives à moyen terme : Généralisation de l'usage du gap et intégration des partitions à plusieurs voix Le gap a montré sa capacité à faire des 'sauts' d'une à plusieurs primitives (mal reconnues ou mal encodées) ce qui s'est avéré fort utile pour favoriser un meilleur alignement de motifs. Or en favorisant certaines bonnes associations, le gap en force également certaines autres à se faire alors que les différences sont volontaires. Forcer les alignements systématiquement peut provoquer certaines fusions intempestives de motifs et au lieu d'améliorer les appariements, on risque de créer du bruit supplémentaire. Afin de reprendre le contrôle sur les alignements, une analyse contextuelle (observation de la ré-occurrence des motifs voisins par exemple) pourrait ainsi aider à ne conserver que les associations porteuses d'une certaine sémantique musicale.

On envisage également de généraliser l'usage du gap en forçant le système à apprendre de ses erreurs par association des motifs erronés et des motifs similaires découverts, en cherchant à redécouvrir le motif cible d'origine (le motif réellement joué). Ce mécanisme pourra partir d'un clustering de motifs similaires (les motifs corrects, et les motifs découverts grâce au gap validés par un rebouclage avec l'expert musicologue) pour observer les situations d'erreurs fréquentes (primitives souvent mal reconnues ? erreur d'encodage due à une écriture difficile à lire ?) pour aider le système à améliorer ses étapes de reconnaissance et d'encodage.

A ce jour des travaux allant dans ce sens sont en court d'étude avec le laboratoire du CVC de Barcelone qui travaille sur la correction post-OMR sur des partitions musicales manuscrites du centre des archives de la ville (partitions des 18$^{\text{ème}}$ et 19$^{\text{ème}}$ siècle).

Enfin notons que la recherche de motifs ne s'effectue pour le moment que sur des partitions mono-voix pour lesquelles l'encodage est systématique et ne pose pas de problème d'interprétation temporelle. En apportant une composante multi-voix (i.e : jeux à deux mains pour les partitions de pianos par exemple, ou situation d'orchestre à plusieurs instruments), il est nécessaire de traiter les motifs parallèlement et de retrouver la temporalité/séquentialité des

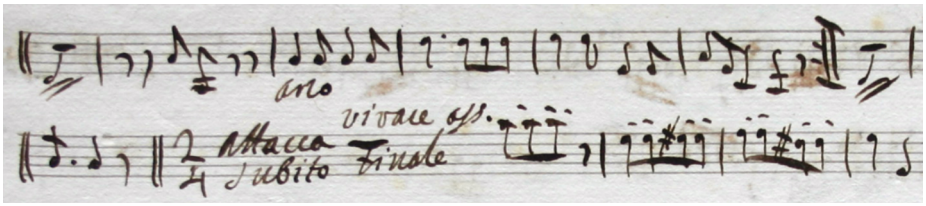


FIGURE 3.2 – Partition musicale avec des surcharges manuscrites

notes. Ce travail nécessite d'intégrer dans l'approche CSMA plusieurs dimensions d'analyse permettant de garder le lien entre les différentes voix jouées en même temps sur la partition.

Perspectives à long terme : reconnaître les annotations et surcharges manuscrites

De même que la reconnaissance de texte manuscrit pose des problèmes qui n'existent pas avec du texte imprimé, la reconnaissance de partitions manuscrites, particulièrement de partitions anciennes, invalide certaines hypothèses utilisées en OMR classique et soulève d'autres difficultés [Couasnon, 1996] [Baró et al., 2018]. Pour cette raison, les recherches entreprises dans ce domaine sont considérablement moins nombreuses et moins avancées que dans l'OMR imprimé. Bien que notre objectif ne se soit pas situé dans la mise en place d'une transcription de la partition, pour une analyse musicologique plus avancée, il est nécessaire de s'intéresser à l'information non musicale touchant les annotations et les surcharges manuscrites très présentes dans ces documents [Fornés et al., 2005], voir figure 3.2 [5]. Les annotations sont des informations périphériques importantes car elles doivent faciliter les interactions avec l'utilisateur et ses parcours analytiques. Leur intérêt a déjà été souligné en 2003 au sein du projet européen MusicWeb [6] pointant les relations sonores et morphologiques de motifs musicaux par un système interactif d'annotations musicales. Nous envisageons ainsi d'étendre la détection de primitives à celles des annotations manuscrites et autres surcharges dont le vocabulaire est généralement limité et connu.

