Hes·so VALAIS WALLIS
Haute Ecole de Gestion
Hochschule für Wirtschaft

Filière Informatique de gestion
Studiengang Wirtschaftsinformatik

Travail de Bachelor 2024

Convergence du visuel et de l'audible par l'intelligence artificielle

Étudiant : Guillaume Rey

Professeur : Henning Müller

SOURCE DE L'IMAGE DE LA PAGE DE TITRE

L'illustration de la page de titre a été générée par Adobe Firefly 3.

RÉSUMÉ

La problématique centrale de ce projet interroge les possibilités d'utilisation de techniques de deep learning pour transformer des œuvres d'art visuelles (des peintures) en expériences sonores. L'objectif de ce travail est donc d'utiliser la technologie pour créer une nouvelle forme d'appréciation artistique. Cette question guide l'exploration des moyens par lesquels la technologie peut transcender les frontières entre les sens et rendre l'art plus accessible.

Après avoir étudié en détail la littérature existante dans les domaines de la sonification, de la synthèse sonore et de l'analyse d'image, nous avons entrepris le développement de divers prototypes permettant d'extraire des caractéristiques spécifiques à l'œuvre analysée. Sur la base de ces prototypes, des modèles définitifs ont été entraînés et évalués.

Par la suite, un patch de synthèse sonore a été créé. Celui-ci utilise les caractéristiques extraites lors de l'analyse de l'œuvre pour générer et enregistrer une composition musicale propre à celle-ci. Le processus de sonification final a ensuite été implémenté et automatisé.

Mots-clés : sonification, intelligence artificielle, machine learning, synthèse sonore, analyse d'images

AVANT-PROPOS

Ce travail de Bachelor, réalisé dans le cadre de la formation en Informatique de Gestion à la HES-SO Valais/Wallis, représente l'aboutissement de quatre années d'études, alliant théorie académique et pratique professionnelle.

Le projet présenté dans ce document est né d'une passion pour l'art visuel et la musique, deux formes d'expression artistique qui, bien qu'elles soient fondamentalement distinctes, partagent une capacité unique à évoquer des émotions, à stimuler l'imaginaire et à transcender les frontières culturelles. L'objectif de ce travail s'est orienté vers l'exploration des avancées en intelligence artificielle, pour déterminer comment celles-ci pouvaient être mises au service de l'art afin de créer de nouvelles expériences sensorielles.

Au cours de ce travail, l'objectif a été de développer un système capable de capturer l'essence visuelle d'une œuvre d'art et de la traduire en une composition sonore unique. Ce processus, connu sous le nom de sonification, repose sur l'analyse fine des caractéristiques visuelles de l'œuvre, telles que la couleur, la texture, le style et l'émotion dégagée, pour générer des éléments sonores reflétant le plus fidèlement possible l'esprit de l'œuvre originale.

La réalisation de ce projet a été un défi à la fois technique et artistique. Il a nécessité une compréhension approfondie des concepts théoriques en musique et en machine learning, ainsi qu'une sensibilité artistique pour assurer que les compositions générées ne soient pas seulement des transcriptions mécaniques, mais qu'elles possèdent, dans l'idéal, une véritable valeur esthétique.

REMERCIEMENTS

Je tiens à remercier toutes les personnes qui m'ont soutenu lors de la réalisation de ce travail. Je pense notamment à :

- Prof. Dr. Henning Müller, le professeur responsable de ce projet, pour son accompagnement, son soutien et ses conseils tout au long de la réalisation de ce projet.
- Mme Valérie Félix, responsable de la filière HEA de l'EDHEA, qui a pris de son temps pour répondre à mes questions, partager avec moi ses connaissances, et orienter mes recherches.
- M. Christophe Pignat et le service informatique de la HES-SO Valais/Wallis pour la mise à disposition de la machine virtuelle qui m'a permis de réaliser ce travail.
- À tous mes proches qui ont pris le temps de relire, de faire des suggestions et de corriger ce rapport.

TABLE DES MATIERES

TABLE DES FIGURES ..... **IX**

LISTE DES TABLEAUX ..... **XII**

LISTE DES ABBRÉVIATIONS ..... **XIII**

GLOSSAIRE ..... **XIV**

1. INTRODUCTION **1**

2. REVUE DE LA LITTÉRATURE **3**

2.1. SONIFICATION 3

2.2. ANALYSE D'IMAGE & MACHINE LEARNING 5

2.2.1. Analyse des émotions *6*

2.2.2. Analyse de la texture *12*

2.2.3. Analyse des couleurs *14*

2.2.4. Analyse du style *18*

2.3. SYNTHÈSE SONORE 20

3. MÉTHODES **24**

3.1. CONTRAINTES 24

3.2. CHOIX DES OUTILS POUR L'ANALYSE D'IMAGES 24

3.2.1. Analyse des émotions *24*

3.2.2. Analyse de la texture *26*

3.2.3. Analyse des couleurs *28*

3.2.4. Analyse du style *37*

3.3. CHOIX DE L'OUTIL DE SYNTHÈSE SONORE 38

3.3.1. Comparaison des outils *39*

3.4. CHOIX DE LA MÉTHODE D'ANNOTATION DES ŒUVRES (MODÈLE DE CLASSIFICATION DES ÉMOTIONS) 39

3.5. BIBLIOTHÈQUES & FRAMEWORKS 41

4. IMPLÉMENTATION & ÉVALUATION **43**

4.1. MÉTHODOLOGIE 43

4.2. ARCHITECTURE 43

4.3. DATASETS 46

4.3.1. Modèle d'analyse des émotions *46*

4.3.2. Modèle de nommage des couleurs *48*

4.3.3. Modèle de classification du style *48*

4.3.4. Test de l'application *48*

4.4. LEXIQUE 48

4.5. Analyse d'image 52
4.5.1. Texture 53
4.5.2. Couleur 58
4.5.3. Émotion 62
4.5.4. Style 66
4.6. Synthèse sonore 70
4.6.1. Extraction du contenu du fichier JSON & définition des variables 70
4.6.2. Définition des synthétiseurs 72
4.6.3. Styles artistiques 74
4.6.4. Enregistrement du fichier audio 75
4.7. Automatisation du pipeline 75
4.8. Développement du backend 77
4.9. Développement du frontend 79

5. RÉSULTATS 82

5.1. Modèle de classification des émotions 83
5.2. Modèle de nommage des couleurs 84
5.3. Modèle de reconnaissance du style 85

6. DISCUSSION DES RÉSULTATS 88

6.1. Analyse de l'œuvre 88
6.2. Synthèse sonore 95
6.3. Difficultés rencontrées 97
6.4. Améliorations et perspectives 98

7. UTILISATION DE L'IA 100

8. CONCLUSION 101

9. RÉFÉRENCES 103

10. ANNEXES 112

10.1. Choix de la méthodes d'analyse des émotions 112
10.2. Choix du logiciel de synthèse sonore 114
10.3. Choix de la méthode d'annotation des œuvres 115
10.4. Résultats de l'analyse d'œuvres 116
10.5. Interface utilisateur de la web app 133
10.6. Datasets utilisés 135
10.7. Product Backlog 136
10.8. Répartition des heures 137
10.9. Tâches des sprints 138

10.10. BURN DOWN CHARTS 144
10.11. DONNÉES DU TRAVAIL DE BACHELOR 147
10.12. PROJET GITHUB 150

DÉCLARATION DE L'AUTEUR 151

TABLE DES FIGURES

Figure 1 Echelles sonochromatiques pure et musicale 4
Figure 2 Jono, Des objets reconnus 5
Figure 3 Descripteurs de couleurs 7
Figure 4 Image initiale, repeinte & segmentée 8
Figure 5 Visualisation du clustering par KMeans 9
Figure 6 Non-Linear Matrix Completion (NLMC) 11
Figure 7 Support Vector Machine (SVM) 12
Figure 8 Exemples de caractéristiques de textures Tamura 14
Figure 9 Histogramme de couleurs 16
Figure 10 Exemple d'une image originale et de l'image masquée 17
Figure 11 Résultats - Modèle de Lafay & al. (2021) 19
Figure 12 Onde sinusoïdale et ses paramètres 20
Figure 13 Onde sinusoïdale à 440 Hertz 21
Figure 14 Onde en dents de scie à 440 Hertz 22
Figure 15 Cercle chromatique d'Itten 29
Figure 16 Scriabin color mapping 30
Figure 17 Evaluation des méthodes d'extraction de la couleur dominante 1 32
Figure 18 Evaluation des méthodes d'extraction de la couleur dominante 2 33
Figure 19 Comparaison des résultats 34
Figure 20 Exemples d'œuvres où la couleur dominante n'a pas été reconnue 35
Figure 21 Justesse du prototype sur 12 époques pour les valeurs d'entraînement et de validation 38
Figure 22 Structure du processus de sonification 44
Figure 23 Structure du serveur 45
Figure 24 Web App pour l'annotation des images pour l'analyse des émotions 47
Figure 25 Utilisation de certaines caractéristiques de texture 49
Figure 26 Chargement des modèles d'analyse d'image 52
Figure 27 Chargement de l'œuvre 53
Figure 28 Prétraitement de l'image pour l'analyse de texture 54
Figure 29 Calcul de la granularité (coarseness) 55
Figure 30 Types de kurtosis 56
Figure 31 Calcul du contraste 56
Figure 32 Calcul de la directionnalité 57
Figure 33 Calcul de la régularité 57
Figure 34 Calcul de la grossièreté 58
Figure 35 Plages de valeur des caractéristiques de texture 58

Figure 36 Prétraitement pour l'extraction de la couleur dominante 59
Figure 37 Calcul de la proportion de pixels noirs 59
Figure 38 Calcul du nombre de clusters optimal 60
Figure 39 Séparation des données en données d'entraînement et de test 61
Figure 40 Initialisation du classificateur Random Forest 61
Figure 41 Utilisation du modèle de nommage de couleurs 62
Figure 42 Extraction des valeurs RGB du jeu de données 63
Figure 43 Extraction de la palette de couleur 63
Figure 44 Segmentation de l'image par l'algorithme Felzenszwalb 64
Figure 45 Extraction des caractéristiques des segments 64
Figure 46 Préparation des données pour l'entraînement du modèle 65
Figure 47 Exemples de méthodes d'augmentation 65
Figure 48 Préparation de l'image et utilisation du modèle de classification d'émotions 66
Figure 49 Chargement des données et définition du nom des colonnes 67
Figure 50 Définition des augmentations de données 67
Figure 51 Chargement du modèle VGG16 et ajout des couches personnalisées 68
Figure 52 Compilation du modèle et définition des callbacks 68
Figure 53 Entraînement et sauvegarde du modèle de classification de style 69
Figure 54 Prétraitement de l'image pour le modèle de classification de style 69
Figure 55 Prédiction du style et conversion en chaîne de caractères 69
Figure 56 Mapping Couleur-Note MIDI 71
Figure 57 Définition des accords et de leur complexité 71
Figure 58 Choix du nombre d'accords et de leur complexité 72
Figure 59 Synthétiseur à onde en dents de scie 73
Figure 60 Synthétiseur de réverbération 73
Figure 61 Synthétiseur granulaire 74
Figure 62 Initialisation des variables pour le cubisme 74
Figure 63 Processus d'enregistrement du fichier audio 75
Figure 64 watch_images.sh 76
Figure 65 Extrait de watch_results.sh 76
Figure 66 Initialisation de l'app Flask & configuration des dossiers 77
Figure 67 Fonction de vérification du statut du fichier audio 77
Figure 68 Définition de la fonction "check_status" 78
Figure 69 Route de chargement d'une image 78
Figure 70 Web app - Page d'accueil 79
Figure 71 Style CSS pour le body et le html 79
Figure 72 Style CSS des boutons 80
Figure 73 Génération et affichage des images pour l'arrière-fond de la web app 80

Figure 74 Style CSS des images d'arrière-plan.....81
Figure 75 Calcul de la justesse d'un modèle.....82
Figure 76 Matrice de confusion pour une classification multi-classe 82
Figure 77 Matrice de confusion - Modèle de classification des émotions.....83
Figure 78 Matrice de confusion - Modèle de nommage des couleurs.....84
Figure 79 Justesse du modèle de détection du style.....85
Figure 80 Matrice de confusion - Modèle de détection du style 86
Figure 81 Résultats de l'analyse de l'œuvre « Swiss landscape with flowering apple tree » de Gustave Courbet, labellisée "réalisme" 88
Figure 82 Résultats de l'analyse d'une œuvre de Aki Kuroda, labellisée "peinture en champs de couleur".....89
Figure 83 Résultats de l'analyse de l'œuvre « A lane in Headingley, Leeds » de John Atkinson Grimshaw, labellisée "romantisme".....89
Figure 84 Analyse de l'émotion d'œuvres similaires 90
Figure 85 Diverses œuvres dans des styles variés 92
Figure 86 Résultats de l'analyse de l'œuvre ?? de ?? 93
Figure 87 Résultat de l'œuvre ?? de ?? 94
Figure 88 Echantillon de couleur dominante 94
Figure 89 Exemples d'œuvres pour lesquelles les compositions sont similaires 96
Figure 90 Exemples d'œuvres pour lesquelles les compositions reflètent l'ambiance de l'image .97

LISTE DES TABLEAUX

Tableau 1 Tableau comparatif des méthodes d'extraction d'émotions 26
Tableau 2 Comparatif des logiciels de synthèse sonore 39
Tableau 3 Comparatif des méthodes d'annotation des œuvres 41
Tableau 4 Correspondance entre les notes, les couleurs et les valeurs MIDI 50

LISTE DES ABBRÉVIATIONS

IA	Intelligence Artificielle
BoVW	Bag-of-Visual-Words
NLMC	Non-Linear Matrix Completion
SGL	Sparse Group Lasso
SVM	Support Vector Machine
RGB	Red Green Blue
BGR	Blue Green Red
PLSA	Probabilistic Latent Semantic Analysis
CNN	Convolutive Neural Network, ou Réseau de neurones convolutifs
ResNet	Réseaux neuronaux résiduels profonds
FM	Frequency Modulation
OS	Operating System
FFT	Fast Fourier Transform
GLCM	Gray Level Co-occurrence Matrix
HSV	Hue/Saturation/Value
LLM	Large Language Model
CSS	Cascading Style Sheets
HTML	HyperText Markup Language

GLOSSAIRE

Ce glossaire vise à préciser certains termes qui ne sont pas traduits ou définis dans le rapport.

