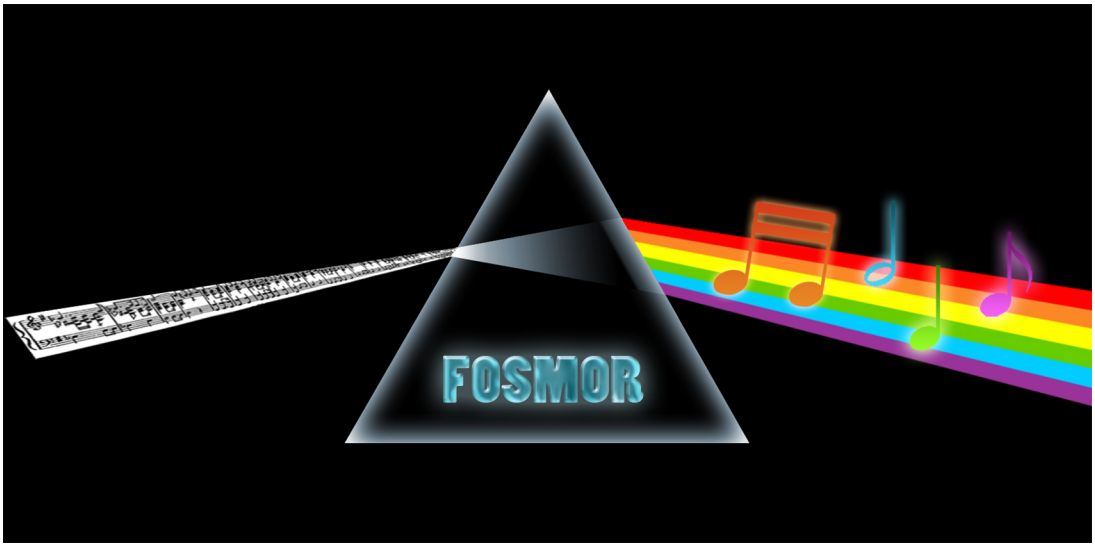
FOSMOR

Fooo Optical Sheet Music Recognition

Cahier des charges

le 21 Novembre 2008



Félix *Flx* **Abecassis** (*abecas_e*)
Christopher *Vjeux* **Chedeau** (*chedea_c*)
Vladimir *Vizigrou* **Nachbaur** (*nachba_v*)
Alban *Banban* **Perillat-Merceroz** (*perill_a*)

Table des matières

1 *Introduction* 3
1.1 Recherche Scientifique 4
1.2 Ambitions 4

2 *Le groupe de projet* 5

3 *Pré-traitement de l'image* 7
3.1 Conversion en bitmap 7
3.2 Passage en niveaux de gris 7
3.3 Réduction du nombre de niveaux de gris 8
3.4 Rotation de l'image 8
3.4.1 Détection du biais 8
3.4.2 Rotation de l'image 9
3.5 Réduction du bruit 9
3.5.1 Les filtres linéaires 9
3.5.2 Les filtres non-linéaires 10

4 *Analyse de l'image et extraction des symboles* 11
4.1 Détection des portées 11
4.1.1 Interligne et Epaisseur de ligne 11
4.1.2 Projection horizontale 12
4.2 Détection des Formes 13
4.2.1 Suppression des lignes de portée 13
4.2.2 Extraction des primitives 13
4.3 Objectifs 14

5 *Analyse des Symboles* 15
5.1 Problématique 15
5.2 Représentation matricielle 17
5.3 Transformation vectorielle 18
5.4 Moments de Zernike 19

5.5 Objectifs 20

6 *Apprentissage* 21
6.1 Présentation 21
6.2 Analogie biologique 22
6.3 Structure 24
6.4 Différents modes d'apprentissage 25
6.5 Différents types de réseaux de neurones 26
6.6 Perceptron Multicouche 28
6.7 Objectifs 30

7 *Conclusion* 31

Chapitre 1

Introduction

La reconnaissance de partitions musicales s'inscrit dans le domaine plus vaste de la reconnaissance de documents numérisés. On la désigne généralement par l'acronyme OMR pour Optical Music Recognition.

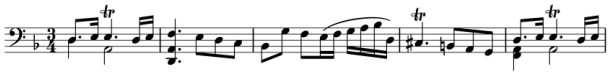


FIGURE 1.1 – Extrait de Sarabande de Johann Sebastian Bach

A priori, la reconnaissance des partitions musicales semble assez simple. En effet, l'alphabet que nous avons a manipuler est petit et la position de chaque élément obéit à des règles de solfège.

Les problèmes à résoudre sont néanmoins très nombreux. On peut citer des difficultés liées à l'interconnexion de plusieurs symboles, des défauts d'impression ou la mise en page approximative, à des symboles qui diffèrent de police.

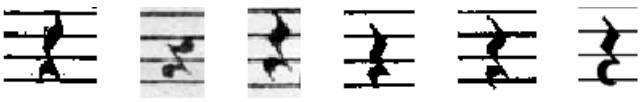


FIGURE 1.2 – Exemple de difficulté : Différente représentations d'un symbole

La forte structure des partitions musicales n'est qu'apparente, les règles sont en réalité très souples et peuvent varier énormément d'une partition à une autre.

Les spécificités de la notation musicale rendent le processus de reconnaissance très différent des domaines relatifs à l'analyse de documents comme l'OCR. Les problèmes qu'ils soulèvent nécessitent des solutions innovantes.

1.1 Recherche Scientifique

La reconnaissance musicale est un domaine très récent, les premières recherches ont été effectuées dans les années 70. Il faudra attendre les années 90 pour voir apparaître le premier logiciel commercial. Cet écart s'explique par la faible puissance de calcul des ordinateurs de l'époque ainsi que de la piètre qualité des images numérisées.

Les solutions actuelles existantes sont encore loin de fournir des résultats probants. Sur la dizaines de logiciels testés, pas un seul n'a été capable de fournir une transcription parfaite d'une partition simple générée par ordinateur, donc sans aucune imperfection.