Pour finir, nous pouvons conclure qu'au-delà d'un outil d'aide à la musicologie, la détection de motifs dans les partitions musicales est également un outil d'annotation efficace pouvant servir à des applications de détections de plagiat musical (reprise de certains fragments d'un

5. Devienne / François / 1759-1803 / 0220. La Bataille de Gemmap, sinfonie a grand orchestre exécutée au Concert du théâtre de la rue Feydeau[1794] - Source BNF - Gallica

6. `http://musicweb.koncon.nl`

auteur). Combiner ainsi l'information séquentielle et de nature textuelle issue de la lecture des notes à une information sonore issue de son signal pourrait conduire à l'élaboration d'une méthodologie plus complète d'études des influences musicales (études des variantes de motifs partielles ou complètes, détection de certaines formes de similarités non liées aux variantes elles-mêmes).

Annexes

Annexe A

Introduction à la musicologie

A.1 La ligne de portée (staff lines)

La portée est constituée de 5 lignes et 4 espaces qui permettent de connaître les hauteurs des notes et de positionner les figures des notes, silences, altérations, ainsi que tous les symboles utilisés pour la notation musicale.

La portée débute par un trait vertical qui marque son début, puis une clé et enfin une métrique.

La clé : Par son point d'accroche, la clé indique le nom et la hauteur de la note référente. Il existe 3 types de clé : sol, fa et ut.

A partir de ce positionnement, on peut placer toutes les autres notes de la gamme dans un ordre défini (do, ré, mi, fa, sol, la si et do). La gamme comporte 7 notes : do, ré, mi, fa sol, la, si

La clé de sol (généralement positionnée sur la deuxième ligne de la portée) attribue les fréquences naturelles pour chaque note. Plus on monte dans et hors la portée, plus le son est aigu, plus on descend, plus il est grave. Pour faciliter la lecture du musicien et la garder centrale sur la portée, on transpose dans une autre clé. Ainsi la note présente sur la ligne -2 de la portée à clé de sol (avec une clé posée sur la $2^{\text{ème}}$ ligne) est équivalente à la note présente sur la $5^{\text{ème}}$ ligne de la portée à clé de fa (clé de fa posée sur $4^{\text{ème}}$ ligne) (c.f. Figure A.1) [1].

1. Note : la source des figures présentes dans cet annexe proviennent du site `https://www.musictheory.net/lessons`

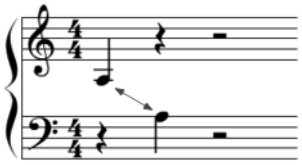


FIGURE A.1 – Clefs

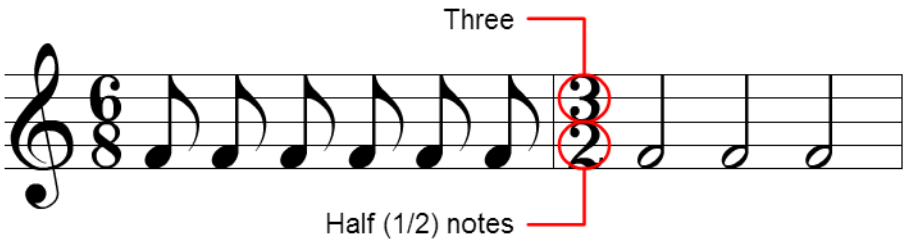


FIGURE A.2 – Mesures

A.2 La mesure

C'est une notion qui regroupe différentes définitions. Dans le cas présent, c'est une segmentation de la portée qui se matérialise par l'espace entre deux barres (appelées barre de mesure). Elle se définit par deux nombres (ou une lettre) positionnées juste après la clé. Cette division représente la rythmique et le temps.

Dans l'exemple de la figure A.2, la mesure 6/8 veut dire que cette mesure contient six battements et chaque battement à une durée équivalente à une croche (1/8).

Les différents types de notes sont présentés dans la figure A.3.

Remarque : dans une mesure 3/4 une blanche occupe deux battements.

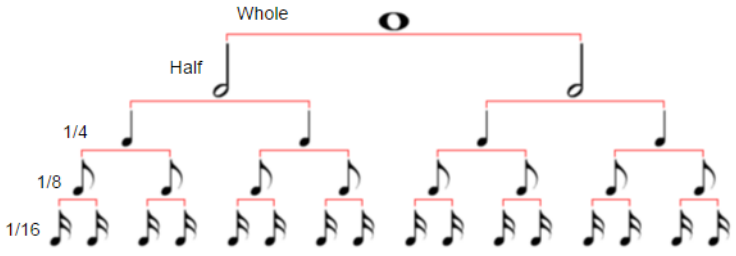


FIGURE A.3 – Les types de notes

FIGURE A.4 – Les points rythmiques

A.3 Les points et les liens

Le point positionné à côté de la note, comme illustré dans la figure A.4, matérialise une augmentation de sa durée de la moitié de sa valeur.