Clustering / cluster : le clustering, dans le domaine du machine learning, est une discipline visant à séparer des données partageant des caractéristiques communes en groupes homogènes, appelés « clusters » (Kassel, 2020).

Frontend : le terme « frontend », en développement web, fait référence aux éléments visibles d'un site ou d'une application web (interface utilisateur) (Front-End, n.d.).

Backend : a contrario du frontend, le backend réunit tous les composants qui assurent le bon fonctionnement d'une application ou d'un logiciel mais qui sont invisibles pour l'utilisateur (accès aux données) (Back-End Définition, n.d.).

Pipeline : un pipeline peut être définie comme un ensemble d'outils ou de processus automatisés, où chaque étape utilise, en entrée, la sortie de l'étape précédente (Pipeline : qu'est-ce que c'est ?, n.d.). Dans le cadre de ce travail, il s'agit du processus complet de sonification, de l'analyse de l'œuvre à l'enregistrement de la composition sonore.

Callback : en machine learning, un callback est une fonction qui automatise certaines tâches durant l'entraînement d'un modèle, pour interrompre automatiquement l'entraînement lorsque la performance cesse de s'améliorer par exemple (Pramod, 2024).

Bin : dans le contexte des histogrammes, une « bin » est un intervalle qui regroupe une plage de valeurs (matplotlib.pyplot.hist, n.d.).

1. Introduction

L'interrelation entre l'art et la technologie est un phénomène séculaire. L'évolution technologique a souvent influencé, voire redéfini les courants artistiques. La guitare électrique a permis la naissance du rock'n'roll, l'invention du synthétiseur a rendu possible la création de sonorités innovantes et de façonner de nombreux styles musicaux, et le hip-hop ne serait pas ce qu'il est aujourd'hui sans la création du « sampler ». Plus récemment, l'informatique a rendu accessible la composition et la production musicale à un large public. L'avènement de l'intelligence artificielle (IA) a accentué cette tendance, en révolutionnant le processus créatif jusqu'à remettre en question le rôle même de l'humain dans la création artistique.

Un phénomène analogue s'observe dans les arts visuels. Longtemps cantonnés à la peinture et à la sculpture, les artistes ont vu leurs moyens d'expression s'élargir avec l'invention de la photographie. Par la suite, les logiciels de retouche d'image ont offert des possibilités inédites de modification, de superposition et de réarrangement des œuvres. Depuis plusieurs années, il est possible de créer une œuvre entièrement sur ordinateur, et depuis peu, on peut la générer automatiquement par intelligence artificielle.

Chaque avancée technologique ouvre un champ quasi infini de nouvelles possibilités, qui ne manquent pas d'être explorées par des artistes curieux et novateurs. Toutefois, chaque mutation majeure suscite souvent des critiques de la part des tenants du statu quo. Ces critiques peuvent prendre diverses formes et entraîner des conséquences variées. Dans les années 1990, les « beatmakers » (terme utilisé pour nommer le compositeur d'un morceau hip-hop), faute d'accès aux studios et aux musiciens renommés, ont commencé à utiliser des « samplers » pour enregistrer et réutiliser des échantillons de morceaux existants pour créer leurs instrumentaux, ce qui a provoqué l'ire des détenteurs des droits sur les œuvres originales, entraînant de nombreuses plaintes et la nécessité de réviser la législation relative aux droits d'auteur. Aujourd'hui, l'essor de l'intelligence artificielle suscite à nouveau un débat éthique et moral sur la légitimité des œuvres générées par IA.

Motivations

Bien que l'innovation technologique puisse être accueillie par des critiques parfois sévères, elle tend généralement à élargir le champ des possibles, en générant des retombées positives sur le domaine concerné. C'est cet aspect, celui de l'élargissement des horizons artistiques grâce à la technologie, qui suscite mon intérêt et motive ma réflexion sur la transformation de l'expérience artistique et les possibilités qui s'y cachent.

Très intéressé par les mondes du visuel et de l'audio, je me suis questionné sur les relations concrètes qui peuvent être tissées entre ces deux domaines, et plus précisément sur les

caractéristiques qu'une œuvre d'art peut partager avec une composition musicale. Cette base de réflexion a dessiné les grandes lignes de ce travail dans lequel il sera question de sonification. Le terme « œuvre d'art » est extrêmement vague et peut qualifier une peinture, une sculpture, une installation, etc. Dans ce travail, nous nous concentrerons sur la peinture, et nous nous limiterons à certains styles. De ces œuvres, certaines caractéristiques seront extraites et ces dernières seront utilisées pour générer et moduler divers attributs musicaux, créant ainsi une composition unique à chaque œuvre sonifiée.

Une approche artistique a été choisie pour ce travail, l'objectif est donc d'obtenir des résultats qui sont esthétiques et musicaux. Cette démarche ne permet pas de générer une couche sonore qui permet d'interpréter une œuvre avec précision, mais vise à rendre possible l'évaluation des propriétés visuelles essentielles à la compréhension d'une œuvre, et lesquelles parmi elles peuvent être convertie en propriété sonores.

Structure

Nous commencerons ce travail de recherche par l'étude de la littérature des domaines concernés par ce projet : la sonification, l'analyse d'image et la synthèse sonore. Cela nous permettra d'évaluer les diverses approches existantes, leurs avantages et inconvénients ainsi que leur validité dans le cadre de ce travail.

Nous discuterons dans un second temps des méthodes qui seront utilisées pour la réalisation du processus de sonification, de l'analyse des caractéristiques de l'œuvre à la génération de la composition sonore. Diverses approches seront comparées et jugées pour déterminer lesquelles sont les plus adaptées.

Une fois ces méthodes définies, nous passerons à l'implémentation des divers outils qui permettront l'analyse d'une œuvre, l'extraction de ses caractéristiques et la génération d'une composition sonore. Nous définirons également le lexique de sonification et automatiserons le processus.

Les résultats seront finalement analysés et discutés dans la dernière partie de ce rapport.

2. Revue de la littérature

Dans cette revue de la littérature, nous nous intéresserons tout d'abord à la sonification, nous nous concentrerons ensuite sur l'analyse d'images et terminerons ce chapitre en se penchant sur la synthèse sonore.

2.1. Sonification

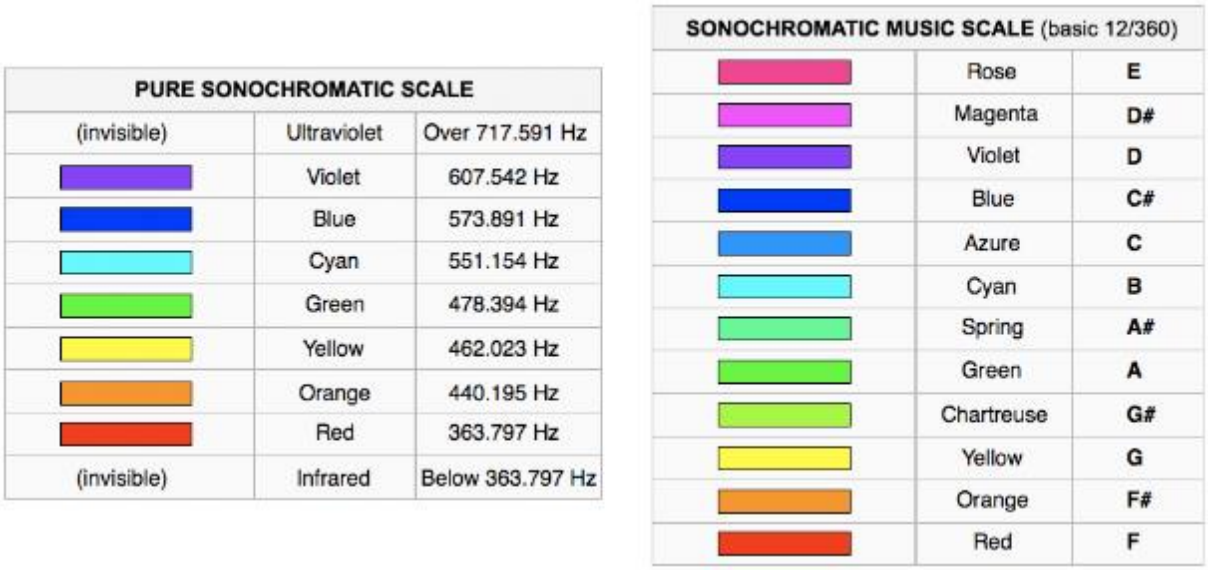
La sonification, définie comme l'utilisation de l'audio non-verbal pour transmettre des informations, représente une intersection fascinante entre la science, l'ingénierie et les arts. Elle vise à transformer des données (temporelles, multidimensionnelles, spatiales, etc.) en signaux acoustiques afin d'en faciliter l'interprétation (Kramer, et al., 2010). L'évolution de ce domaine a été soutenue par des développements technologiques rapides et par une reconnaissance croissante de son potentiel pour diverses applications scientifiques et pratiques. Historiquement, certains dispositifs comme le compteur Geiger, inventé au début du 20ème siècle, peuvent être considérés comme des exemples précoces de sonification. Le compteur Geiger utilise des clics pour indiquer les niveaux de radiation, exploitant la capacité humaine à détecter des changements auditifs subtils mieux que des variations visuelles (Kramer, et al., 2010).

Le développement théorique de la sonification a souligné l'importance de comprendre la perception auditive. Les recherches dans ce domaine ont exploré comment les caractéristiques acoustiques telles que l'intensité, la fréquence, et la localisation influencent la capacité des auditeurs à extraire des informations des signaux sonores (Gresham-Lancaster, 2012). Avec le temps, divers outils et techniques ont été développés pour faciliter la sonification. Parmi eux, des logiciels informatiques comme CSound, SuperCollider, Pure Data et Max/MSP. Ceux-ci sont couramment utilisés pour la synthèse sonore et le contrôle de paramètres acoustiques et permettent une flexibilité et une précision dans la création de représentations sonores de données complexes (Kramer, et al., 2010). La synthèse sonore peut être réalisée par des méthodes comme la synthèse additive, la modulation de fréquence ou la modélisation physique des objets sonores (Kramer, et al., 2010). Ces techniques seront discutées plus en détail dans le chapitre portant sur la synthèse sonore.

Les applications de sonification sont vastes et variées, allant de la surveillance de processus industriels à l'assistance pour les personnes malvoyantes en passant par les créations artistiques. Dans le contexte de ce projet, l'approche artistique est celle qui nous intéresse le plus et c'est sur elle que nous allons nous pencher dans les paragraphes qui suivent.

Neil Harbisson est un artiste britannique qui, en 2004, est devenu la première personne à se faire implanter une antenne dans la tête. La particularité d'Harbisson est qu'il est atteint d'achromatopsie de naissance, une maladie qui ne lui permet pas de voir en couleur (Neil Harbisson, 2024). Pour pallier

cette limitation, il collabore avec Adam Montandon et met au point une antenne équipée d'une caméra qui capte les couleurs et les transforme en ondes sonores (Neil Harbisson, 2024). Pour mettre en relation les couleurs et les fréquences sonores (ou les notes), Harbisson conçoit l'échelle sonochromatique pure et l'échelle sonochromatique musicale basées sur la roue des couleurs (Sonochromatism, 2023).



PURE SONOCHROMATIC SCALE		
(invisible)	Ultraviolet	Over 717.591 Hz
	Violet	607.542 Hz
	Blue	573.891 Hz
	Cyan	551.154 Hz
	Green	478.394 Hz
	Yellow	462.023 Hz
	Orange	440.195 Hz
	Red	363.797 Hz
(invisible)	Infrared	Below 363.797 Hz

SONOCHROMATIC MUSIC SCALE (basic 12/360)		
	Rose	E
	Magenta	D#
	Violet	D
	Blue	C#
	Azure	C
	Cyan	B
	Spring	A#
	Green	A
	Chartreuse	G#
	Yellow	G
	Orange	F#
	Red	F

Figure 1 Echelles sonochromatiques pure et musicale

Source : https://en.wikipedia.org/wiki/Sonochromatism

Un processus de sonification très différent apparaît dans le travail de l'architecte et compositeur franco-grec Iannis Xenakis. Dans les années 1950, il utilise des calculs de probabilités pour composer des œuvres comme « Metastasis et Pithoprakta » (Serra M.-H. , 1993). Xenakis invente alors la musique stochastique qui repose entièrement sur l'application de lois de probabilités et sur des procédures mathématiques pour déterminer les aspects de la composition (Musique, 2024) (Serra M.-H. , 1993). Pour cette application, il a créé un algorithme de synthèse sonore appelé la « synthèse stochastique dynamique » qui permet de générer une variété de timbres et de sons riches (Serra M.-H. , 1993).

Enfin, l'œuvre « Des objets reconnus » de l'artiste Jono, bien qu'elle se concentre principalement sur la reconnaissance d'objets dans des peintures à l'huile, utilise également les couleurs prédominantes d'une œuvre pour générer une couche sonore (Jono, 2019). Jono ne détaille pas les techniques utilisées pour convertir les couleurs en ondes sonores, mais son approche de la sonification est intéressante par le lien qu'elle tisse entre les couleurs d'une peinture et la couche sonore qui en émane. Ce même lien est au cœur de ce travail et sera discuté plus en détail dans les chapitres suivants.