La littérature propose de nombreuses solutions aux différents problèmes. Malheursement très rares sont les programmes complets qui utilisent l'ensemble de ces techniques.

Il est très difficile de comparer les différents algorithmes pour plusieurs raisons. Tout d'abord les explications demeurent très théoriques et il faut avoir de la chance pour trouver quelque code que ce soit. Si par chance celui-ci existe, il est certainement développé dans un language exotique. Ainsi pour comparer deux algorithmes il faut les réimplémenter soi-même.

Ensuite, il faut utiliser des partitions de base afin d'étudier les algorithmes. Or il n'existe pas d'ensemble unifié de partitions sur lesquels procéder aux tests. Chacun va choisir les partitions qui l'arrangent, et l'on se retrouve avec des chiffres proches de 100

Il existe une multitude de difficultés différentes. Il est donc très difficile de toutes les gérer. Une combinaison naïve d'algorithme ne suffit pas, une structure extrêmement souple est nécessaire.

1.2 Ambitions

Nous n'avons pas la prétention de pouvoir faire mieux que les nombreux chercheurs qui se sont lancés sur ce sujet. Notre objectif est de réaliser une solution complète capable, à partir d'une partition sous forme d'image, d'exporter le contenu musical vers un format de musique normalisé.

Les recherches existantes vont être la base de notre travail. La combinaison et adaptation originale des algorithmes va nous permettre d'arriver à un programme qui fonctionne.

A l'instar de la majorité des recherches à ce sujet, nous voulons que notre travail puisse être utilisé par tout le monde. Ainsi grâce à diverses techniques que vous allez découvrir prochainement, il ne sera pas nécessaire d'être sous une machine Unix avec Ocaml pour utiliser notre outil.

Chapitre 2

Le groupe de projet

L'année dernière nous étions déjà tous ensembles dans la Fooo Team pour la réalisation de notre projet de jeu de stratégie en temps réel. Notre groupe a bien fonctionné puisque nous sommes restés unis jusqu'au bout, il n'y a eu aucun problème, nous avons collaborés de manière efficace.

Nous sommes aujourd'hui fiers de ce que nous avons obtenu l'année dernière et nous espérons travailler sur ce nouveau projet cette année avec autant d'efficacité et de bonne entente.

Voici les quatre membres de notre équipe et la partie dont il aura la responsabilité pendant le développement du projet FOSMOR.

Chedeau Christopher : A pour mission d'analyser l'image afin d'y extraire les différents symboles et leur donnner du sens. C'est un travail qui n'est pas encore arrivé à maturité. Vu qu'il est actuellement très difficile de laisser une trace dans le monde scientifique, voici une bonne occasion de voir s'il est possible d'ajouter sa pierre à l'édifice.

Félix Abecassis : S'occupe de la partie apprentissage par réseau de neurone avec Vladimir. Il se concentrera aussi sur les différentes méthodes pour caractériser l'aspect d'un symbole afin d'en faciliter la classification et la comparaison.

Élève appréciant les challenges algorithmiques, il fera son possible pour obtenir les meilleurs résultats possibles en termes de reconnaissance de symboles, même si cela doit demander un important investissement et des recherches acharnées sur certaines notions mathématiques et algorithmiques.

Vladimir Nachbaur : Chef de projet de la Fooo Team pour la deuxième année consécutive, Vladimir travaillera aussi en grande partie avec Félix au niveau du réseau de neurones. Plus précisément, la recherche et les optimisations au niveau des algorithmes. Tout comme Félix, Vladimir a toujours aimé relever les défis algorithmiques au moins autant que la recherche dans

ce domaine.

Alban Perillat-Merceroz : Réalisera le pré-traitement de l'image, afin que celle-ci soit exploitable plus facilement par la suite. Le traitement d'image est pour lui un domaine nouveau dans lequel il va plonger avec le même enthousiasme que pour le projet de l'an dernier.

Il s'occupera accessoirement de mettre en place un site web permettant de suivre l'avancement du projet ainsi que de télécharger les sources et binaires du programme. Le domaine fooo.fr permettra d'acceder à la fois à l'ancien et au nouveau site de la Fooo Team.

Chapitre 3

Pré-traitement de l'image

Le pré-traitement de l'image a pour but de transformer n'importe quelle image quelle que soit sa qualité d'origine en une image "parfaite" : les défauts qui pourraient nuire au traitement des informations doivent être gommés.

Les principaux défaut que l'on peut rencontrer sont le bruit et l'inclinaison du texte.

Afin de faciliter le traitement, on modifiera aussi l'image afin de la simplifier : conversion en bitmap, passage en niveau de gris et enfin réduction du nombre de tons de couleurs à 2 (binarisation) 4 ou 8 tons par exemple.

3.1 Conversion en bitmap

Afin de ne pas passer trop de temps à implémenter le support de nombreux les formats d'image, il est nécessaire d'une part de convertir toutes les images dans un même format facile à utiliser.

Afin de ne pas perdre de temps dans la reconnaissance de tous les formats d'image, nous utiliserons une bibliothèque de fonctions nous permettant de convertir n'importe quel format d'image vers le format que nous utiliserons.

Il existe de nombreuses bibliothèques de fonctions parmi lesquelles on peut citer ImageMagic [1] ou bien DevIL [2]

3.2 Passage en niveaux de gris

Les couleurs éventuelles du document ne sont pas utiles à la reconnaissance du document. Pour simplifier le problème, on modifiera donc l'image

1. http ://www.imagemagick.org
2. http ://openil.sourceforge.net

pour n'afficher que des niveaux de gris selon la luminance de chaque pixel (l'intensité de lumière - indépendante de la couleur).

Pour calculer la luminance de chaque pixel en fonction de sa couleur, on utilise la formule suivante : Y = 0.299 R + 0.587 G + 0.114 B (R G et B étant les composantes Rouges Vertes et Bleues du pixel).

3.3 Réduction du nombre de niveaux de gris

Afin de simplifier la reconnaissance, il est encore nécessaire de simplifier les données. Il s'agit tout simplement de réduire le nombre de niveaux de gris de 255 à une valeur beaucoup plus faible : 2, 4 ou 8 par exemple.