La liaison de prolongation relie deux notes de même hauteur, même au-delà des barres de mesures. Elle prolonge sa durée de la valeur de la note suivante.

Un lien est un trait reliant deux notes du même pitch (type + durée) ajoute une continuité de jeu entre les mesures.

A.4 Le bémol et le dièse

Dans l'échelle diatonique, le demi-ton représente le plus petit intervalle entre les notes. Cela représente à peu près la moitié de la valeur de l'intervalle qui existe entre deux notes conjointes. Il existe deux types de demi-ton : le demi-ton diachronique et le demi-ton chromatique. C'est le second qui nous intéresse ici.

Exemples : MI# =FA ; FAb = MI ; SI# = DO ; DOb = SI (b : bémol)

A.5 Tonalité d'un morceau

La tonalité est le système dans lequel on va écrire la musique. Elle se différencie de la modalité (sept modes utilisés dans le chant grégorien). Le système tonal se limite à deux modes : le majeur et le mineur. Ce système s'appuie sur une gamme dont l'articulation est la suivante : la note tonique (celle qui débutera la gamme référente, par exemple Do Majeur - ou Do Mineur), sa dominante et sa note sensible. Pour connaître la tonalité d'une gamme, il faut se reporter à l'armure positionnée en début de mesure.

Il existe une manière d'écriture des alternatives (b et #), elle consiste à les mettre au

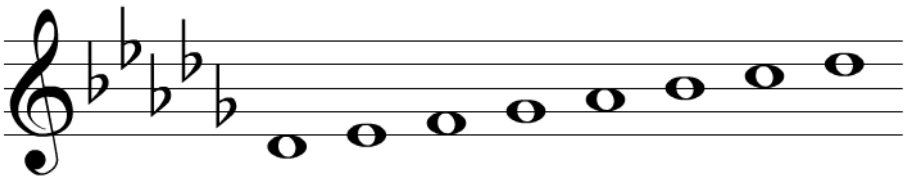


FIGURE A.5 – Écriture avec armure

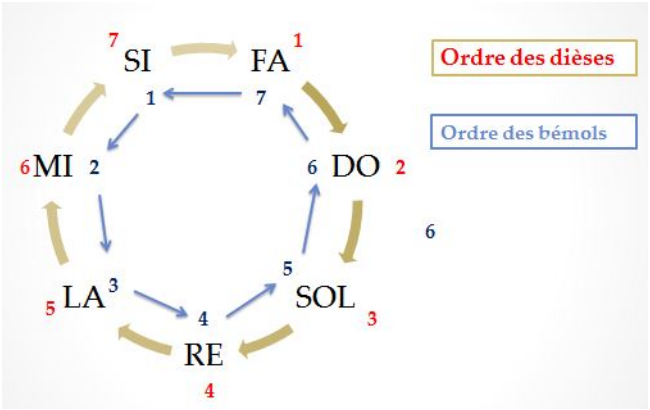


FIGURE A.6 – Ordres des altérations

début du solfège. Cette méthode est appelée une armure. La construction de l'armure est illustrée dans la figure A.5.

Cette armure définit une clé de signature qui nous permet de trouver systématiquement la tonalité du morceau. Ceci en se basant sur le type de symboles présents dans l'armure (b ou #) et avec l'aide du cercle suivant :

Pour trouver le type de tonalité on compte le nombre de dièses (resp. bémols) présents au début du solfège et on parcourt le cercle dans l'ordre des dièses (resp. dans l'ordre des bémols) et on se positionne au chiffre correspondant pour le # (resp. chiffre -1 correspondant pour le b). La note associée au nombre de dièses est majeure.

Par exemple si on a cinq bémols dans l'armure, alors la note correspondante dans le cercle est RE (celle avec le chiffre 4). Donc on conclut qu'on a deux possibilités (REb Majeur ou SI mineur). La deuxième possibilité vient de la gamme relative qui peut être déduite en diminuant un ton et demi de la note majeur (REb - 3/2 ton = SI). Du coup, pour chaque type d'armure il existe deux possibilités (tonalité majeur ou bien mineur). Pour prendre la décision finale il faut vérifier la dernière note du solfège si c'est un RE donc on est sur un REb Majeur et si c'est un SI on est sur un SI mineur.