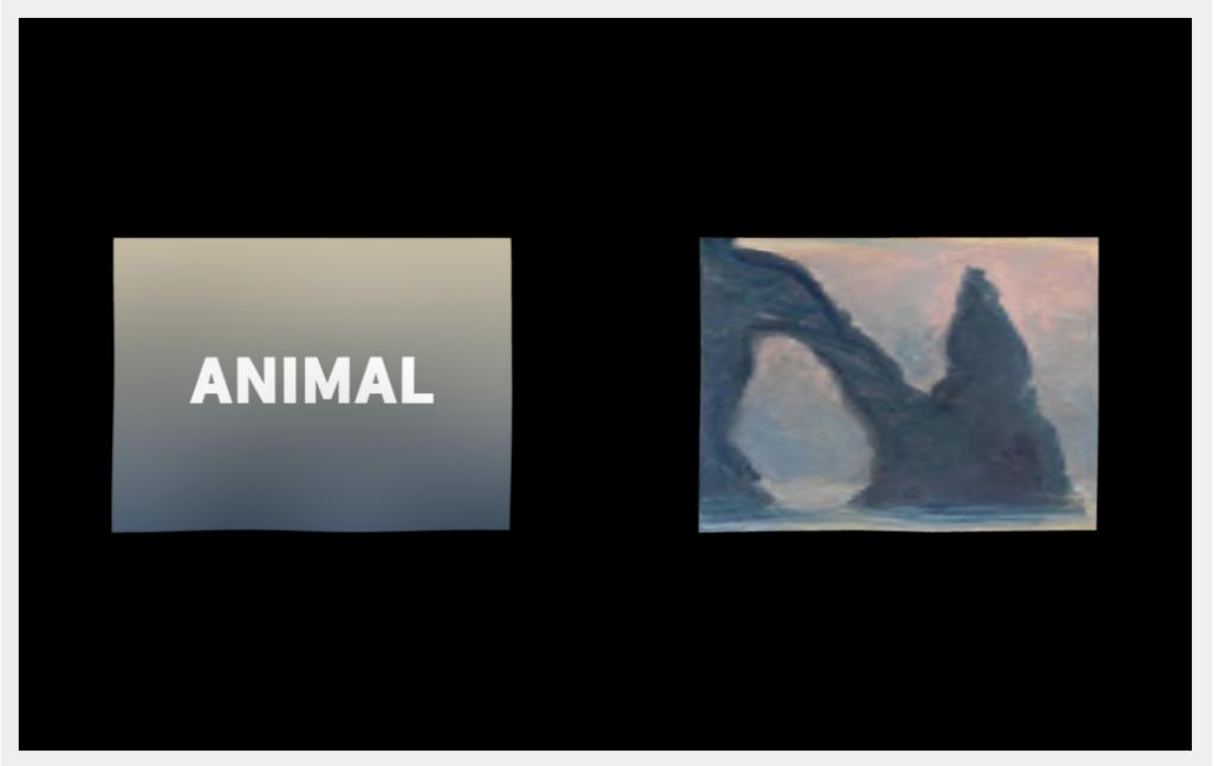


Figure 2 Jono, Des objets reconnus

Source : https://www.jono.fyi/des-Objets-Reconnus

2.2. Analyse d'image & machine learning

L'analyse d'images en machine learning est un domaine crucial qui combine la puissance des algorithmes et des modèles statistiques pour interpréter, comprendre et prendre des décisions basées sur des données visuelles. Cette technologie est devenue de plus en plus influente dans divers secteurs comme la santé, la sécurité, le divertissement et le commerce de détail, en raison de sa capacité à traiter rapidement et précisément d'énormes quantités de données d'images.

Au cœur de l'analyse se trouve l'extraction d'informations significatives à partir des images. Les modèles d'apprentissage automatique, en particulier les techniques de deep learning, sont formés sur de grands ensembles de données d'images pour apprendre des motifs, des caractéristiques et des représentations qui leur permettent d'accomplir des tâches telles que la détection d'objets, la classification, la segmentation ou l'amélioration des images (McKee, 2023).

L'un des principaux progrès dans l'analyse d'images a été le développement des réseaux de neurones convolutifs (CNN). Les CNN ont révolutionné le domaine en offrant des performances de pointe dans la reconnaissance faciale, l'analyse d'images médicales ou encore la conduite autonome (Image Processing Using Machine Learning, n.d.).

Les applications spécifiques de l'analyse d'images en apprentissage automatique sont variées et en constante évolution (McKee, 2023). Dans ce chapitre, nous ne nous concentrerons que sur celles qui nous intéressent dans le cadre de ce travail de recherche, à savoir l'analyse de l'émotion dominante, de la texture, du style et des couleurs d'une image.

2.2.1. Analyse des émotions

Bien que la majorité des recherches se concentrent sur la reconnaissance des émotions à partir d'expressions faciales humaines, certains travaux se sont intéressés aux effets que les couleurs peuvent avoir sur la perception des émotions dans une œuvre d'art.

La couleur est intrinsèque à une image et suscite l'intérêt de nombreux domaines de recherche comme la physique, la psychologie, les neurosciences et l'art. Ces disciplines se sont penchées sur la nature de la couleur, la manière dont elle est perçue et les influences qu'elle peut exercer (Sartori, Culibrk, Yan, & Sebe, 2015a). Bien que la perception des couleurs puisse sembler très subjective, des études font ressortir une certaine homogénéité dans la manière dont elles sont perçues : les couleurs claires évoquent généralement des sensations plus agréables et moins dominantes que les couleurs sombres (Valdez & Mehrabian, 1994).

Au-delà des couleurs elles-mêmes, le contexte et les relations entre les couleurs jouent un rôle crucial dans la perception humaine. Il est difficile de définir une couleur sans la comparer à une autre, et cet aspect est essentiel pour classifier les émotions : le positionnement et le mélange de certaines couleurs peuvent engendrer des réactions émotionnelles variées (Sartori & al., 2015a). Ou, Luo, Woodcock, & Wright (2004) montrent que sur cet aspect-là également, une certaine uniformité dans la perception des combinaisons de couleurs est retrouvée, avec toutefois des variations entre les genres.

De nombreux articles de recherche se concentrent sur l'analyse des émotions dans l'art abstrait, car ce dernier, par sa simplicité, évoque des émotions uniquement à travers des éléments basiques tels que les couleurs, les formes et les textures (Sartori, Yanulevskaya, Akdag Salah, Uijling, & Bruni, 2015b). Les paragraphes suivants détaillent les étapes généralement adoptées pour la détection des émotions dans une œuvre, ainsi que les approches méthodologiques utilisées dans les articles étudiés.

Extraction des caractéristiques visuelles

La couleur, la forme et la texture sont essentielles pour capturer les éléments qui peuvent influencer l'émotion perçue par l'observateur. Cette première étape nécessite donc l'extraction de certains attributs visuels. Pour ce faire, des descripteurs de couleurs sont généralement utilisés. Ce sont des outils mathématiques qui permettent de représenter numériquement et de quantifier les caractéristiques de couleur d'une image (Manjunath, Ohm, V. Vasudevan, & Yamada, 2001). Ils aident

donc à capturer certains éléments spécifiques aux couleurs d'une image. La figure 3 montre l'existence d'un grand nombre de descripteurs généralement classifiables en trois catégories : les méthodes basées sur les histogrammes, les méthodes statistiques et les descripteurs MPEG-7 (Chavda & M Mahesh, 2019). Nous n'allons pas tous les étudier dans ce travail, nous nous concentrerons sur quelques approches utilisées dans des travaux portant sur l'analyse des émotions.

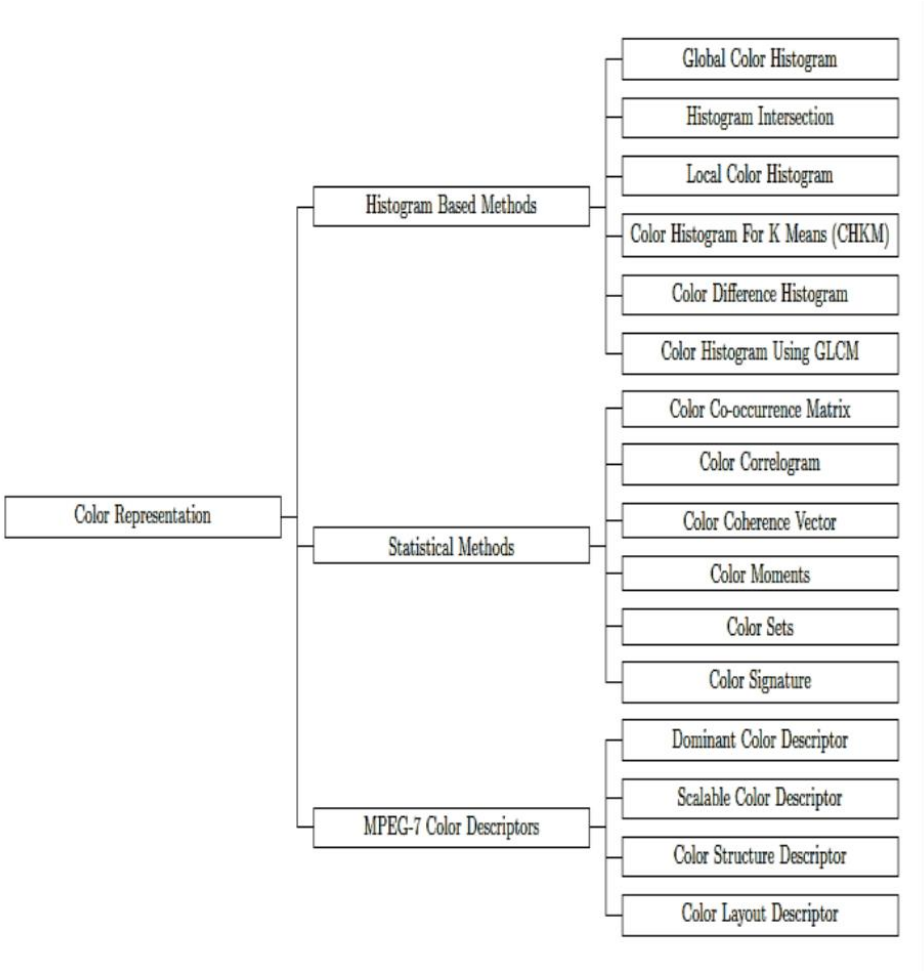


Figure 3 Descripteurs de couleurs

Source : Chavda & M Mahesh, 2019

Dans son livre « The Art of Color : The subjective experience and objective rationale of color », le peintre suisse Johannes Itten met au point un cercle chromatique à 12 teintes. Cette roue chromatique est centrée sur les couleurs primaires, puis secondaires, et enfin tertiaires (obtenues en associant une couleur primaire à une couleur secondaire) (Itten, 1961). Alameda-Pineda, Ricci, Yan, & Sebe (2016) et Sartori & al. (2015a), utilisent dans leur approche des descripteurs de couleurs dérivés de cette théorie. Cette approche permet d'analyser de manière détaillée les combinaisons de couleurs, et les relations entre celles-ci, et de comprendre comment différentes teintes et mélanges peuvent influencer les émotions perçues.

Sartori & al. (2015b) et Yanulevskaya, Uijlings, Bruni, & Sartori (2012), utilisent quant à eux l'espace colorimétrique LAB (ainsi que des descripteurs de texture SIFT) pour représenter leurs peintures. LAB est un espace tridimensionnel défini par la luminance (L), l'axe rouge-vert qui mesure les variations entre ces couleurs (A), et l'axe jaune-bleu qui mesure les variations entre ces autres couleurs (B), qui a été conçu pour correspondre à la façon dont la couleur est perçue par l'œil humain (Sartori & al.,2015b).

Segmentation des images

Dans un deuxième temps, les images sont communément segmentées en régions distinctes. Cette segmentation est cruciale car elle permet d'isoler les éléments visuels significatifs et d'analyser leur impact émotionnel de manière plus précise. À nouveau, plusieurs techniques de segmentation se montrent efficaces pour l'analyse d'émotions.

Sartori & al. (2015a) proposent une approche en deux étapes. Dans un premier temps, une palette de couleurs est extraite du dataset d'entraînement. L'algorithme KMeans est ensuite utilisé pour repeindre l'image avec les couleurs de la palette. Cet algorithme utilise la distance euclidienne, soit la longueur du segment qui sépare deux éléments, pour faire correspondre les couleurs de l'image à celles de la palette (Sartori & al., 2015a) (Distance euclidienne, 2024). L'image est ensuite segmentée en parcelles de couleurs homogènes. La méthode de segmentation proposée par Felzenszwalb et Huttenlocher est utilisée (Sartori & al., 2015a). Cette approche permet de préserver les détails dans les régions à faible variabilité tout en ignorant les détails dans les régions où la variabilité est importante, pour ne préserver que les informations essentielles.