Une valeur faible permet un traitement simplifié au détriment de la qualité de l'image.

Il va donc falloir trouver le bon compromis entre la qualité et la simplicité.

3.4 Rotation de l'image

Il est indispensable de fournir aux traitements ultérieurs une image dont les partitions sont horizontales pour permettre une reconnaissance la mieux qu'il soit.

Il faut donc pour cela détecter la meilleure orientation possible de l'image et appliquer la rotation adéquate.

3.4.1 Détection du biais

Naïvement, on utilise une méthode brute pour trouver la meilleure orientation de l'image : pour chaque rotation de l'image, on compare la variance du nombre de pixels par ligne.

Une grande variance indique que les lignes sont globalement soit très remplies, soit très peu remplies. La variance maximale indique donc le meilleur angle de rotation de l'image.

Plusieurs optimisations sont nécessaires à l'obtention d'un temps d'exécution acceptable : pour détecter le biais, il n'est pas nécessaire d'utiliser une image de très haute qualité. L'image peut donc être préalablement réduite.

Une réduction par deux semble être la méthode la plus simple et la plus rapide, tout en gardant les informations nécessaires au calcul du biais.

Dans un deuxième temps on utilisera une technique de rotation rapide mais approximative : une rotation ne prenant que la valeur du pixel aux coordonnées entières. Encore une fois l'approximation de cette rotation est suffisante pour calculer le biais.

3.4.2 Rotation de l'image

Il existe de nombreuses techniques de rotation. Hormis la rotation simple citée ci-dessus qui après rotation ne retiens que les pixels ayant une valeur entière, on retiendra l'interpolation bilinéaire.

On calcule les coordonnées théoriques du pixel après rotation (coordonnées qui ne sont pas entières) et on récupère dans l'image d'origine les quatre points les plus proches (dont les coordonnées sont entières). La couleur du pixel de destination sera donc une interpolation pondérée par des coefficients inversement proportionnels à la distance entre le point théorique et les quatre points de référence.

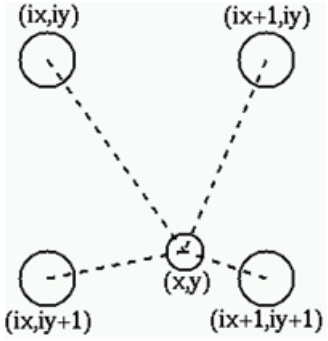


FIGURE 3.1 – La couleur du pixel (x,y) dépend de celle quatre points aux coordonnées entières

3.5 Réduction du bruit

Les algorithmes de réduction de bruits son nombreux et ont chacun leurs avantages. Il existe deux types de filtres : les filtres linéaires et les filtres non-linéaires.

3.5.1 Les filtres linéaires

Les filtres linéaires (ou convulsions) sont des fonctions qui remplacent chaque pixel par une somme pondérée de ses voisins. Une matrice définissant la taille du voisinage contient la taille respective de chaque pixel voisin : c'est le masque de convulsion.

Le filtre Gaussien par exemple permet d'atténuer les variations de lumière dans le voisinage d'un pixel. Un des principaux défauts de ce filtre est qu'il crée un flou sur les contours des formes de l'image.

3.5.2 Les filtres non-linéaires

Le filtrage non-linéaire fait intervenir les pixels voisins selon une loi non linéaire.

Le filtre médian permet d'éliminer les pixels isolés qui pourraient induire de fortes réponses lors de la détection des contours.

Pour cela, on utilise un masque de taille n*n. Le pixel central reçoit comme valeur la valeur moyenne des pixels environnant. Plus le masque est grand, moins le détecteur est sensible au bruit et plus l'erreur de localisation grandit. Il faut donc ne pas utiliser de filtres de tailles trop grandes.

Chapitre 4

Analyse de l'image et extraction des symboles

Le processus globalement utilisé est constitué de quatre étapes allant du plus bas niveau au plus haut niveau. A chaque étape les données à traiter sont de plus en plus faibles en mémoire. Ainsi le temps de traitement est plus rapide, ce qui permet une complexification des algorithmes sans pour autant affecter les performances.

- Détection des lignes de portée.
- Segmentation, généralement après suppression des lignes de portée.
- Reconnaissance des primitives segmentées, et réassemblage des symboles composés.
- Analyse syntaxique et sémantique.

4.1 Détection des portées

Les portées constituent le support graphique sur lequel sont disposés les symboles musicaux. Selon leur position par rapport à la portée, les symboles prennent un sens différent. Elle est formée de cinq lignes horizontales équidistantes.

On peut distinguer trois méthodes principales de détection des interlignes.

4.1.1 Interligne et Epaisseur de ligne

Les cinq lignes de portée ont toutes la même épaisseur et sont séparées verticalement de la même distance. Ainsi, lorsque l'on liste les pixels noirs contigüs (appelés empans) sur une colonne de l'image, le minimum va correspondre à une ligne. L'empan maximum de couleur blanche sera un interligne.

Cette méthode a été utilisée pour la première fois par Kato et Inokuchi sur une dizaine de colonnes à intervalles réguliers.

4.1.2 Projection horizontale

Si l'on considère que les lignes sont rectilignes et à peu près horizontales, cette méthode est extrêmement simple. Il suffit de faire la moyenne des points de chaque ligne de l'image. On obtient cinq pics représentants chacun une ligne de la portée.

Cette méthode ne fonctionne plus lorsqu'il y a une faible inclinaison d'un demi-interligne minimum. En effet, les pics fusionnent. Pour pallier le problème, Martin propose de trouver l'angle pour lequel les pics sont maximums. Il part à la recherche des plus grands segments, qui sont selon toute vraisemblance les lignes de portée, et calcule l'angle entre leurs deux points extrêmes.

Kato et Inokuchi réutilisent quant à eux les calculs de l'interligne et de l'épaisseur de ligne, en plus de la projection horizontale sur la gauche et la droite de l'image. Ainsi, ils récupèrent les positions extrêmes pour ensuite les relier par une droite.