Annexe B

Bases de données image de partitions musicales

B.1 HOMUS

Le jeu de données HOMUS (Handwritten Online Musical Symbols) [1] est un corpus de référence contenant environ 15 000 échantillons de 32 types de symboles musicaux manuscrits de 100 musiciens différents. Il est conçu pour la reconnaissance de la notation musicale manuscrite en ligne. L'ensemble des symboles traités dans cet ensemble de données est le suivant :

- **Notes** : ronde, blanche, noire, croche, double-croche, triple-croche, quadruple-croche.
- **Silences** : pause, soupir, demi-soupir, quart de soupir, huitième de soupir, huitième de soupir.
- **Altérations** : bémol, dièse, bécarre.
- **Division du temps** : 4-4, 2-2, 2-4, 3-4, 3-8, 6,8, 9-8, 12-8.
- **Clefs** : Clef de sol, Clef d'ut, Clef de fa.
- **Autres** : notes rythmiques, barre de mesure.

HOMUS contient 15200 fichiers texte, répartis dans des répertoires indépendants pour chaque musicien. Les fichiers comprennent :

1. `http://grfia.dlsi.ua.es/homus/`

— Libellé de l'échantillon

— Une ou plusieurs lignes, chacune avec un seul trait

— Chaque trait est un ensemble de points 2D séparés par un point-virgule. A son tour, chaque dimension est séparée par une virgule

Grâce au codage mis en place pour les notations musicales, les symboles musicaux peuvent être dessinés. En conséquence, la base de données HOMUS est également utilisée pour la reconnaissance hors ligne de symboles musicaux [Calvo-Zaragoza and Oncina, 2014].

B.2 CVC-MUSCIMA

La base de données CVC-MUSCIMA contient des images manuscrites de partitions de musique, spécialement conçues pour l'identification des auteurs et les tâches de détection de lignes de portée.

La base de données contient 1000 pages de musique écrites par 50 musiciens différents. Chaque auteur a transcrit les mêmes 20 pages de musique, en utilisant le même stylo et le même type de papier à musique (avec des lignes imprimées). L'ensemble des 20 partitions sélectionnées contient des partitions pour instruments solo et des partitions pour chœur et orchestre [Fornés et al., 2012].

B.3 MUSCIMA ++

MUSCIMA ++ est un ensemble de données de notation musicale manuscrite pour la détection de symboles musicaux. MUSCIMA ++ est basé sur le jeu de données MUSCIMA. Il contient 91255 symboles, composés à la fois de primitives musicales et de notations textuelles, tels que des signatures de clé ou des signatures temporelles. Le jeu de données contient 23352 notes, dont 21356 ont un en-tête de note complet, 1648 ont un en-tête de note vide et 348 sont des notes de grâce. Pour chaque objet annoté dans une image, les coordonnées et dimensions de chaque primitive sont donnés, ainsi que le masque de pixels permettant de reconstruire avec précision l'image de la primitive. Les constructions composites, telles que les notes, sont capturées par le biais de liens explicitement annotées entre les primitives de

notation (têtes de notes, hampes, ligatures, etc.). De cette manière, l'annotation constitue un lien entre les symboles de bas niveau (primitives) et de haut niveau (notes de musique complètes) décrits dans la littérature sur la reconnaissance de la musique [Fornés et al., 2012] [Hajič and Pecina, 2017].

Annexe C

Encodage MIDI et encodage primitives

TABLE C.1 – Table de correspondance entre l'encodage MIDI et l'encodage primitives

encodage midi	**encodage primitive**	**équivalence**	**Note**	**Octave**
33	NF26		A	1
34	#NF26	FNF25	A#	1
35	NF25	FNF24	B	1
36	NF24	#NF25	C	2
37	#NF24	FNF23	C#	2
38	NF23		D	2
39	#NF23	FNF22	D#	2
40	NF22	FNF21	E	2
41	NF21	#NF22	F	2
42	#NF21	FNF20	F#	2
43	NF20		G	2
44	#NF20	FNF19	G#	2
45	NF19		A	2
46	#NF19	FNF18	A#	2
47	NF18	FNF17	B	2
48	NF17	#NF18	C	3

49	#NF17	FNF16	C#	3
50	NF16		D	3
51	#NF16	FNF15	D#	3
52	NF15	FNF14	E	3
53	NF14	#NF15	F	3
54	#NF14	FNF13	F#	3
55	NF13		G	3
56	#NF13	FNF12	G#	3
57	NF12		A	3
58	#NF12	FNF11	A#	3
59	NF11	FNF10	B	3
60	NF10	#NF11	C	4
61	#NF10	FNF9	C#	4
62	NF9		D	4
63	#NF9	FNF8	D#	4
64	NF8	FNF7	E	4
65	NF7	#NF8	F	4
66	#NF7	FNF6	F#	4
67	NF6		G	4
68	#NF6	FNF5	G#	4
69	NF5		A	4
70	#NF5	FNF4	A#	4
71	NF4	FNF3	B	4
72	NF3	#NF4	C	5
73	#NF3	FNF2	C#	5
74	NF2		D	5
75	#NF2	FNF1	D#	5
76	NF1	FNF0	E	5