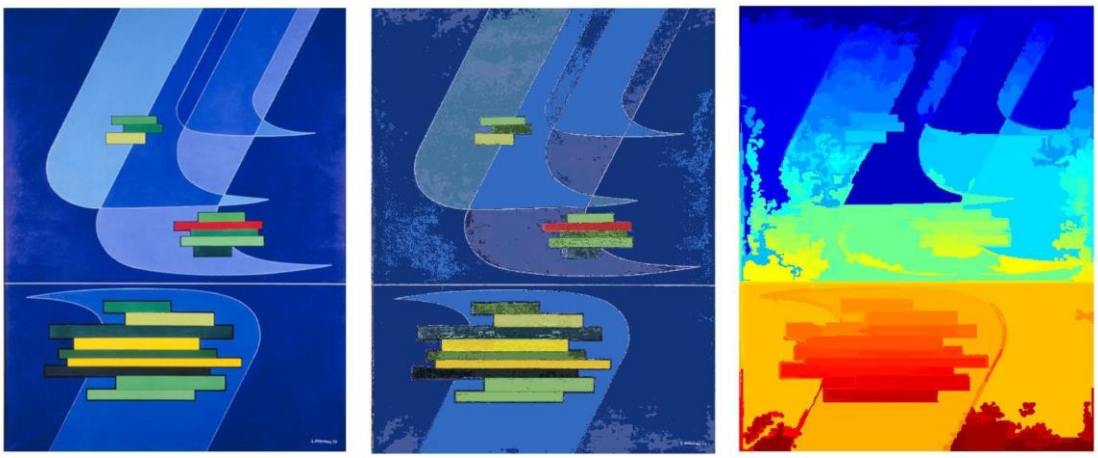


Figure 4 Image initiale, repeinte & segmentée

Source : Sartori & al., 2015a

La segmentation sur laquelle s'appuient Yanulevskaya & al. (2012), consiste quant à elle à diviser l'image en patches auxquels un mot visuel est attaché selon le modèle Bag-of-Visual-Words (BoVW).

Un « vocabulaire visuel » est créé et utilisé en attribuant des mots visuels aux différentes régions de l'image (Yanulevskaya & al., 2012). Sartori & al. (2015b) décrivent un mot visuel comme un motif spécifique d'une image (par exemple, bande jaune ou carré bleu sur fond blanc). Ce procédé nécessite donc l'utilisation d'un large corpus d'images échantillonnées. À nouveau, le clustering est réalisé grâce à l'algorithme KMeans, et ici, ce sont les mots visuels qui représentent les centres des clusters (Yanulevskaya & al., 2012). Cette méthode de segmentation des images en patches et l'utilisation de mots visuels permettent d'extraire des informations visuelles pertinentes et surtout très précises pour l'analyse des émotions.

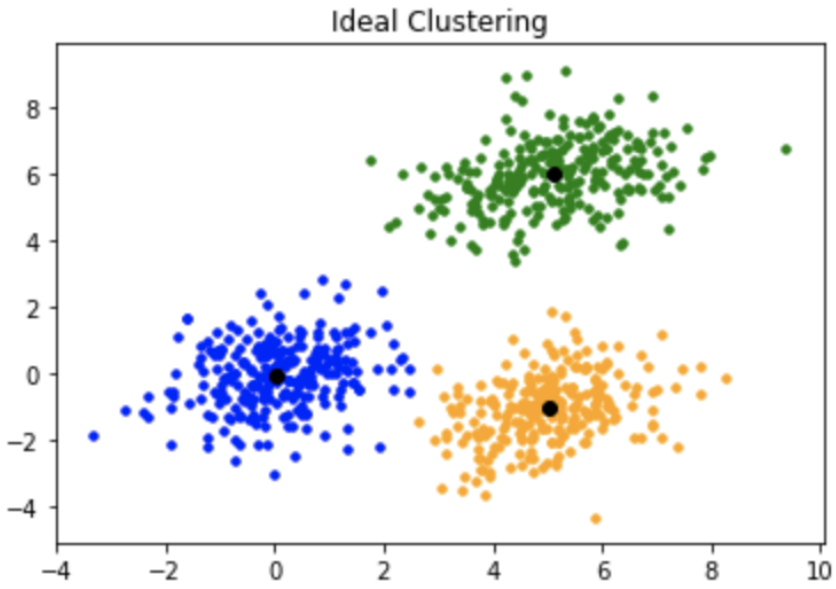


Figure 5 Visualisation du clustering par KMeans

Source : https://commons.wikimedia.org/wiki/File:K_Means_Example_Step_4.svg

Annotation des données émotionnelles

L'annotation des données émotionnelles consiste à obtenir des annotations d'émotion pour les œuvres qui serviront de données d'entraînement et d'évaluation aux modèles de machine learning.

Pour collecter ces précieuses annotations, la méthode « TrueSkill » est utilisée par Alameda-Pineda & al. (2016) et Sartori & al. (2015a). Ce système mis au point par Microsoft Research consiste à demander aux participants de comparer des paires de peintures et de choisir celle qui évoque l'émotion la plus positive (Sartori & al., 2015a). Il permet d'obtenir des données précises tout en réduisant le nombre de comparaisons faites par chaque participant (Sartori & al., 2015a). Cela est dû au fait que « TrueSkill » attribue initialement une compétence (nommée « mu ») et une incertitude moyenne (appelée « sigma ») à toutes les œuvres, et met à jour ces valeurs au fur et à mesure des

comparaisons (Minca, Zaykov, & Tims, 2005). Ainsi, des valeurs fiables peuvent être extraites sans que chaque œuvre n'ait besoin d'être comparée à toutes les autres.

Yanulevskaya & al. (2012) utilisent l'échelle de Likert, où les participants évaluent les émotions suscitées par chaque peinture selon une échelle prédéfinie (1 – émotion extrêmement négative et 7 – émotion extrêmement positive). Cette méthode permet de quantifier les émotions de manière fine, bien que les réponses puissent être influencées par des biais de réponse, comme la tendance à éviter les extrêmes (biais de réponse neutre) (Comment éviter les biais de sondage, n.d.).

Une approche basée sur les descriptions textuelles fournies par les participants est également utilisée dans certains cas. Ces descriptions textuelles offrent une richesse de données qualitatives et permettent de capturer des nuances émotionnelles complexes qui peuvent être difficiles à quantifier par des méthodes numériques (Sartori & al., 2015b).

Modélisation et apprentissage

Les attributs visuels et les annotations émotionnelles sont utilisés pour entraîner un modèle de machine learning, l'objectif étant bien entendu de prédire les émotions suscitées par de nouvelles œuvres.

La dépendance aux annotations peut être un obstacle dans l'utilisation d'un modèle. Certaines méthodes permettent toutefois de pallier cette limitation. La complétion de matrice, par exemple, est utilisée pour combler les valeurs manquantes dans une matrice partiellement observée (Matrix completion, 2024). Alameda-Pineda & al. (2016) étendent les méthodes de complétion de matrice pour gérer des relations non-linéaires et mettent au point la complétion de matrice non-linéaire (NLMC). Ils exploitent les relations entre les différentes étiquettes émotionnelles et les caractéristiques visuelles pour améliorer la précision des prédictions grâce à un système robuste face à des annotations incomplètes.

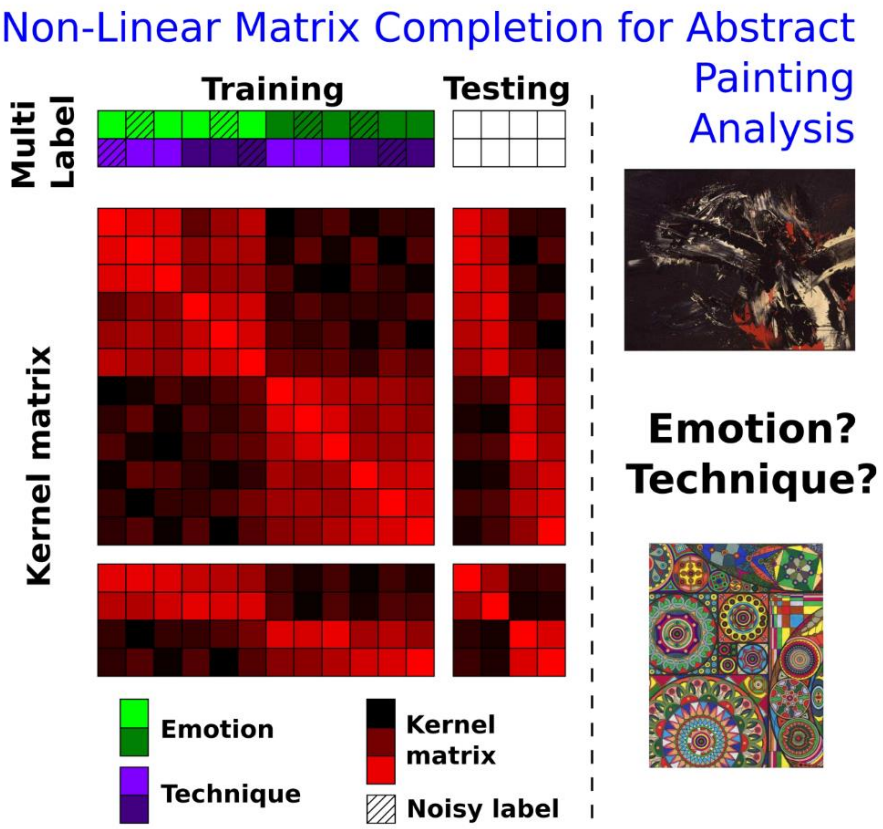


Figure 6 Non-Linear Matrix Completion (NLMC)

Source : Alameda-Pineda & al., 2016

Sartori & al. (2015) suggèrent dans leur publication, une approche également très efficace, dépendante d'un jeu de données parfaitement annoté : le Sparse Group Lasso (SGL). Le SGL consiste à regrouper certaines caractéristiques en différents groupes qui peuvent se chevaucher (Rao, Nowak, Cox, & Rogers, 2015). Par exemple, dans le contexte de l'étude des émotions, les couleurs très semblables (rouge et orange) seront regroupées alors que les couleurs très distinctes (rouge et bleu) seront dans des groupes séparés. L'équation mathématique utilisée prend en compte la différence entre la prédiction du modèle et les émotions réelles, ainsi que le poids des couleurs au sein d'un même groupe (Sartori & al., 2015a).

Une troisième méthode d'apprentissage utilisée pour l'extraction d'émotions est le Support Vector Machine (SVM). Elle fonctionne en trouvant l'hyperplan qui sépare les données de différentes classes avec la plus grande marge possible. La figure 7 illustre l'hyperplan optimal et la marge maximale dans un espace bidimensionnel. Yanulevskaya & al. (2012) utilisent les histogrammes de mots visuels comme caractéristiques d'entrée du SVM. Deux modèles sont entraînés parallèlement sur les descripteurs de couleurs et de textures (LAB et SIFT) et leurs scores sont ensuite combinés (Yanulevskaya & al., 2012). Cette approche est très précise et très adaptée à des jeux de données complexes.

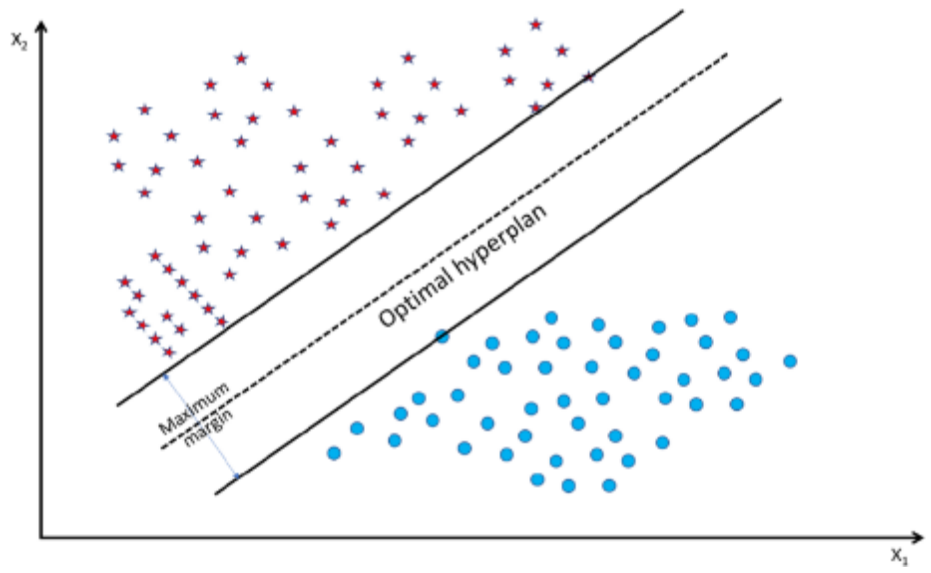


Figure 7 Support Vector Machine (SVM)

Source : https://www.researchgate.net/figure/Support-Vector-Machine-SVM-based-classification_fig3_362604337

2.2.2. Analyse de la texture

La texture d'une image numérique est un concept difficile à définir de manière précise ou complète, bien qu'il soit aisé pour un humain d'utiliser certains adjectifs (rugueux, régulier, doux, etc.) pour la caractériser (Mirmehdi, Xie, & Suri, 2008). Le système visuel humain est capable d'interpréter rapidement les variations d'intensité ou de couleur d'une surface, ce qui peut porter à croire que l'analyse de texture est triviale, mais les traitements réalisés par le cerveau humain sont en réalité très complexes (Mirmehdi & al., 2008). Cette capacité d'interprétation nous permet de facilement discriminer et reconnaître divers objets en fonction de leur texture, et cette aptitude intéresse depuis longtemps le monde de la vision par ordinateur.