Toutes ces méthodes ont pris pour postulat le fait que les lignes étaient rectiligne. Cependant, il n'est pas rare qu'il existe une faible courbure lors de la numérisation. Les solutions trouvées résident dans la recherche de parties de portée sans symboles. Une interpolation est ensuite effectuée à partir des résultats obtenus.

Le soucis avec ces méthodes, c'est leur caractère local. Plus le domaine de recherche est petit, plus les chances d'avoir des erreurs est importante. Elles ne sont pas non plus capables de traiter les portées très denses.

La dernière technique dont nous allons parler repose sur les filtre de Kalman. Le principe est simple, nous allons parcourir notre image en faisant des prédictions. Par exemple nous partons d'un segment horizontal, et nous cherchons un autre segment horizontal. Selon qu'il le trouve plus ou moins loin, avec un angle plus ou moins important, il va être gardé ou non.

En laissant l'algorithme tourner on obtient tous les segments horizontaux. Il est ensuite aisé de retrouver les lignes de portée. A noter qu'avec cette méthode, nous avons déjà un début d'information concernant les symboles.

4.2 Détection des Formes

4.2.1 Suppression des lignes de portée

La première étape consiste à supprimer les lignes de portée afin de d'éviter les interconnexions.

Les premiers systèmes supprimèrent tous les segments fins par érosion, ce qui a pour conséquence de séparer de nombreux symboles les rendant impossible à analyser.

Une autre approche consistait à supprimer une ligne d'épaisseur fixe, puis de relier tous les trous de tailles égaux à l'épaisseur. Malheuresement des symboles comme la clé de Fa n'y resistaient pas bien.

Ensuite, la tendance a plutôt évolué vers la suppression des zones sans symboles détectées grâce aux empans noirs et blancs mais également grâce au calcul de tangeantes.

Aucune de ces techniques ne produit de résultat parfait. Ainsi, une remise en cause de cette étape est obligatoire pour obtenir des formes cohérentes.

4.2.2 Extraction des primitives

Suppression progressive

Une façon simple de trouver les primitives consistes à effectuer une analyse de connexité. De la même façon que l'outil "Pot de Peinture" de Paint, pour chaque pixel on analyse ses voisins, et on avance tant que l'on trouve des noirs. Une fois l'élément trouvé, celui-ci est analysé et supprimé de l'image.

Une version un peu plus évoluée consiste à ordonner sa recherche. On peut d'abord chercher les barres verticales des notes, puis leurs rond etc...

L'inconvénient majeur de ces techniques est qu'elles sont linéaires. La moindre erreur va fausser la suite de la recherche.

Projection

Il est possible d'utiliser les projections afin de mettre en lumière des boîtes englobantes des primitives. Ces boîtes peuvent contenir plusieurs symboles.

La séparation des symboles fait intervenir plusieurs niveau d'abstractions, en passant de la détection pixel par pixel grâce à la reconnaissance de formes jusqu'à l'utilisation de la grammaire afin de déterminer les probabilités d'avoir tel ou tel élément.

Cette méthode pose un véritable problème : celui de confronter tous les niveaux d'abstraction différents en même temps. Les étapes ne sont pas clairement séparées.

Segmentation Complète

Carter a basé son étude sur la théorie des graphes des lignes adjacentes. On obtient grâce à cet algorithme une segmentation complète. Il est possible de reconstituer des primitives. Le point positif, c'est qu'elle est capable de détecter les lignes de portées. Cependant elle ne résoud en rien les fragmentations et regroupement de primitives.

4.3 Objectifs

De nombreuses pistes ont été explorées mais le manque de comparatif ne permet pas de se faire une idée précise des qualités et défauts de chacun d'entre eux. Ainsi, nous allons être obligés de tester différentes approches au cours du développement afin de prendre la méthode la plus adaptée.

Une notion qui n'a pas beaucoup été abordée dans la littérature est la façon de prendre des décisions. Certaines basent leur thèse sur une grammaire, d'autres voient plutôt des coefficients selon les situations. A nous de trouver notre propre moyen !

Chapitre 5

Analyse des Symboles

5.1 Problématique

Supposons que nous possédons une image en niveau de gris. Sa représentation interne n'est qu'une série de valeurs entre 0 et 255 sous forme matricielle.

Pour reprendre un exemple de Thierry Géraud lors de notre séminaire Recherche et Innovation, voici une image simple représentant un oeil :

FIGURE 5.1 – Un oeil élargi (19*15 à l'origine)

Mais voici ce que perçoit l'ordinateur sous le format brut des données !

```
169 163 162 172 189 187 177 173 184 165 161 167 167 166 170 ...
170 175 160 170 191 188 189 176 131 128 149 161 168 173 176 ...
185 180 153 173 193 200 174 113 097 106 106 104 113 138 166 ...
196 167 158 186 200 164 131 147 157 153 138 106 079 073 088 ...
188 157 180 200 146 118 132 146 126 131 164 133 117 104 086 ...
180 179 194 121 088 097 085 081 068 066 085 062 086 120 130 ...
189 190 104 066 066 055 056 052 049 057 070 047 039 066 129 ...
192 110 070 060 050 059 058 049 054 093 177 136 058 047 086 ...
124 092 076 051 060 097 078 073 070 085 197 212 147 088 107 ...
116 128 116 086 056 112 108 078 080 148 212 208 178 115 118 ...
129 136 149 133 098 102 130 130 150 193 202 196 162 142 129 ...
138 141 149 149 141 131 120 120 134 142 137 154 173 166 142 ...
148 149 152 149 148 151 139 131 133 137 135 153 158 157 144 ...
153 157 158 161 160 155 153 155 155 161 168 170 167 166 147 ...
151 157 163 170 168 174 170 167 171 176 173 171 171 163 143 ...
```

Il est clair que cette image est dénuée de sémantique, et il parait difficile d'apprendre à un programme la notion d'oeil sous cette forme. Qu'est ce qu'un oeil ? Ce serait une matrice présentant à peu près ces valeurs ?