77	NF0	#NF1	F	5
78	#NF0	FNF-1	F#	5
79	NF-1		G	5
80	#NF-1	FNF-2	G#	5
81	NF-2		A	5
82	#NF-2	FNF-3	A#	5
83	NF-3		B	5
84	NF-4	#NF-3	C	6
85	#NF-4	FNF-5	C#	6
86	NF-5		D	6
87	#NF-5	FNF-6	D#	6
88	NF-6	FNF-7	E	6
89	NF-7	#NF-6	F	6
90	#NF-7	FNF-8	F#	6
91	NF-8		G	6
92	#NF-8	FNF-9	G#	6
93	NF-9		A	6
94	#NF-9	FNF-10	A#	6
95	NF-10	FNF-11	B	6
96	NF-11	#NF-10	C	7
97	#NF-11	FNF-12	C#	7
98	NF-12		D	7
99	#NF-12	FNF-13	D#	7
100	NF-13	FNF-14	E	7
101	NF-14	#NF-13	F	7
102	#NF-14	FNF-15	F#	7
103	NF-15		G	7
104	#NF-15	FNF-16	G#	7

105	NF-16		A	7
106	#NF-16	FNF-17	A#	7
107	NF-17	FNF-18	B	7
108	NF-18	#NF-17	C	8
109	#NF-18	FNF-19	C#	8
110	NF-19		D	8
111	#NF-19	FNF-20	D#	8
112	NF-20		E	8

Chapitre 4

Références bibliographiques

Bibliographie

[Agrawal and Srikant, 1995] Agrawal, R. and Srikant, R. (1995). Mining sequential patterns. In *Data Engineering, 1995*, pages 3–14. IEEE.

[Agrawal et al., 1994] Agrawal, R., Srikant, R., et al. (1994). Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499.

[Ayres et al., 2002] Ayres, J., Flannick, J., Gehrke, J., and Yiu, T. (2002). Sequential pattern mining using a bitmap representation. In *Proc. of the eighth ACM SIGKDD intern. conf. on Knowledge discovery and data mining*, pages 429–435. ACM.

[Baró et al., 2018] Baró, A., Riba, P., and Fornés, A. (2018). A starting point for handwritten music recognition.

[Bay et al., 2008] Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. (2008). Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3) :346–359.

[Calvo-Zaragoza and Gallego, 2019] Calvo-Zaragoza, J. and Gallego, A.-J. (2019). A selectional auto-encoder approach for document image binarization. *Pattern Recognition*, 86 :37–47.

[Calvo-Zaragoza and Oncina, 2014] Calvo-Zaragoza, J. and Oncina, J. (2014). Recognition of pen-based music notation : the homus dataset. In *22nd Intern. Conf. on Pattern Recognition (ICPR)*, pages 3038–3043. IEEE.

[Calvo-Zaragoza et al., 2017a] Calvo-Zaragoza, J., Pertusa, A., and Oncina, J. (2017a). Staff-line detection and removal using a convolutional neural network. *Machine Vision and Applications*, pages 1–10.

[Calvo-Zaragoza et al., 2017b] Calvo-Zaragoza, J., Vigliensoni, G., and Fujinaga, I. (2017b). Pixel-wise binarization of musical documents with convolutional neural networks. In *Fifteenth IAPR Intern. Conf. on Machine Vision Applications (MVA)*, pages 362–365. IEEE.

[Casas-Garriga, 2003] Casas-Garriga, G. (2003). Discovering unbounded episodes in sequential data. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 83–94. Springer.

[Chen et al., 2015] Chen, L., Wang, S., Fan, W., Sun, J., and Satoshi, N. (2015). Reconstruction combined training for convolutional neural networks on character recognition. In *Intern. Conf. on Document Analysis and Recognition*, pages 431–435. IEEE.

[Comaniciu et al., 2001] Comaniciu, D., Ramesh, V., and Meer, P. (2001). The variable bandwidth mean shift and data-driven scale selection. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 1, pages 438–445. IEEE.

[Couasnon, 1996] Couasnon, B. (1996). *Segmentation et reconnaissance de documents guidées par la connaissance a priori : application aux partitions musicales*. PhD thesis, Rennes 1.

[Couasnon, 2001] Couasnon, B. (2001). Dmos : A generic document recognition method, application to an automatic generator of musical scores, mathematical formulae and table structures recognition systems. In *Proceedings of Sixth International Conference on Document Analysis and Recognition*, pages 215–220. IEEE.

[Coüasnon, 2006] Coüasnon, B. (2006). Dmos, a generic document recognition method : Application to table structure analysis in a general and in a specific way. *International Journal of Document Analysis and Recognition (IJDAR)*, 8(2-3) :111–122.