Selon Mirmehdi & al. (2008), une texture peut avoir un motif périodique ou aléatoire (souvent un mélange des deux) et elle peut être décrite par sa microstructure (taux de détail) et sa macrostructure (uniformité générale) (Mirmehdi & al., 2008). De nombreux domaines de recherche s'intéressent à l'analyse de texture et pour chacun d'entre eux, certaines caractéristiques de texture ont une importance toute particulière et diverses méthodes d'analyse sont utilisées. Toutefois, pour l'analyse d'images numériques, les approches connues peuvent généralement être catégorisées en 4 grandes familles (Bharati, Liu, & MacGregor, 2004) :

- Les approches statistiques
- Les approches basées sur les transformées

- L'analyse multivariée des images

- L'analyse de texture par ondelettes

Chacune de ces approches englobe diverses méthodes qui permettent d'extraire des caractéristiques différentes. Leur utilisation est donc fortement dépendante du domaine d'application et des objectifs de l'analyse de texture. Parmi les méthodes les plus répandues, on retrouve :

- La matrice de cooccurrence de niveaux de gris (GLCM) (approche statistique) qui mesure l'intensité des pixels (niveau de gris) en observant des paires de pixels. Les résultats sont enregistrés dans une matrice, et à partir de celle-ci, différentes caractéristiques comme le contraste ou l'entropie peuvent être mesurées (Bharati & al., 2004).

- La transformée en ondelettes (WTA) (analyse de texture par ondelettes) qui décompose une image en plusieurs sous-images des hautes fréquences (qui capturent les détails fins) et des basses-fréquences (qui capturent les formes globales). La quantité de détail de chaque sous-image est calculée et permet d'extraire, par exemple, l'entropie ou la variance (taux de variation des niveaux de gris) (Bharati & al., 2004).

- La transformée de Fourier rapide (FFT) (approche basée sur les transformées) qui transforme une image en une représentation fréquentielle. Cette transformation permet d'obtenir un spectre fréquentiel dans lequel les fréquences basses indiquent des variations lentes ou des grandes formes, et les fréquences hautes mettent en valeur les détails fins. Ainsi, le contraste, la directionnalité ou encore l'homogénéité de la texture peuvent être évalués (Bharati & al., 2004).

Dans leur publication intitulée « Textural Features Corresponding to Visual Perception », Tamura, Mori et Yamawaki (1978) se sont intéressés aux liens entre la perception visuelle humaine et diverses caractéristiques texturales. Ils se sont basés sur six caractéristiques texturales, la granularité (coarseness), le contraste, la directionnalité, la linéarité (line-likeness), la régularité et la grossièreté (roughness), et les mesures computationnelles ont été comparées avec les évaluations psychologiques de 48 personnes (Tamura & al., 1978). Les résultats font ressortir une corrélation importante entre les mesures computationnelles et les évaluations humaines, principalement pour le contraste, la directionnalité et la grossièreté (Tamura & al., 1978).

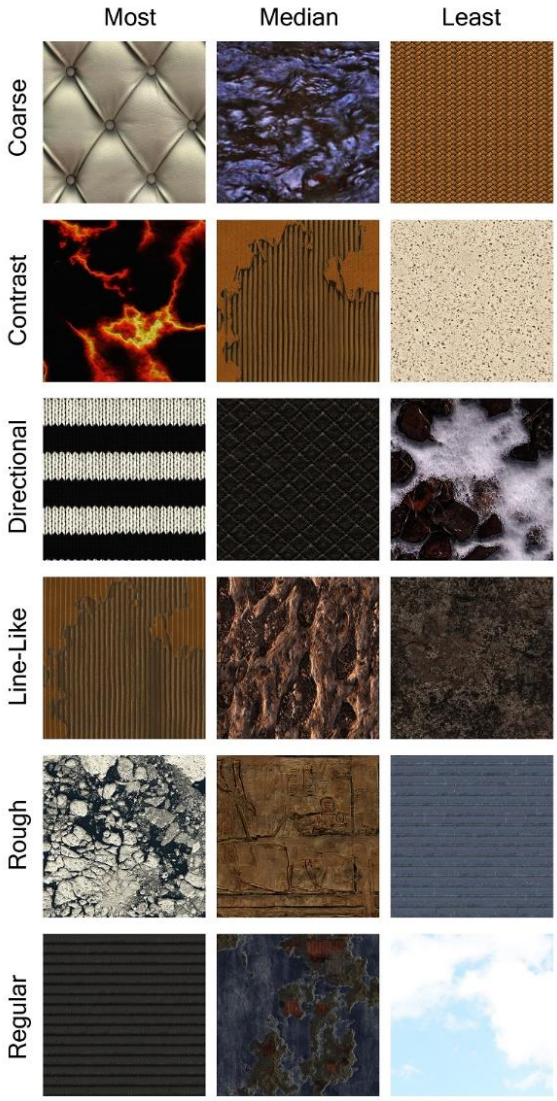


Figure 8 Exemples de caractéristiques de textures Tamura

Source : Brown, & al. (2021)

Ces caractéristiques semblent donc être intéressantes à étudier dans le cadre de ce travail. Nous reviendrons sur certaines de ces méthodes au chapitre 3.2.2.

2.2.3. Analyse des couleurs

L'analyse des couleurs a été brièvement discutée dans le chapitre sur l'analyse des émotions. Nous allons dans ce chapitre revenir sur ce sujet plus en détail et analyser les différentes étapes qui peuvent être nécessaires à l'évaluation correcte des propriétés de couleur dans le cadre de ce travail.

Prétraitement

Bien que le besoin de cette étape soit dépendant du but de l'analyse, le prétraitement d'une image peut s'avérer essentiel à l'extraction précise des caractéristiques souhaitées. Pour l'analyse de couleurs, les étapes suivantes sont généralement considérées.

- **Segmentation d'image** : la segmentation permet d'isoler les régions d'intérêt ou d'extraire des régions homogènes en couleur et en texture. En fonction des objectifs de l'analyse, la division en régions cohérentes, préservant les détails fondamentaux des couleurs, peut s'avérer importante (Marchenko, Chua, & Aristarkhova, 2005 ; Carson, Belongie, Greenspan, & Malik, 2002).

- **Conversion d'espace colorimétrique** : certains espaces colorimétriques (CIE Luv ou CIE Lab) sont conçus pour être uniformes à la perception humaine des couleurs, contrairement, par exemple, à l'espace RGB. Ainsi, une conversion vers l'un de ces espaces permet une analyse plus précise de certaines caractéristiques de couleur comme la saturation, l'intensité ou la teinte (Marchenko & al., 2005).

Extraction des caractéristiques

L'extraction de caractéristiques implique l'identification et la quantification des propriétés des couleurs. Cela peut inclure le calcul d'histogrammes de couleurs, l'identification des couleurs dominantes, ou l'analyse de divers aspects des couleurs comme la température, la palette ou le contraste. Les caractéristiques extraites peuvent varier grandement en fonction du but de l'analyse, mais voici quelques exemples :

- **Histogrammes de couleurs et couleurs dominantes** : le calcul d'histogrammes de couleurs permet de quantifier la distribution des couleurs dans une image. Les couleurs dominantes sont identifiées en analysant les pics dans ces histogrammes. Cette méthode est couramment utilisée pour capturer la distribution globale des couleurs et faciliter la recherche d'images par similarité de couleur (Marchenko et al., 2005).

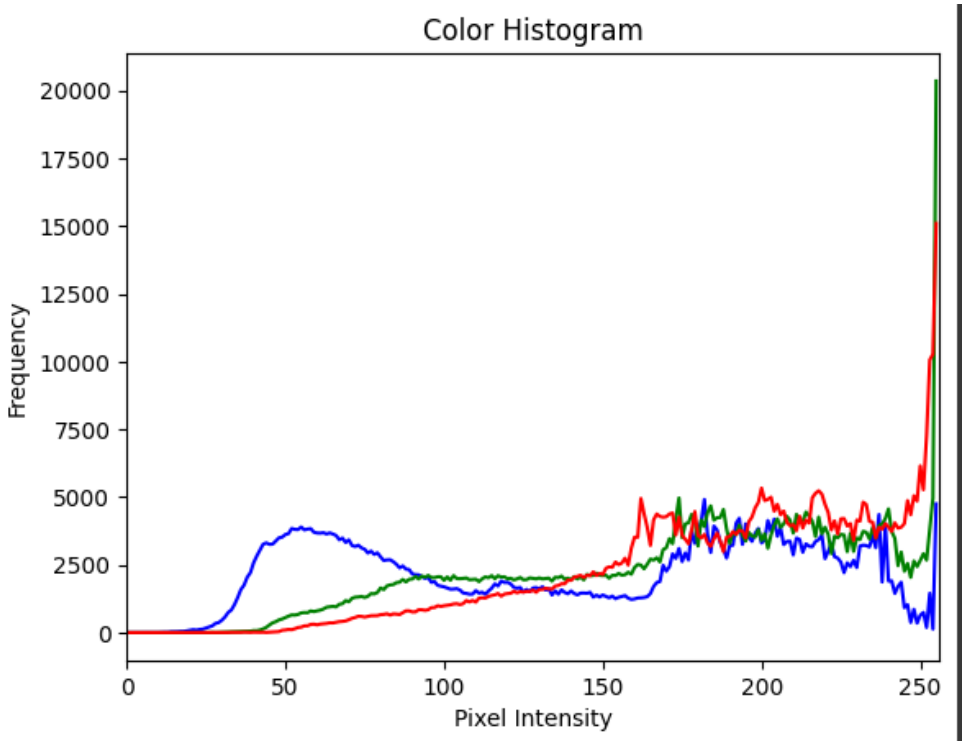


Figure 9 Histogramme de couleurs

Source : https://zilliz.com/learn/demystifying-color-histograms

- **Analyse de la température des couleurs** : les couleurs peuvent être classées en "chaudes", "froides" ou "neutres" en fonction de leurs teintes et intensités. Cette classification est souvent réalisée en calculant la moyenne des valeurs des pixels dans une région et en utilisant des algorithmes de machine learning, comme les SVM pour attribuer des concepts de température (Marchenko et al., 2005). Cette approche permet d'analyser comment différentes températures de couleur influencent la perception globale de l'image.

- **Analyse de la palette de couleurs** : l'examen de la distribution spatiale des couleurs permet de classifier les régions selon des palettes de couleurs primaires, complémentaires et tertiaires. Le calcul du vecteur de cohérence des couleurs aide à déterminer la prévalence des différentes catégories de palettes dans une image, ce qui est utile pour analyser les schémas de couleurs globaux et leur impact perceptuel (Marchenko et al., 2005).

Entraînement d'un modèle

L'entraînement de modèles d'apprentissage automatique pour la reconnaissance et la catégorisation de couleurs repose sur l'utilisation de jeux de données étiquetées. Les applications qui nous intéressent dans le cadre de ce projet sont la détection de la couleur dominante et le nommage de celle-ci. Nous nous intéresserons donc à des travaux ayant pour objectif l'un ou l'autre de ces aspects.

Couleur dominante

Pour la détection de couleurs dominantes, l'entraînement des modèles sur des ensembles de données où les couleurs dominantes ont été identifiées par des sujets humains, permet de prendre en compte la couverture des couleurs, la luminosité et la diversité des couleurs. Cette approche aide à identifier avec précision les couleurs dominantes dans les images, facilitant des tâches comme la classification et la recherche basée sur le contenu. Pour cette tâche, Weingerl, Hladnik, & Javoršek (2020) utilisent la régression Lasso pour sélectionner les caractéristiques les plus pertinentes. La régression Lasso est une technique de régression linéaire qui permet d'éliminer les variables non pertinentes (Yapi, n.d.).

Nommage des couleurs

Pour le nommage des couleurs, van de Weijer, Schmid, & Verbeek (2007) utilisent un modèle probabiliste, le Probabilistic Latent Semantic Analysis (PLSA). Le PLSA est généralement utilisé pour analyser les relations entre une collection de documents et les termes qu'ils renferment (Analyse sémantique latente probabiliste, 2018). Dans le cadre du nommage des couleurs, les images sont traitées comme des documents et les couleurs comme des mots. Cette méthode utilise un modèle génératif pour trouver des mélanges de noms de couleurs dans les images (van de Weijer, Schmid, & Verbeek, 2007). Schauerte & Stiefelhagen (2012) apportent une suggestion d'amélioration au modèle en utilisant des méthodes de détection d'objets saillants pour diminuer l'influence des arrières-plans.

Dans leur publication " Color Naming for Describing Object in Image using Different Classification Algorithms and Color Spaces ", Sainui & Tongsamrit (2020) explorent plusieurs approches pour le nommage de couleurs en jouant avec divers espaces colorimétriques et algorithmes de classification. Ils utilisent pour un jeu d'images annotées où la zone d'intérêt est définie par un masque, et leurs résultats expérimentaux montrent que l'algorithme SVM linéaire combiné à l'espace LAB est la combinaison la plus prometteuse avec une précision moyenne de 73.18% (Sainui & Tongsamrit, 2020).

Figure 10 Exemple d'une image originale et de l'image masquée

Source : Sainui & Tongsamrit (2020)

Enfin, plusieurs ressources en ligne utilisent simplement des jeux de données construits sur la base de valeurs RGB associées à un nom de couleur. Siraudin (2022) utilise un tel dataset pour entraîner un modèle en comparant 5 classificateurs différents : SVM, Nearest Neighbors, Gaussian Naive Bayes, Random Forest et XGBoost.

2.2.4. Analyse du style

La détection des styles artistiques implique la classification des œuvres d'art en différents styles à l'aide de méthodes computationnelles. Cette tâche est devenue de plus en plus importante dans le domaine de la vision par ordinateur et de l'apprentissage automatique en raison de ses applications potentielles dans l'indexation de grandes bases de données artistiques, l'amélioration des expériences artistiques numériques et l'assistance à la recherche en histoire de l'art notamment. Ce chapitre passe en revue les techniques de pointe dans la détection des styles artistiques, en mettant l'accent sur les contributions des modèles d'apprentissage profond, en particulier les réseaux de neurones convolutifs.