Pour nous cela semble facile d'identifier une image, nous reconnaissons directement ce dont il s'agit, et pourtant le phénomène cognitif impliqué chez l'être humain est encore pire ! Nous ne recevons que des stimuli lumineux sur notre rétine, un amas de photons, et pourtant nous arrivons immédiatement à prendre une décision. Cependant une réaction très complexe a été mise en oeuvre au niveau de notre cerveau, dont nous essayerons d'imiter le fonctionnement dans le cas du réseau de neurone et de l'apprentissage décrit plus loin.

Nous allons d'abord simplifier le travail en tentant de dégager une sémantique purement structurelle de l'image, c'est à dire indépendante du contenu, afin de pouvoir comparer facilement plusieurs images pour pouvoir les classifier.

Nous allons donc envisager plusieurs méthodes pour caractériser une image. Notre recherche a été grandement guidée par les slides de Damien Lefortier utilisés dans sa conférence "Perceptrons multi-couches et OCR" pour la promotion 2011 le 7 février 2008.

5.2 Représentation matricielle

Pour cette méthode, on va comparer directement les valeurs des pixels dans la représentation matricielle de chaque symbole. Pour comparer un symbole inconnu par rapport à un symbole de référence on calcule la distance (en terme de nombre de différences) entre les deux matrices. On itère l'opération sur l'ensemble des symboles que nous connaissons et nous déterminons donc de quelle représentation notre symbole s'approche le plus.

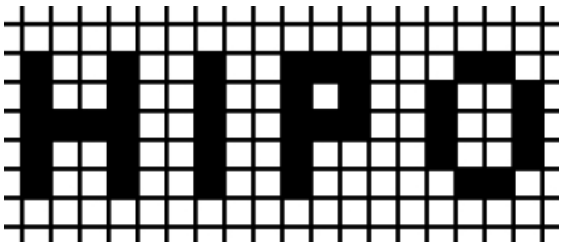


FIGURE 5.2 – Une représentation matricielle de plusieurs symboles

Cette méthode est facile à implémenter dans un programme, la fonction de distance entre deux matrices est tout simplement triviale. Les résultats obtenus sont généralement bons si l'on ne soumet pas des représentations trop ambiguës de certains symboles. Cependant la convergence est assez lente.

Le problème est que, bien évidemment, pour comparer deux matrices, elles doivent être de même dimension. Cet obstacle est non négligeable, en effet la partie du projet de séparation des symboles n'assure pas de rendre en sortie des symboles de mêmes tailles (surtout avec des symboles aussi variés en hauteur et en largeur comme pour les symboles des partitions de musique).

Un important travail de normalisation doit donc être effectué, pour être sur de manipuler des matrices de même dimension.

5.3 Transformation vectorielle

Contrairement à la méthode précédente qui utilisait une représentation matricielle d'une image, l'approche vectorielle consiste à modifier la représentation interne de l'image. Au lieu d'avoir la valeur de chaque pixel, l'image est composée d'objets géométriques individuels tels que des segments de droites, des polygones, des arcs de cercles, etc. Ces figures géométriques possèdent des attributs tels que leur position, leur taille, leur couleur. Cette transformation permet plus facilement la comparaison de symboles.

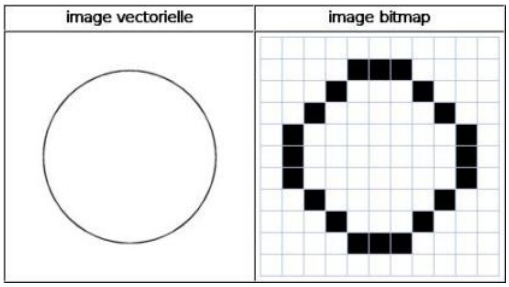


FIGURE 5.3 – La différence est nette

La première étape est de transformer notre image initiale sous format vectoriel. Pour cela on applique des algorithmes pour repérer les formes géométriques et en rendre compte dans un format de représentation vectorielle. Ceci demande des notions avancées de mathématiques comme les courbes paramétrées tels que les courbes de Bézier, mais aussi les polynômes de Lagrange, etc.

L'implémentation semble donc très difficile, cependant cette technique à le mérite d'obtenir de très bons résultats, ne nécessite pas de normalisation de taille contrairement à l'approche précédente (les images vectorielles sont redimensionnable à volonté sans perte de qualité), et enfin la reconnaissance de symboles très déformés est possible.

5.4 Moments de Zernike

Les polynômes de Zernike furent inventés en 1934 par le physicien Frederik Zernike. Ce sont des polynômes complexes formant une base orthogonale sur le cercle unitaire $x^2 + y^2 < 1$.
Leur formule générale en coordonnées polaire est la suivante :
$V_{mn}(r, \theta) \ = \ R_{mn}(r) \exp(jn\theta)$

On désigne par moment de Zernike une série de calculs utilisée pour transformer une image en un vecteur de composantes réelles représentant les moments géométriques A_{ij} de cette image.

Prenons une image que nous supposons de taille N*N, f(x,y) est alors la fonction décrivant le contenu de l'image en chacun de ses points de coordonnée x,y. On normalise la formule utilisée pour que x et y varient de -1 a +1.
La formule du (p+q)-ième moment géométrique de l'image est :
$m_{pq} \ = \ \sum_{x=-1}^{+1} \sum_{y=-1}^{+1} x^p y^q f(x, y)$

L'avantage des moments de Zernike est qu'ils sont invariants par rotation, translation, agrandissement et réduction. La base d'apprentissage n'a donc pas besoin d'être trop importante et aucun travail de normalisation n'est nécessaire avant traitement.

Les moments de Zernike sont exprimés avec la formule suivante :
$A_{mn} = \frac{m+1}{\pi} \int_x \int_y f(x, y)[V_{mn}(x, y)]^* \ dx \ dy$
Avec m, n des entier correspondant à l'ordre étudié et satisfaisant les contraintes suivantes : $m- \mid n \mid = \ pair \ \ , \ \ \mid n \mid \leq \ m$ et * dénote le conjugué complexe.