[Coupé et al., 2011] Coupé, P., Manjón, J. V., Fonov, V., Pruessner, J., Robles, M., and Collins, D. L. (2011). Patch-based segmentation using expert priors : Application to hippocampus and ventricle segmentation. *NeuroImage*, 54(2) :940–954.

[Csurka et al., 2004] Csurka, G., Dance, C., Fan, L., Willamowski, J., and Bray, C. (2004). Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, volume 1, pages 1–2. Prague.

[Daher et al., 2010] Daher, H., Gaceb, D., Eglin, V., Bres, S., and Vincent, N. (2010). Décomposition des manuscrits anciens en graphèmes et construction des codes book basée sur la coloration de graphe. In *COmpression et REprésentation des Signaux Audiovisuels (CORESA)*, page 105.

[Dai et al., 2016] Dai, J., Li, Y., He, K., and Sun, J. (2016). R-fcn : Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387.

[Dalitz et al., 2008] Dalitz, C., Droettboom, M., Pranzas, B., and Fujinaga, I. (2008). A comparative study of staff removal algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(5) :753–766.

[Donin et al., 2010] Donin, N., Goldszmidt, S., and Theureau, J. (2010). Instrumenter les opérations d'écoute analytique ? un bilan du projet "écoutes signées"(2003-2006). *Journées d'Informatique Musicale (JIM 2010)*, pages 165–174.

[Dorfer et al., 2017] Dorfer, M., Hajič, J., and Widmer, G. (2017). On the potential of fully convolutional neural networks for musical symbol detection. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 2, pages 53–54. IEEE.

[Dunham, 2006] Dunham, M. H. (2006). *Data mining : Introductory and advanced topics*. Pearson Education India.

[Dutta et al., 2010] Dutta, A., Pal, U., Fornes, A., and Llados, J. (2010). An efficient staff removal approach from printed musical documents. In *2010 20th International Conference on Pattern Recognition*, pages 1965–1968. IEEE.

[Echeveste et al., 2011] Echeveste, J., Cont, A., Jacquemard, F., and Giavitto, J.-L. (2011). Formalisation des relations temporelles entre une partition et une performance musicale dans un contexte d'accompagnement automatique.

[Egho, 2014] Egho, E. (2014). *Extraction de motifs séquentiels dans des données séquentielles multidimensionnelles et hétérogenes–Une applicationa l'analyse de trajectoires de patients*. PhD thesis, Université de Lorraine.

[Fang et al., 2017] Fang, L., Li, S., Cunefare, D., and Farsiu, S. (2017). Segmentation based sparse reconstruction of optical coherence tomography images. *IEEE Transactions on Medical Imaging*, 36(2) :407–421.

[Fayyad et al., 1996] Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. (1996). The kdd process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 39(11) :27–34.

[Floratou et al., 2011] Floratou, A., Tata, S., and Patel, J. M. (2011). Efficient and accurate discovery of patterns in sequence data sets. *IEEE Transactions on Knowledge and Data Engineering*, 23(8) :1154–1168.

[Fornes et al., 2011] Fornes, A., Dutta, A., Gordo, A., and Llados, J. (2011). The icdar 2011 music scores competition : Staff removal and writer identification. In *Intern. Conf. on Document Analysis and Recognition*, pages 1511–1515. IEEE.

[Fornés et al., 2012] Fornés, A., Dutta, A., Gordo, A., and Lladós, J. (2012). Cvc-muscima : a ground truth of handwritten music score images for writer identification and staff removal. *Intern. Conf. on Document Analysis and Recognition*, pages 1–9.

[Fornés et al., 2005] Fornés, A., Lladós, J., and Sánchez, G. (2005). Primitive segmentation in old handwritten music scores. In *International Workshop on Graphics Recognition*, pages 279–290. Springer.

[Fornés et al., 2007] Fornés, A., Lladós, J., and Sánchez, G. (2007). Old handwritten musical symbol classification by a dynamic time warping based method. In *International Workshop on Graphics Recognition*, pages 51–60. Springer.

[Fournier-Viger et al., 2014] Fournier-Viger, P., Gomariz, A., Campos, M., and Thomas, R. (2014). Fast vertical mining of sequential patterns using co-occurrence information. In *Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, pages 40–52. Springer.

[Fournier-Viger et al., 2017] Fournier-Viger, P., Lin, J. C.-W., Kiran, R. U., and Koh, Y. S. (2017). A survey of sequential pattern mining. *Data Science and Pattern Recognition*, 1(1) :54–77.

[Frery et al., 2015] Frery, J., Largeron, C., and Juganaru-Mathieu, M. (2015). Author identification by automatic learning. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 181–185. IEEE.