Les approches traditionnelles de détection des styles artistiques ont longtemps reposé sur des caractéristiques conçues à la main et des algorithmes d'apprentissage automatique classiques. Des méthodes telles que l'utilisation des histogrammes de gradients, des enveloppes spatiales et des noms de couleurs discriminants avec des classificateurs comme les SVM et les Random Forest étaient courantes (Lecoutre, Negrevergne, & Yger, 2017). Cependant, ces techniques ont rapidement rencontré des limitations en raison notamment de la complexité inhérente et de la variabilité des styles artistiques, qui manquent souvent de caractéristiques clairement identifiables (Lecoutre & al., 2017).

L'introduction de l'apprentissage profond, en particulier des CNN, a marqué une avancée significative dans la détection de styles. Les CNN apprennent automatiquement des caractéristiques hiérarchiques à partir des données de pixels bruts, ce qui les rend très efficaces pour les tâches de classification d'images. Les premières tentatives de Karayev & al. (2014) ont utilisé l'architecture « AlexNet », pré-entraînée sur le dataset ImageNet (qui est une base de données d'images annotées), pour la reconnaissance du style artistique. Cette approche a surpassé les méthodes précédentes reposant sur des caractéristiques conçues à la main, atteignant des améliorations notables en termes de précision (Lecoutre & al., 2017).

S'appuyant sur le succès des CNN, des architectures plus profondes et plus sophistiquées ont été explorées. Lecoutre & al. (2017) ont utilisé des réseaux neuronaux résiduels profonds (ResNet), qui introduisent des connexions de raccourci pour atténuer le problème de la dégradation des gradients dans les réseaux très profonds (Lecoutre & al., 2017). Leur travail a démontré que ResNet, lorsqu'il est pré-entraîné sur ImageNet et ensuite réentraîné sur le dataset Wikipaintings, atteignait une précision de pointe de 62% sur 25 styles artistiques différents. Ils ont découvert que le réentraînement

d'environ 20 couches du réseau produisait les meilleures performances, soulignant l'importance de l'ajustement fin des modèles pré-entraînés pour des tâches spécifiques (Lecoutre & al., 2017).

Pour répondre au défi des données d'entraînement limitées, les chercheurs ont utilisé des techniques d'augmentation des données. En appliquant des transformations aléatoires telles que des inversions, des rotations et des zooms aux images d'entraînement, les modèles deviennent plus robustes et généralisables. Lecoutre & al. (2017) ont également utilisé le bagging, où plusieurs prédictions à partir de différentes variations des données d'entrée sont moyennées, ce qui conduit à une amélioration de la précision.

Les efforts récents ont continué à repousser les limites de la détection des styles artistiques. Munir (2019) a exploré l'application de diverses architectures CNN, y compris le modèle pré-entraîné VGG19, pour classer des peintures selon leur style. En utilisant l'apprentissage par transfert et l'ajustement fin de ces modèles sur le dataset WikiArt (qui est une collection d'images récupérées sur WikiArt.org), Munir a atteint une précision de classification élevée, démontrant l'efficacité des architectures CNN modernes.

De même, Lafay, Sustrac, Mas, & Lavallee (2021) ont mis en œuvre une approche basée sur le réseau neuronal convolutif VGG16 pour la reconnaissance des styles artistiques, en se concentrant sur les techniques de prétraitement et l'importance d'un pipeline d'entraînement bien structuré. Leur travail a mis en évidence les aspects pratiques de la mise en œuvre de modèles d'apprentissage profond, tels que la gestion des données, l'entraînement des modèles et l'évaluation des résultats. Le modèle de Lafay & al. (2021) indique un pourcentage de confiance pour chacun des 8 styles évalués.

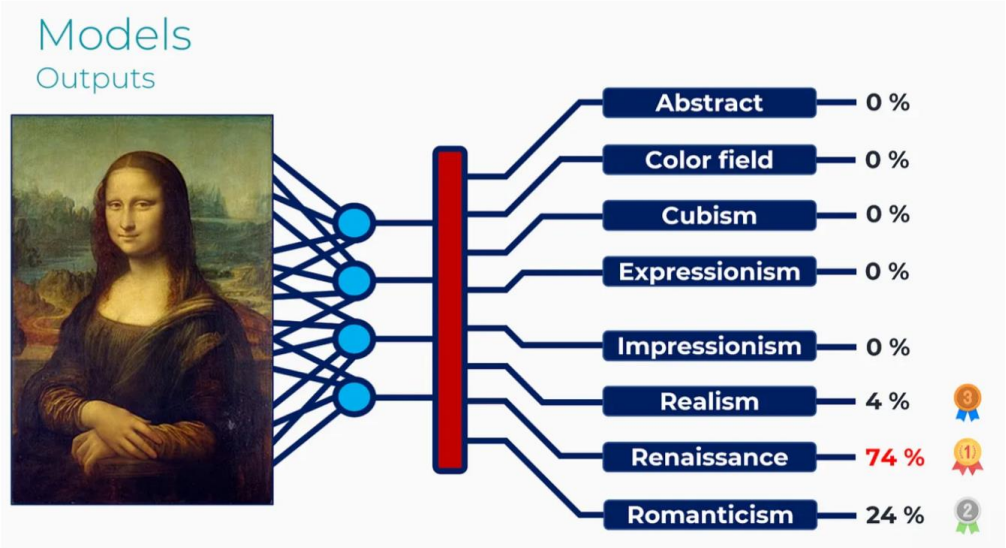


Figure 11 Résultats - Modèle de Lafay & al. (2021)

Source : Lafay & al. (2021)

2.3. Synthèse sonore

Le Larousse définit la synthèse sonore comme une « technique qui permet d'obtenir des sons à partir d'un matériel électronique et/ou informatique » (Synthèse sonore, (s.d.)). Wikipédia formule sa définition par « un ensemble de techniques permettant la génération de signaux sonores » (Synthèse sonore, 2024). Nous allons dans ce chapitre nous concentrer sur son application dans le domaine de la musique bien que la synthèse sonore ne se limite pas à ces frontières (son application peut s'étendre à télécommunications, à la réalité virtuelle, etc.).

Avant de s'intéresser aux techniques de synthèse sonore, nous allons nous pencher brièvement sur la définition et les caractéristiques d'une onde sonore. Il s'agit d'une onde longitudinale qui peut être décrite par sa longueur d'onde (la distance entre deux points de phase égale), sa fréquence (le nombre de cycles par seconde), son amplitude (le déplacement maximum de particules) et sa vélocité (la vitesse de propagation dans un médium donné) (Groves, et al., 2018). À l'origine de la plupart des types d'onde sonore, on trouve l'onde sinusoïdale illustrée ci-dessous.

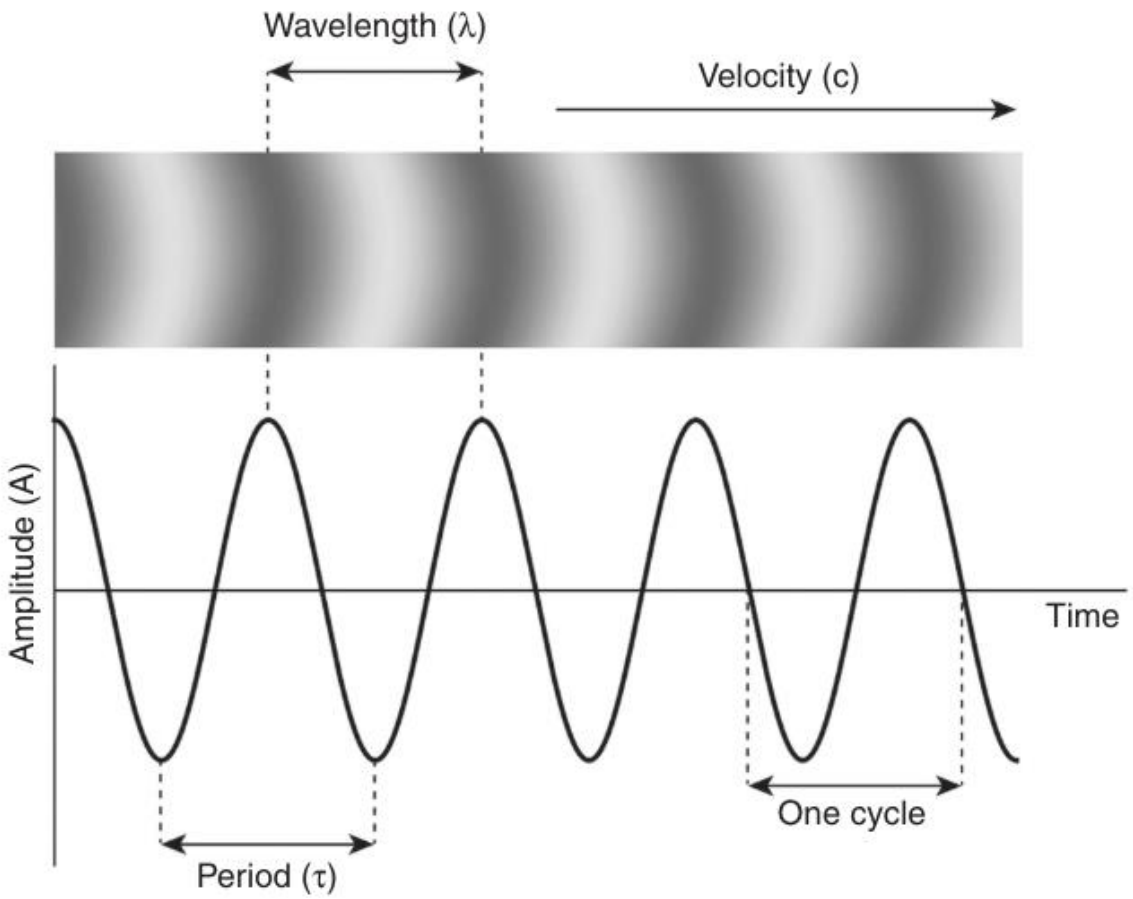


Figure 12 Onde sinusoïdale et ses paramètres

Source : Groves, et al., 2018

Cette onde est fondamentale dans le sens qu'elle n'est composée d'aucune harmonique. L'image ci-dessous illustre la simplicité d'une onde sinusoïdale à travers son spectre fréquentiel. On y voit que pour une fréquence générée, ici de 440 Hz, aucune information fréquentielle hors de cette dite fréquence ne peut être observée. Une onde sinusoïdale produit donc un son simple et pauvre harmoniquement.

Figure 13 Onde sinusoïdale à 440 Hertz

Source : Données de l'auteur

D'une onde sinusoïdale peuvent naître de multiples ondes plus complexes. Parmi elles les ondes carrées, impulsionnelles, triangulaires ou en dents de scie, qui ont toutes un spectre fréquentiel plus complet avec de multiples harmoniques et inharmoniques (Newjazz, 2014). Certaines de ces ondes seront discutées plus en détails lors de l'implémentation de l'outil de synthèse sonore.

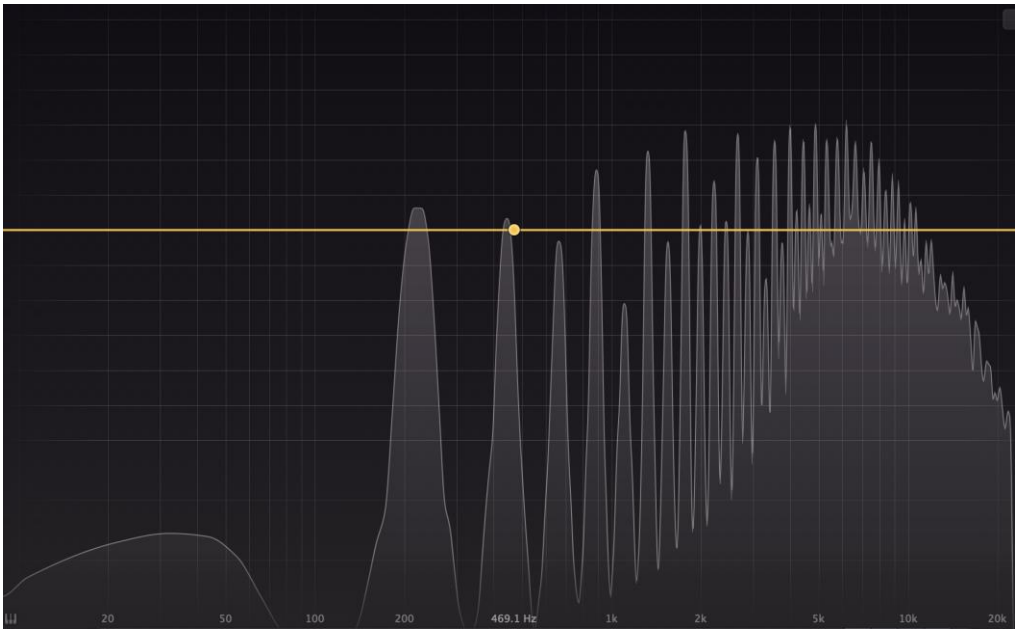


Figure 14 Onde en dents de scie à 440 Hertz

Source : Données de l'auteur

Les ondes dont nous venons de discuter peuvent être générées, combinées ou modifiées grâce à différentes techniques de synthèse sonore. Celles-ci peuvent être divisées en 4 grandes familles :

- La synthèse par algorithmes abstraits : dans cette catégorie, nous trouvons notamment la synthèse par modulation de fréquences (FM) et la synthèse soustractive. Ces méthodes utilisent divers algorithmes pour la génération de sons (Synthèse sonore, 2024).

- La synthèse basée sur des échantillons : les techniques de cette catégorie utilisent des enregistrements existants sur lesquels différents traitements sont appliqués pour générer de nouveaux sons. Le sampling, la synthèse granulaire et la synthèse par table d'ondes font notamment partie de cette catégorie (Synthèse sonore, 2024).

- La synthèse par modèles de signaux : les techniques de ce type de synthèse superposent des signaux simples (harmoniques) en se basant généralement sur les travaux du mathématicien Joseph Fourier (Synthèse sonore additive, 2023). La synthèse additive est la technique la plus utilisée.