Cette méthode présente l'avantage d'être invariante à de nombreuses transformations ce qui donne de bons résultats, et l'intérêt mathématique est séduisant mais cela rend l'implémentation difficile.

5.5 Objectifs

Notre objectif est de développer une méthode efficace pour caractériser les données d'une image, le résultat obtenu servira ensuite d'entrée pour la propagation dans le réseau de neurones.

Les différentes techniques que nous avons présenté ne sont pas les seules, et il est trop tôt pour déterminer avec exactitude laquelle nous utiliserons, cela dépendra de celle qui nous semble la plus simple à mettre en place après de multiples recherches et tentatives.

Nous pouvons concilier plusieurs méthodes pour obtenir un résultat plus précis, en comparant les résultats avant de prendre une décision sur la nature d'un symbole.

Le planning n'est pas pleinement défini, mais nous allons vraisemblablement commencer par l'implémentation de l'approche matricielle, qui devrait fournir de bons résultats assez rapidement, ensuite nous essayerons une nouvelle technique, probablement celle des moments de Zernike qui présente de nombreux avantages, nous pourrions aussi envisager une approche bâtonnalle (que nous n'avons pas détaillée ici) qui obtient aussi de bons résultats et demande des recherches dans le champ des algorithmes génétiques.

Chapitre 6

Apprentissage

6.1 Présentation

Un réseau de neurone est un modèle de calcul basé sur un paradigme biologique, celui du neurone formel, c'est-à-dire la représentation mathématique du fonctionnement du neurone biologique.

Les neurologues McCulloch et Pitts menèrent les premiers travaux sur ce sujet en publiant leur article fondateur : *What the frog's eye tells to the frog's brain*. Ils proposent une première représentation mathématique d'un neurone : il s'agit d'un neurone binaire simple, dont la sortie ne peut être que 0 ou 1. Pour calculer cette sortie, le neurone effectue une somme pondérée de toutes ses entrées, c'est-à-dire soit 0 ou 1 puisque ce sont les sorties d'autres neurones binaires, puis applique une fonction dite d'activation : si la somme pondérée dépasse une certaine valeur dite seuil d'activation, la sortie du neurone est 1, sinon elle vaut 0.

L'objectif initial est de simuler le processus d'apprentissage de notre cerveau lorsqu'il est soumis à des stimuli inconnus, provenant d'un objet ou d'un phénomène encore jamais rencontré. Le principe utilisé ici est celui de l'induction, c'est-à-dire l'apprentissage par expérience sur un nombre de cas finis pour permettre une généralisation sur l'ensemble des cas possibles.

Grâce à leur capacité de classification et de généralisation (en se basant sur tous les exemples d'apprentissage), les réseaux de neurones sont souvent utilisés pour résoudre des problèmes statistiques : par exemple la prédiction des flux boursiers, la classification d'espèces animales selon leur ADN, la robotique, et dans le cas qui nous intéresse, pour la reconnaissance de motifs tels que des symboles, des objets, etc.

6.2 Analogie biologique

Le modèle de neurone formel a été construit en voulant imiter le fonctionnement du neurone biologique, il parait donc nécessaire d'en rappeler brièvement le fonctionnement.

Au repos, il existe une différence de potentiel négative, dit "potentiel de repos" entre la face interne de la membrane du neurone et sa face externe. Le neurone biologique reçoit l'information sous forme d'un potentiel d'action grâce à ses dendrites conductrices. L'influx nerveux est communiqué à d'autres neurones via un axone si et seulement si le signal en entrée dépasse un certain seuil dit seuil d'excitabilité du neurone.

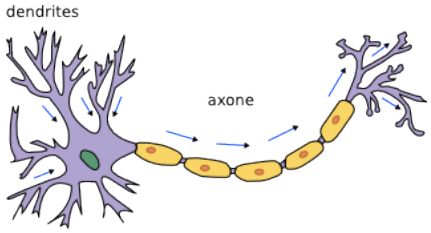


FIGURE 6.1 – Schéma d'un neurone biologique.

A l'image du neurone biologique, le réseau de neurone fonctionne comme support de la transmission de l'information.

Les neurones directement impliqués dans notre perception, par exemple ceux immédiatement connectés à notre rétine, seront modélisés par une couche d'entrée.

Le potentiel d'action conduit par les dendrites se transforme en booléen représentant si il y a une impulsion électrique ou non et dont l'amplitude sera modélisée par un poids synaptique.

La sortie, représentant l'axone, est égale à 1 si la somme des poids est strictement supérieure à une valeur seuil correspondant au seuil d'excitabilité.

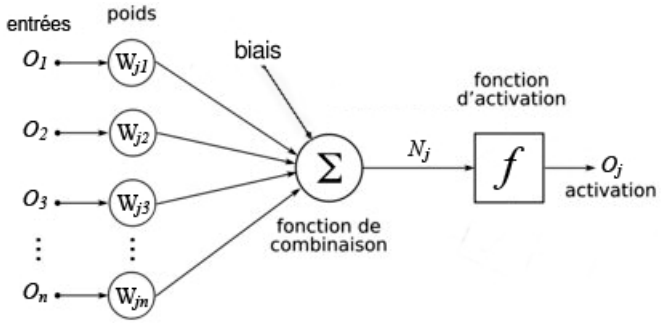


FIGURE 6.2 – Neurone formel.

6.3 Structure

Un réseau de neurone est composé d'une succession de couches dont chacune prend ses entrées sur les sorties de la précédente. Chaque couche est composée de N neurones, prenant leurs entrées sur les N' neurones de la couche précédente.

À chaque synapse est associée un poids synaptique, de sorte que les entrées des N' sont multipliés par ce poids, puis additionnés par les neurones du niveau courant.