[Fuchs, 2012] Fuchs, B. (2012). Co-construction interactive de connaissances, application à l'analyse mélodique. In *IC 2011, 22èmes Journées francophones d'Ingénierie des Connaissances*, pages 705–722.

[Fukunaga and Hostetler, 1975] Fukunaga, K. and Hostetler, L. (1975). The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on information theory*, 21(1) :32–40.

[Géraud, 2014] Géraud, T. (2014). A morphological method for music score staff removal. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 2599–2603. IEEE.

[Girshick et al., 2014] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proc. of the IEEE Conf. on computer vision and pattern recognition*, pages 580–587.

[Guedes et al., 2009] Guedes, C., Capela, G. A., Rebelo, A. M., da Costa, J. P., and Cardoso, J. (2009). Staff detection with stable paths.

[Hajič and Pecina, 2017] Hajič, jr., J. and Pecina, P. (2017). In Search of a Dataset for Handwritten Optical Music Recognition : Introducing MUSCIMA++. *CoRR*.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[Ho et al., 2016] Ho, A. K. N., Eglin, V., Ragot, N., and Ramel, J.-Y. (2016). Multi one-class incremental svm for document stream digitization. In *12th IAPR International Workshop on Document Analysis Systems (DAS 2016)*.

[Hong et al., 2007] Hong, Y., Yi, J., and Zhao, D. (2007). Improved mean shift segmentation approach for natural images. *Applied mathematics and computation*, 185(2) :940–952.

[Howe, 2011] Howe, N. R. (2011). A laplacian energy for document binarization. In *2011 International Conference on Document Analysis and Recognition*, pages 6–10. IEEE.

[Howe, 2013] Howe, N. R. (2013). Document binarization with automatic parameter tuning. *International Journal on Document Analysis and Recognition (IJDAR)*, 16(3) :247–258.

[Hsu et al., 1998] Hsu, J.-L., Chen, A. L., and Liu, C.-C. (1998). Efficient repeating pattern finding in music databases. In *Proc. of the 7th intern. conf.on Information and knowledge management*, pages 281–288. ACM.

[Jiménez et al., 2011] Jiménez, A., Molina-Solana, M., Berzal, F., and Fajardo, W. (2011). Mining transposed motifs in music. *Journal of Intelligent Information Systems*, 36(1) :99–115.

[Jurie and Triggs, 2005] Jurie, F. and Triggs, B. (2005). Creating efficient codebooks for visual recognition. In *Tenth IEEE Intern. Conf. on Computer Vision, ICCV 2005.*, volume 1, pages 604–610. IEEE.

[Katz et al., 2007] Katz, S., Tal, A., and Basri, R. (2007). Direct visibility of point sets. In *ACM Transactions on Graphics (TOG)*, volume 26, page 24. ACM.

[Kefali et al., 2014] Kefali, A., Sari, T., and Bahi, H. (2014). Foreground-background separation by feed-forward neural networks in old manuscripts. *Informatica*, 38(4).

[Kruskal, 1983] Kruskal, J. B. (1983). An overview of sequence comparison : Time warps, string edits, and macromolecules. *SIAM review*, 25(2) :201–237.

[Laxman and Sastry, 2006] Laxman, S. and Sastry, P. S. (2006). A survey of temporal data mining. *Sadhana*, 31(2) :173–198.

[Lazzara and Géraud, 2014] Lazzara, G. and Géraud, T. (2014). Efficient multiscale sauvola's binarization. *International Journal on Document Analysis and Recognition (IJDAR)*, 17(2) :105–123.

[Lee et al., 2016] Lee, S., Son, S. J., Oh, J., and Kwak, N. (2016). Handwritten music symbol classification using deep convolutional neural networks. In *Intern. Conf. on Information Science and Security (ICISS)*, pages 1–5. IEEE.

[Liu et al., 1999] Liu, C.-C., Hsu, J.-L., and Chen, A. L. (1999). Efficient theme and non-trivial repeating pattern discovering in music databases. In *15th Intern. Conf. on Data Engineering*, pages 14–21. IEEE.

[Liu et al., 2016] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). Ssd : Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer.

[Long et al., 2015] Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440.

[Mannila et al., 1997] Mannila, H., Toivonen, H., and Verkamo, A. I. (1997). Discovery of frequent episodes in event sequences. *Data mining and knowledge discovery*, 1(3) :259–289.

[Manning et al., 2008] Manning, C. D., Raghavan, P., and Schutze, H. (2008). Introduction to information retrieval. *Cambridge University Press*, 20 :405–416.

[Méger and Rigotti, 2004] Méger, N. and Rigotti, C. (2004). Constraint-based mining of episode rules and optimal window sizes. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 313–324. Springer.