- La synthèse par modélisation physique : les modèles physiques simulent le comportement d'instruments réels en utilisant des équations numériques pour reproduire la manière dont les ondes sonores sont générées naturellement (Serra X. , 2007)

L'exemple le plus parlant d'un outil qui utilise la synthèse sonore est bien entendu le synthétiseur. Son histoire commence au début du 20e siècle, avec les premiers instruments électroniques comme

la Thérémine et l'Ondes Martenot, mais c'est véritablement dans les années 1960 et 1970 que les synthétiseurs commencent à prendre leur essor, notamment grâce aux avancées technologiques et à l'intérêt croissant pour la musique électronique (Martinez-Zorrilla, 2008). Dans les années 1980, un synthétiseur ne fait en générale qu'un type de synthèse (Synthèse sonore, 2024), mais le développement de la synthèse sonore a progressé significativement avec l'avènement des ordinateurs, permettant des innovations majeures dans la manière de produire et de manipuler le son (De Poli, 2023), ainsi qu'une complexification des possibilités de synthèse.

3. Méthodes

3.1. Contraintes

Pour la réalisation de ce travail, le service informatique de la HES-SO Valais/Wallis a mis à disposition une machine virtuelle Linux (Ubuntu 22.04.3 LTS, Kernel version 5.15.0-116-generic, 4 CPUs Intel(R) Xeon(R) Platinum 8358 CPU @ 2.60 GHz, 100 GB de mémoire). Cette machine sera utilisée comme serveur sur lequel tous les outils nécessaires au processus de sonification seront installés.

Le système d'exploitation (OS) de cette machine impose certaines contraintes dans le choix des outils utilisés, notamment pour les logiciels de synthèse sonore qui ne sont pas nécessairement tous compatibles avec un OS Linux.

Pour chacun des choix faits ci-après, le coût sera également pris en considération, favorisant les outils gratuits et Open Source.

3.2. Choix des outils pour l'analyse d'images

3.2.1. Analyse des émotions

Pour le choix de la méthode d'extraction des émotions, les 4 approches utilisées dans les articles suivants ont été comparées :

- Recognizing Emotions from Abstract Paintings using Non-Linear Matrix Completion (Alameda-Pineda & al., 2016) (abbrégé en "NLMC" dans le tableau comparatif)
- Who's afraid of Itten: Using the art theory of color combination to analyze emotions in abstract paintings (Sartori & al., 2015a) (abbrégé en "SGL" dans le tableau comparatif)
- In the eye of the beholder: Employing statistical analysis and eye tracking for analyzing abstract paintings (Yanulevskaya & al., 2012) (abbrégé en "SVM" dans le tableau comparatif)
- Affective Analysis of Professional and Amateur Abstract Paintings Using Statistical Analysis and Art Theory (Sartori & al., 2015b) (abbrégé en "SVM + texte" dans le tableau comparatif)

Ces travaux de recherche ont comme point commun qu'ils utilisent tous la couleur et la texture comme éléments d'extraction de l'émotion dominante. Cet aspect est important dans le cadre de ce travail qui vise à rendre la sonification possible autant pour des œuvres abstraites que figuratives.

Ces démarches ont été analysées et notées selon les aspects suivants :

- L'approche est adaptée aux besoins du projet
- La facilité de réalisation
- La disponibilité de bibliothèques existantes
- Le niveau de détail de l'article
- La précision du modèle
- La flexibilité pour l'analyse de différents types d'œuvres d'art
- Le coût computationnel
- La disponibilité de jeux de données annotés

Critères / Approches	NLMC	SGL	SVM	SVM + texte
Adaptation aux besoin	4	4	3	4
Facilité de réalisation	3	4	4	3
Bibliothèques existantes	3	4	5	4
Article détaillé	4	5	4	4
Performance et précision	4	4	4	4
Flexibilité	4	4	3	4
Coût computationnel	3	4	4	3
Disponibilité de jeux de données annotés	3	3	4	4

Score	28	**32**	31	30

Tableau 1 Tableau comparatif des méthodes d'extraction d'émotions

Cette analyse montre que :

- NLMC est avantageux en termes de complexité (relations multi-label) mais complexe à mettre en œuvre et coûteux en termes de calcul.
- SGL est bon en termes de facilité de réalisation et de bibliothèques existantes. L'article est également très détaillé.
- SVM est assez simple à la réalisation et utilise des bibliothèques existantes, mais il est moins précis et donc moins adapté aux besoins du projet.
- SVM + texte est plus coûteux en ressources du fait de son utilisation de données textuelles, et la nature des données extraites peut rendre l'intégration au processus de sonification plus complexe.

Le SGL, avec un score final de 32 sur 40, semble donc être la méthode la plus adaptée aux besoins de ce projet et sera utilisé comme base de travail pour l'implémentation du modèle d'analyse des émotions.

L'annexe 10.1 « Choix de la méthode d'analyse des émotions » détaille les notes attribuées dans le tableau ci-dessus.

3.2.2. Analyse de la texture

Parmi les approches discutées dans la revue de la littérature, quelques-unes ont été testées et évaluées : Tamura, Fast Fourrier Transform (FFT) et Gray Level Co-occurrence Matrix (GLCM). Celles-ci ont été sélectionnées pour la disponibilité d'implémentations et d'articles de recherche à leur sujet.

Tamura

Les fonctionnalités de Tamura ont été introduites pour mieux correspondre à la perception humaine de la texture. Les six caractéristiques principales définies par Tamura sont la granularité ou grossièreté (« coarseness »), le contraste, la directionnalité, la linéarité (line-likeness), la régularité et la rugosité (Tamura & al., 1978).

Avantages : cette méthode est alignée avec la perception humaine et un lien passablement intuitif peut être tiré entre les caractéristiques visuelles et sonores (dynamique, granularité, direction, etc.).

Limitations : dans le contexte de ce travail, la normalisation des valeurs est importante et les plages de valeurs étant très variables, une normalisation très précise est nécessaire.

Fast Fourier Transform

Le Fast Fourier Transform convertit une image du domaine spatial au domaine fréquentiel. L'analyse des fréquences constituant l'image est utilisée pour identifier des structures périodiques et les orientations des textures (Bharati & al., 2004). Le FFT permet d'extraire les fréquences dominantes d'une image, de révéler des motifs répétitifs ou des régularités spatiales et la magnitude des différentes fréquences qui peut être utilisée pour classifier les textures (Transformation de Fourier rapide, 2023).

Avantages : cette méthode est très adaptée pour la détection de motifs périodiques ou répétitifs et le passage au domaine fréquentiel permet de réduire la complexité en se focalisant sur les composantes majeures plutôt que sur des détails fins.

Limitations : l'analyse peut être coûteuse en temps de calcul pour des images en haute résolution.

Gray Level Co-occurrence Matrix

La matrice de cooccurrence de niveaux de gris exprime à quelle fréquence des paires de pixels, avec des valeurs de niveau de gris spécifiques, apparaissent dans une image à une certaine distance et orientation, et à partir de laquelle plusieurs caractéristiques texturales peuvent être extraites (Bharati & al., 2004). Cette méthode permet d'extraire l'uniformité de la texture en calculant à quel point les valeurs des pixels sont répétitives, le contraste, la corrélation des niveaux de gris de pixels voisins ou encore l'homogénéité des textures (Bharati & al., 2004).

Avantages : GLCM permet une analyse détaillée des textures (contraste, corrélation, énergie), peut être adaptée pour analyser des textures à différentes échelles.

Limitations : les matrices complexes (pour des images de haute résolution) sont coûteuses en temps de calcul, et les résultats peuvent être très variables en fonction des paramètres de la méthode.

Choix de l'outil

En complément des avantages et inconvénients discutés ci-dessus, les éléments suivants ont également été pris en compte pour le choix de l'approche : la disponibilité d'implémentations existantes et l'utilité des caractéristiques extraites.

En considérant tous ces aspects, mon choix se porte sur la méthode Tamura. La raison principale étant la relation intuitive entre les caractéristiques visuelles extraites et certaines caractéristiques

sonores comme la dynamique, la granularité ou le positionnement d'un son. De plus, certaines limitations de cette méthode peuvent être éludées par le prétraitement des œuvres et l'analyse du jeu de données utilisé. Enfin, il existe de nombreuses implémentations existantes qui peuvent être comparées et modifiées.

3.2.3. Analyse des couleurs

Pour choisir les outils d'analyse les plus adaptés aux besoins du projet, il est nécessaire de définir l'utilisation qui sera faite des caractéristiques extraites. L'approche vis-à-vis des couleurs est ici différente que lors de l'analyse des émotions. La couleur dominante sera utilisée pour définir la tonalité de la composition. Les raisons et les correspondances exactes entre les couleurs et les notes seront discutées dans un chapitre suivant. Ce qui est important de prendre en compte ici est la taille de la palette qui sera utilisée pour cette correspondance, soit 12 couleurs qui correspondent aux 12 notes qui forment la base du vocabulaire de la musique occidentale (gamme chromatique).

Palette de couleurs

Pour définir cette palette, divers travaux portant sur la théorie des couleurs ont été étudiés. Celui de Marchenko & al. (2005) porte sur la théorie des couleurs d'Itten. Cette théorie est largement acceptée pour son approche méthodique de la classification des couleurs et de leur interaction, et est référencée dans plusieurs travaux de recherche traitant de l'analyse des couleurs et des émotions. La roue chromatique d'Itten comporte 12 couleurs tertiaires et semble donc, de prime abord, être adaptée aux prérequis discutés plus haut. Cette proposition manque toutefois de diversité. Les couleurs d'une peinture peuvent être beaucoup plus complexes que ce que les douze couleurs tertiaires d'Itten permettent de représenter.

Figure 15 Cercle chromatique d'Itten

Source : https://www.chambaud-abstrait.com/non-classifiee/johannes-itten-et-le-cercle-chromatique-retour-sur-son-oeuvre-et-sa-theorie-de-la-couleur/

Dans un second temps, les recherches ont été axées sur la synesthésie, condition qui fait que certaines personnes perçoivent des couleurs en réponse à des sons (Synesthésie, 2024). Le compositeur et pianiste russe Alexander Scriabin a créé et utilisé un « clavier à lumières » pour son œuvre « Prometheus : Poem of Fire ». Il a pour l'occasion créé une association entre des couleurs et les notes de la gamme chromatique. Alexander Scriabin prétendait être synesthète et a construit son échelle selon ses propres perceptions des couleurs et des sons (Clavier à lumières, 2024). Cette approche n'est donc pas scientifique, d'autant plus que d'atures synesthètes assurent avoir une perception différente (Synesthésie, 2024). Il semble donc difficile de définir un système universel qui lie une couleur à une note. Cette conclusion est renforcée par le fait qu'une même note est perçue différemment en fonction de l'octave à laquelle elle est jouée (O'Toole, Glowinski, Pitt, & Mancini, 2021).

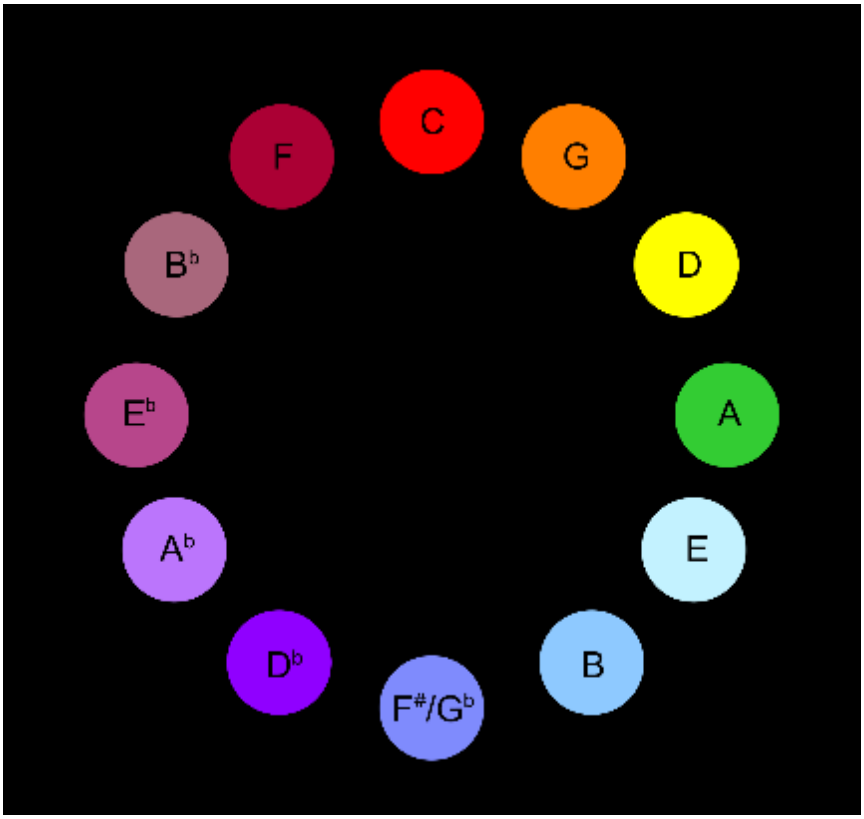


Figure 16 Scriabin color mapping

Source : https://fr.wikipedia.org/wiki/Clavier_%C3%A0_lumi%C3%A8res#/media/Fichier:Scriabin-Circle.svg

Pour le choix des couleurs, j'ai donc décidé de me baser sur la publication « Basic Color Terms: Their Universality and Evolution » de Berlin & Kay (1971) qui définit 11 couleurs fondamentales, reconnues dans plusieurs langues, et qui permettent de décrire n'importe quelle couleur. Ces couleurs sont : noir, bleu, brun, vert, gris, orange, rose, violet, rouge, blanc et jaune (Berlin & Kay, 1971). Pour arriver aux 12 couleurs nécessaires, j'ai considéré le fait que certaines langues, comme le russe, différencient deux teintes de bleu dans les couleurs de base : bleu clair et bleu foncé (Berlin & Kay, 1971). Le tableau des relations entre les notes et ces couleurs est discuté au chapitre portant sur le lexique de sonification.