Mettre l'une derrière l'autre les différentes couches d'un réseau de neurones reviendrait à calculer plusieurs matrices de transformation à la suite et pourrait se ramener plus simplement à une seule matrice, s'il n'y avait intervention de la fonction de sortie qui implique une non linéarité à chaque étape. Le choix de la fonction de sortie est d'une importance cruciale pour obtenir une bonne discrimination, un réseau de neurones dont les sorties seraient linéaires ne présenterait aucun intérêt.

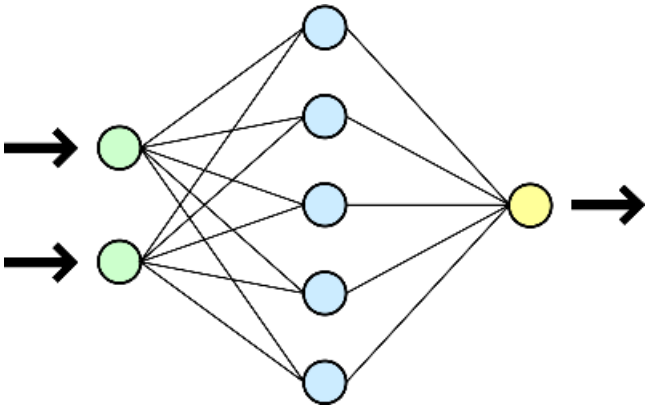


FIGURE 6.3 – Réseau de neurones simple.

6.4 Différents modes d'apprentissage

L'apprentissage regroupe deux aspects distincts et difficiles à concilier. Tout d'abord la mémorisation, c'est à dire être capable d'assimiler les résultats d'un grand nombre d'exemples. Ensuite, il est important que le réseau de neurones soit capable de généraliser, face à de nouveaux exemples inconnus mais similaires à des cas connus, il doit être capable de les traiter et de fournir une déduction correcte dans la plupart des cas.

L'apprentissage est supervisé lorsqu'on présente un motif en entrée à notre réseau et que l'on force à converger vers une sortie bien précise. Tant qu'il y a erreur, on utilise un algorithme pour corriger les poids synaptiques de chaque neurone pour tendre vers le résultat demandé.

Une simple analogie pourrait être de comparer l'apprentissage du réseau de neurones aux expériences vécues par un enfant en bas âge à qui l'ont montrerait une série d'objets verts pour lui faire assimiler la représentation de cette couleur.

Au contraire, si l'apprentissage est non-supervisé, le réseau est libre de converger vers n'importe quel état final lorsqu'on lui présente un motif. Dans le cas de l'apprentissage supervisé, un problème de surapprentissage peut survenir, si l'on force le réseau à répondre de manière quasi parfaite aux exemples qu'on lui présente (i.e si le seuil d'erreur accepté est très petit) alors que l'exemple est bruité ou imparfait. Le réseau pourra répondre incorrectement si on lui présente un autre exemple plus net, de meilleure qualité.

6.5 Différents types de réseaux de neurones

Les types de réseau de neurones diffèrent sur plusieurs aspects : la topologie des connexions entre les neurones et l'architecture générale du réseau.

Le **perceptron** est un modèle linéaire et monocouche, les sorties des neurones ne peuvent prendre que 3 états : 1, 0 ou -1. La règle en vigueur pour la modification des poids est la règle de Widrow-Hoff : si la sortie du réseau n'est pas égale à la sortie désirée, le poids est modifié proportionnellement à la différence entre la sortie obtenue et la sortie désirée. Les perceptrons ne peuvent pas simuler les comportements de fonctions non linéairement séparables (tel que le XOR).

Le **perceptron multicouche (PMC)** est une extension du modèle du perceptron. Dans le réseau multicouche, les neurones sont disposés en couches successives. Le réseau est représenté par 3 couches de neurones : une couche d'entrée symbolisant la rétine, une couche cachée car n'ayant pas de contact avec l'extérieur du réseau représente les neurones d'association et enfin une couche de sortie représente les neurones de décision.

Le **réseau de Hopfield** : Il s'agit d'un réseau constitué de neurones pouvant prendre deux états, la loi d'apprentissage est la règle de Hebb (1949), une synapse augmente son activité (i.e le poids synaptique) si l'activité de ses deux neurones est liée, c'est à dire si ils sont activés en même temps.

Le **réseau de Kohonen** : Cette modélisation se veut plus proche de la réalité. Ces réseaux sont inspirés des observations biologiques du fonctionnement des systèmes nerveux des mammifères. Une loi de Hebb modifiée est utilisée pour l'apprentissage. La connexion est renforcée dans le cas ou les neurones reliés ont une activité simultanée, et diminuée dans le cas contraire (à la différence du réseau de Hopfield). On utilise aussi parfois des lois de concurrence, ou algorithmes génétiques, pour modifier les neurones (création et destruction de neurones selon certains critères).

La fonction de combinaison peut aussi varier : chaque neurone reçoit depuis la couche précédente plusieurs valeurs via ses connexions synaptiques, et il transfère en aval une certaine valeur en utilisant une fonction de combinaison. Cette fonction de combinaison peut être donc n'importe quelle fonction prenant en paramètre un vecteur et renvoyant un scalaire. Les réseaux peuvent se diviser en deux grandes catégories selon la fonction de combinaison utilisée :

Les réseaux de type **MLP** (Multi-Layer Perceptron) calculent une combinaison linéaire des entrées, un simple produit scalaire entre le vecteur des entrées et le vecteur des poids synaptiques.

Les réseaux de type **RBF** (Radial Basis Function) calculent la distance entre les entrées, autrement dit, il s'agit de la norme euclidienne du vecteur issu de la différence vectorielle entre les vecteurs d'entrées. Plusieurs fonctions d'activations peuvent être utilisées, le choix de la fonction d'activation est important pour obtenir de bons résultats, mais son choix se fait généralement de manière empirique.