[Navarro, 2001] Navarro, G. (2001). A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33(1) :31–88.

[Oikonomopoulos et al., 2005] Oikonomopoulos, A., Patras, I., and Pantic, M. (2005). Spatiotemporal salient points for visual recognition of human actions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 36(3) :710–719.

[Pacha et al., 2018a] Pacha, A., Choi, K.-Y., Coüasnon, B., Ricquebourg, Y., Zanibbi, R., and Eidenberger, H. (2018a). Handwritten music object detection : Open issues and baseline results. In *Inter. Workshop on Document Analysis Systems*.

[Pacha et al., 2018b] Pacha, A., Hajič, J., and Calvo-Zaragoza, J. (2018b). A baseline for general music object detection with deep learning. *Applied Sciences*.

[Pardoen, 2012] Pardoen, M. (2012). Projet Le Magasin de Musique BIS. `http://ladehis.ehess.fr/index.php?592`. Accessed : 2018-11-14.

[Pastor-Pellicer et al., 2015] Pastor-Pellicer, J., España-Boquera, S., Zamora-Martínez, F., Afzal, M. Z., and Castro-Bleda, M. J. (2015). Insights on the use of convolutional neural

networks for document image binarization. In *International Work-Conference on Artificial Neural Networks*, pages 115–126. Springer.

[Piątkowska et al., 2012] Piątkowska, W., Nowak, L., Pawłowski, M., and Ogorzałek, M. (2012). Stafflines pattern detection using the swarm intelligence algorithm. In *International Conference on Computer Vision and Graphics*, pages 557–564. Springer.

[Pratikakis et al., 2016] Pratikakis, I., Zagoris, K., Barlas, G., and Gatos, B. (2016). Icfhr2016 handwritten document image binarization contest (h-dibco 2016). In *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 619–623. IEEE.

[Rebelo et al., 2010] Rebelo, A., Capela, G., and Cardoso, J. S. (2010). Optical recognition of music symbols. *Inter. journal on document analysis and recognition*, 13(1) :19–31.

[Rebelo and Cardoso, 2013] Rebelo, A. and Cardoso, J. S. (2013). Staff line detection and removal in the grayscale domain. In *2013 12th International Conference on Document Analysis and Recognition*, pages 57–61. IEEE.

[Ren et al., 2015] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn : Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99.

[Ronneberger et al., 2015] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net : Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241. Springer.

[Sahli et al., 2013] Sahli, M., Mansour, E., and Kalnis, P. (2013). Parallel motif extraction from very long sequences. In *Proc. of the 22nd ACM Intern. Conf. on Information & Knowledge Management*, pages 549–558. ACM.

[Srikant and Agrawal, 1996] Srikant, R. and Agrawal, R. (1996). *Mining sequential patterns : Generalizations and performance improvements*. Springer.

[Su et al., 2012a] Su, B., Lu, S., Pal, U., and Tan, C. L. (2012a). An effective staff detection and removal technique for musical documents. In *2012 10th IAPR International Workshop on Document Analysis Systems*, pages 160–164. IEEE.

[Su et al., 2012b] Su, B., Lu, S., and Tan, C. L. (2012b). Robust document image binarization technique for degraded document images. *IEEE transactions on image processing*, 22(4) :1408–1417.

[Szegedy et al., 2017] Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, volume 4, page 12.

[Vincent et al., 2010] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked denoising autoencoders : Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(Dec) :3371–3408.

[Visani et al., 2013] Visani, M., Kieu, V. C., Fornés, A., and Journet, N. (2013). Icdar 2013 music scores competition : Staff removal. In *Intern. Conf. on Document Analysis and Recognition*, pages 1407–1411. IEEE.

[Wang et al., 2006] Wang, Y.-G., Yang, J., and Chang, Y.-C. (2006). Color–texture image segmentation by integrating directional operators into jseg method. *Pattern Recognition Letters*, 27(16) :1983–1990.

[Wen et al., 2015] Wen, C., Rebelo, A., Zhang, J., and Cardoso, J. (2015). A new optical music recognition system based on combined neural network. *Pattern Recognition Letters*, 58 :1–7.

[Yang et al., 1999] Yang, Y., Liu, X., et al. (1999). A re-examination of text categorization methods. In *Sigir*, volume 99, page 99.

[Zaki, 2001] Zaki, M. J. (2001). Spade : An efficient algorithm for mining frequent sequences. *Machine learning*, 42(1) :31–60.

[Zheng et al., 2009] Zheng, L., Zhang, J., and Wang, Q. (2009). Mean-shift-based color segmentation of images containing green vegetation. *Computers and Electronics in Agriculture*, 65(1) :93–98.