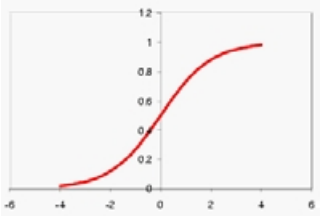


FIGURE 6.4 – Le sigmoïde

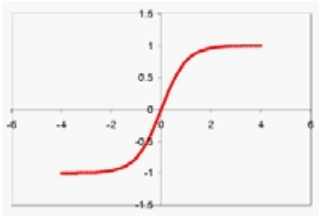


FIGURE 6.5 – La tangente hyperbolique

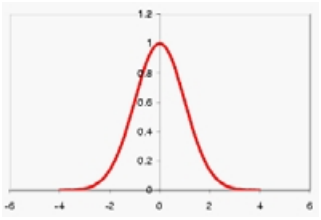


FIGURE 6.6 – La fonction Gaussienne

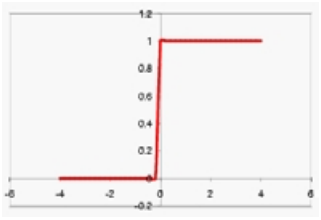


FIGURE 6.7 – Une fonction à seuil

6.6 Perceptron Multicouche

Nous avons choisi le modèle du perceptron multicouche pour l'apprentissage des motifs à reconnaître.

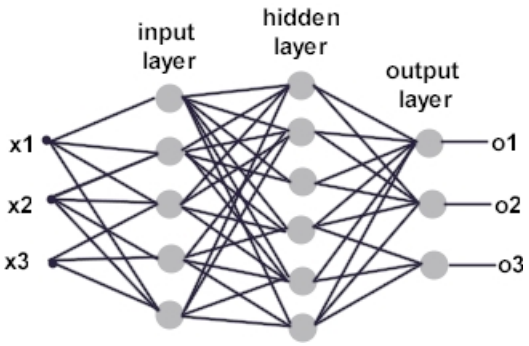


FIGURE 6.8 – Exemple de perceptron multicouche utilisant le sigmoïde comme fonction d'activation

L'apprentissage sera supervisé en utilisant l'algorithme classique de rétropropagation de l'erreur quadratique aussi appelé rétropropagation du gradient. Cet algorithme consiste à déterminer l'erreur commise par chaque neurone puis à modifier la valeur des poids pour minimiser cette erreur. Il faut

effectuer des rétropropagations jusqu'à ce que l'erreur quadratique moyenne devienne inférieure au seuil que l'on désirait, qui ne doit pas être trop petit pour éviter les problèmes de surapprentissage. En premier lieu, on effectue une propagation de l'entrée à travers le réseau par la méthode de calcul détaillée plus haut afin de déterminer l'erreur commise par chaque neurone de sortie. Ensuite l'algorithme consiste à remonter progressivement cette erreur depuis les sorties jusqu'à l'entrée en modifiant les poids au passage. Soit S le vecteur de sortie obtenu après la propagation de l'entrée X à travers le réseau, et Y le vecteur des sorties que l'on visait. Prenons un neurone appelé i sur la couche de sortie, l'erreur pour ce neurone est : $S[i] - Y[i]$.

L'erreur quadratique est, avec k neurones sur la sortie : $\sum_{i=1}^{k} (S[i] - Y[i])^2$

Soit p la taille de la couche cachée. Le neurone i a alors reçu les p sorties des neurones de la couche cachée modifiées par les poids caractéristiques. Ce sont des valeurs mal ajustées de ces poids qui ont induit ces erreurs, il faut donc les modifier. Pour relativiser l'importance d'un poids par rapport à un autre dans l'erreur totale il faut moduler sa modification par la sortie du neurone auquel il est rattaché. Il faut maintenant utiliser le gradient de l'erreur afin de pouvoir à terme faire converger la valeur du poids.

Le réseau à tendance à se conformer au dernier exemple présenté. C'est pourquoi on peut modifier la procédure d'apprentissage en présentant successivement tous les exemples avec une seule rétropropagation (par lot) à chaque fois et si l'erreur est supérieure au seuil au moins une fois au cours de ce balayage, ou bien en cumulant les erreurs, on refait un balayage complet. Cependant la convergence et sa vitesse sont assez chaotiques avec cette méthode.

6.7 Objectifs

Nous nous proposons donc de réaliser un perceptron multicouche pour la reconnaissance individuelle de nos notes de musique. Le travail sera fait en Objective Caml et la finalité recherchée sera de créer un module indépendant dont le travail sera double : tout d'abord permettre l'apprentissage sur une base d'exemples, et ensuite être capable de prendre en entrée un symbole sous forme d'image et de renvoyer l'inférence sur la nature de ce symbole. Le fonctionnement devra donc être celui d'une boite noire vis à vis du reste du programme.

Pour la **première soutenance**, une grande importance sera accordée à la recherche pour préciser au maximum la technique qui sera utilisée, et des expérimentations seront réalisées pour tester un peu l'implémentation de ces méthodes.

Pour la **seconde soutenance**, l'implémentation et les choix techniques se préciseront, une exigence minimum sur les résultats en termes de pourcentage de reconnaissance et de rejets pour encourager l'efficacité et la recherche de la solution la plus performante.

Enfin, lors de la **dernière soutenance**, le maximum devra être réalisé, il est bien sur évident que le travail d'optimisation et de recherche sur l'efficacité d'un réseau de neurone ne pourra jamais être terminé. Mais nous espérons tenir l'objectif minimum de 80% de reconnaissance sur les symboles inconnus.

Chapitre 7

Conclusion

Notre travail sur FOSMOR devrait nous apprendre beaucoup autant d'un point de vue algorithmique que sur la recherche. De plus, la reconnaissance de partitions de musique est un domaine où dans lequel beaucoup de pistes n'ont pas encore été explorées, ce qui rend le sujet intéressant et motivant de par son originalité.

D'une manière générale, nous pensons que la numérisation de documents anciens est un projet d'avenir, puisque certaines firmes comme les célèbres Google se sont déjà juré de numériser, par exemple, tous les livres. Et il parrait évident vu la situation actuelle que de plus en plus de médias pourront se retrouver directement par Internet. Ces medias devront donc, s'ils n'avaient pas été créés numériquement à la base, être numérisés afin d'intégrer le monde informatique.