Couleur dominante

Pour sélectionner l'approche la plus adaptée pour l'extraction de la couleur dominante, un sondage a été réalisé sur la base d'échantillons de couleur extraits à l'aide de diverses techniques.

Tout d'abord, deux prototypes de détection d'objets saillants ont été mis en place. La détection d'objets saillants consiste à estimer les éléments importants d'une image sans connaissance préalable de celle-ci (Cheng, Zhang, Mitra, & Huang, 2011). Les approches testées sont les suivantes :

1. **Médiane de luminance** : une carte de saillance est calculée après que l'image ait été convertie de l'espace BGR vers l'espace LAB, où le « L » représente la luminance, qui peut être décrite comme la luminosité d'une image en niveaux de gris (Deguerre, Chatelain, & Gasso, 2021). Cette carte met en exergue les pixels qui s'éloignent le plus de la médiane. L'image est ensuite binarisée et un flou gaussien est ajouté pour adoucir les bords.

    a. Avantages : cette approche est extrêmement rapide et nécessite peu de puissance de calcul.

    b. Inconvénients : les caractéristiques contextuelles et texturales de l'image ne sont pas prises en compte, ce qui peut entraîner des résultats imprécis.

2. **VGG16** : VGG16 est un réseau de neurones convolutifs pré-entraîné sur un large jeu de données (ImageNet) (Krajnc, 2024).

    a. Avantages : l'utilisation d'un réseau de neurones permet d'extraire des caractéristiques complexes et le pré-entraînement sur un large jeu de données améliore sa capacité à détecter les régions saillantes.

    b. Inconvénients : un prétraitement des images est nécessaire (modification des dimensions, normalisation) et VGG16 requiert des ressources de calcul importantes.

Une fois le masque de saillance appliqué, les techniques suivantes ont été implémentées pour l'extraction de la couleur dominante :

1. **KMeans** : pour l'extraction de la couleur dominante, KMeans utilise les pixels comme données et chaque cluster représente une potentielle couleur dominante, et celui avec le plus de pixels est choisi comme couleur dominante. Le nombre de clusters a été fixé à 5 pour ce prototype.

2. **Histogrammes de couleur** : un histogramme est créé pour chaque canal de l'espace colorimétrique (HSV dans notre cas), et le pic de l'histogramme, qui représente la couleur la plus présente, est choisi comme couleur dominante.

3. **Mean Shift** : Mean Shift est un autre mode de clustering qui ne nécessite pas un nombre de clusters prédéfini. Cette approche consiste en un déplacement itératif de chaque point de données vers la moyenne des points voisins dans un rayon donné, jusqu'à ce que les points convergent vers les régions à forte concentration de données (Mean shift, 2023).

Un échantillon de couleur a été extrait en utilisant les 3 méthodes d'extraction de la couleur dominante, avec et sans détection des objets saillants. Pour la détection d'objets saillants, la méthode de la médiane de luminance et le VGG16 ont été utilisés. Les participants ont donc comparé 9 échantillons de couleurs pour une image donnée.

Figure 17 Evaluation des méthodes d'extraction de la couleur dominante 1

Source : Données de l'auteur

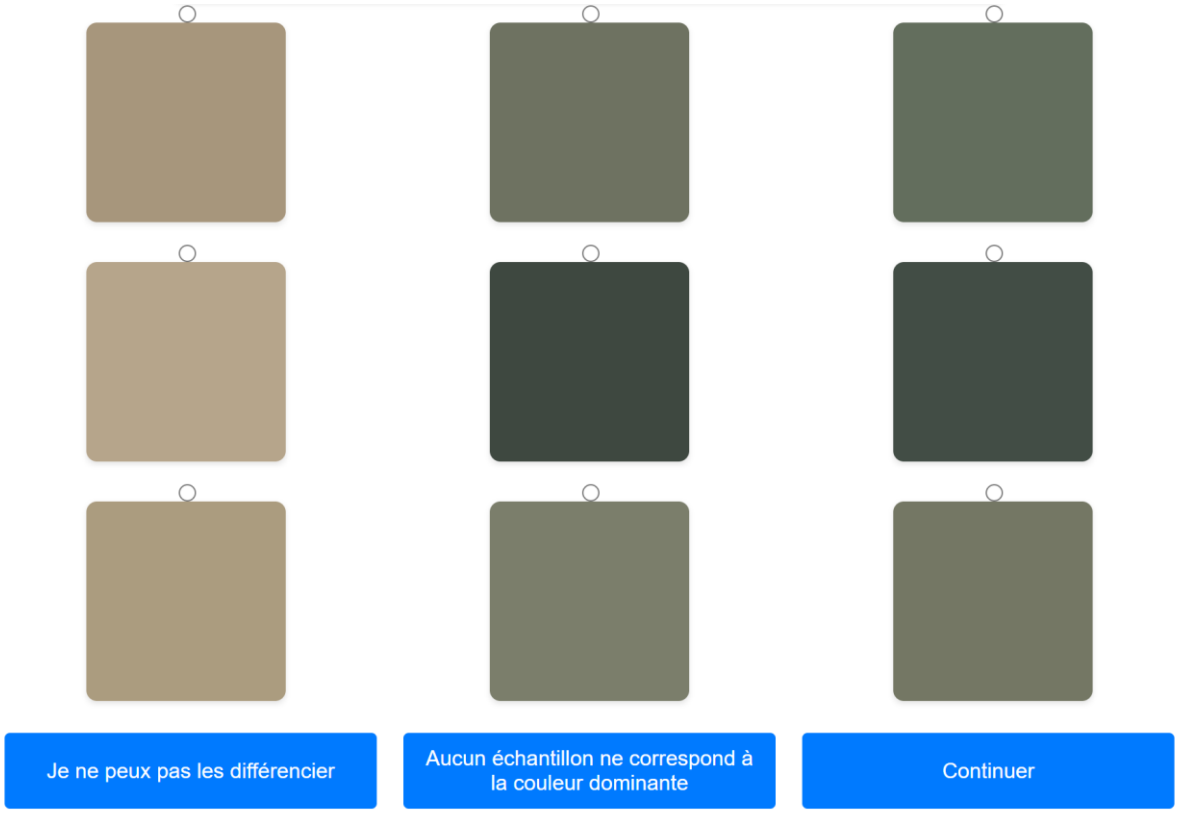


Figure 18 Evaluation des méthodes d'extraction de la couleur dominante 2

Source : Données de l'auteur

Comme le montrent les figures ci-dessus, pour chaque œuvre d'art (sélectionnée aléatoirement dans la base de données), 9 échantillons de couleur étaient présentés. Les participants avaient également la possibilité d'indiquer si aucun échantillon ne correspondait à la couleur qui leur paraissait dominante, ou si la différence de couleur entre les échantillons n'était pas perceptible. Ces deux éléments permettent d'éviter qu'un choix arbitraire soit fait dans le cas où le participant ne sait pas quoi sélectionner.

1047 votes ont été comptabilisés pour 41 participants (26 hommes, 14 femmes, 1 autre) âgés de 7 à 71 ans.

Méthode	Nombre de votes	Pourcentage
Aucun échantillon correspondant à la couleur dominante	90	8,595988539
Impossible de différencier les couleurs	34	3,247373448
Histogrammes	135	12,89398281
K-Means	**180**	**17,19197708**
Mean Shift	154	14,7086915
Médiane + Histogrammes	79	7,545367717
Médiane + K-Means	96	9,169054441
Médiane + Mean Shift	64	6,112702961
VGG-16 + Histogrammes	73	6,972301815
VGG-16 + K-Means	86	8,213944604
VGG-16 + Mean Shift	56	5,348615091

Figure 19 Comparaison des résultats

Source : Données de l'auteur

Ces résultats mettent en lumière le fait que la détection d'objets saillants n'est pas forcément nécessaire dans le cadre de l'extraction de la couleur dominante d'une peinture (du moins avec les techniques testées). Dans 44,79 % des cas, une méthode sans détection des objets saillants a été choisie, et la méthode KMeans est celle qui a été le plus sélectionnée. Dans 8,6 % des cas, les participants ont trouvé que les échantillons proposés ne correspondaient pas à la couleur dominante de l'image. Après avoir étudié les images concernées, il semblerait qu'elles aient comme point commun une certaine complexité et la présence de nombreuses couleurs différentes. Ci-dessous quelques images pour lesquelles « Aucun échantillon ne correspond à la couleur dominante » a été choisi par plusieurs participants.

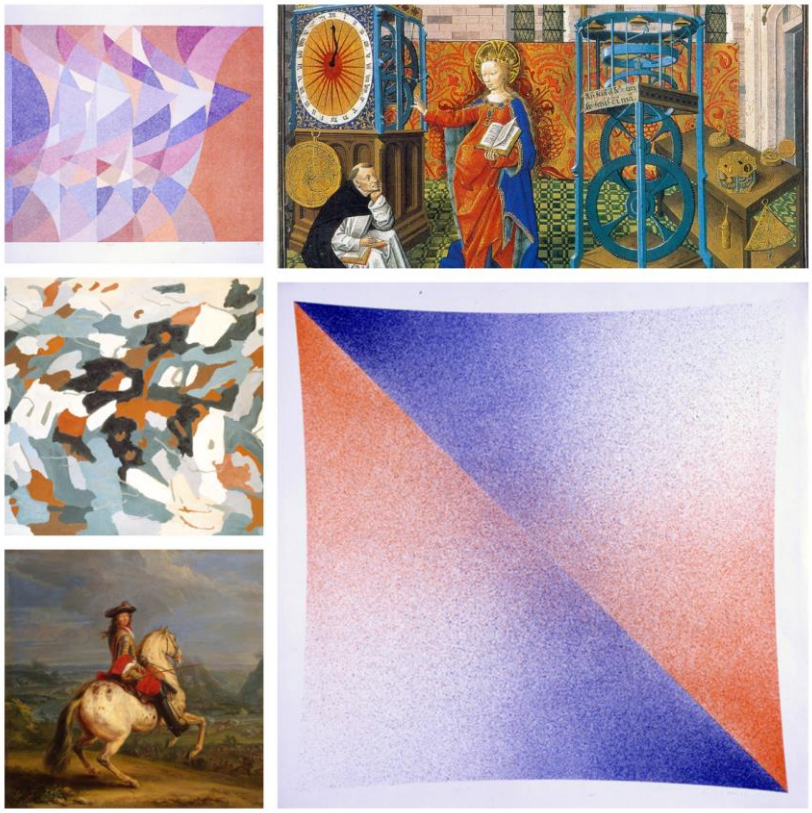


Figure 20 Exemples d'œuvres où la couleur dominante n'a pas été reconnue

Source : Données de l'auteur

Nous nous sommes rendu compte après l'expérience que la position des échantillons a pu influencer la sélection. Il est connu que les participants ont généralement une préférence pour les éléments rencontrés au début ou à la fin d'une séquence (effet de primauté et récence) (Canic & Pachur, 2014). Cet aspect a pu impacter certaines décisions, bien qu'en observant la fréquence de sélection de chaque échantillon au sein de certaines sessions, il semblerait qu'il n'y ait pas eu de réelle tendance à sélectionner, par exemple, le premier échantillon plus que les autres.

Les résultats discutés ci-dessus montrent que les méthodes peuvent être améliorées, notamment pour réduire les cas où la couleur extraite ne correspond pas à la couleur dominante. KMeans étant la méthode qui a reçu le plus de votes, elle sera utilisée pour extraire la couleur dominante des œuvres.

Nommage de la couleur

Van de Weijer & al. (2007) proposent dans leur publication "Learning Color Names for Real-World Images" d'apprendre les noms des couleurs directement à partir d'images du monde réel plutôt que de s'appuyer sur des échantillons de couleur isolés dans un environnement contrôlé. Cette méthode

vise à surmonter les limitations des approches traditionnelles qui utilisent des "color chips" pour étiqueter les couleurs, ce qui peut être inflexible et peu représentatif des conditions réelles d'usage des couleurs (van de Weijer, & al, 2007). Bien que cette approche semble donner de très bons résultats, sa mise en place est conséquente et compliquée. Il est nécessaire d'utiliser un grand jeu d'images masquées et annotées pour que le modèle puisse être entraîné efficacement.

Après une tentative d'implémentation donnant des résultats très imprécis liés au manque de données et à l'imprécision de ces dernières, j'ai décidé de tester une approche plus simple (discutée dans la revue de la littérature) qui consiste à utiliser des échantillons de couleur annotés. Pour la tester, le dataset créé par Chavan (2023) a été utilisé. Ce jeu de données utilise les 11 couleurs définies par Berlin & Kay (1971). Dans un premier temps, le dataset a été modifié pour séparer le bleu clair du bleu foncé pour l'adapter aux besoins de ce projet. Pour ce faire, la moyenne des valeurs de rouge et de vert a été évaluée et les échantillons avec une valeur plus grande que 128 ont été labellisés en bleu clair et ceux avec une valeur inférieure à 128 en bleu foncé. Les valeurs RGB ont également été converties en HSV dans un fichier séparé pour comparer les résultats générés par ces deux espaces colorimétriques.

Pour la classification des couleurs, trois méthodes ont été utilisées : Random Forest, SVM et XGBoost. Elles ont été choisies car elles sont toutes trois très utilisées et il existe de nombreuses bibliothèques existantes, facilitant ainsi la mise en place de prototypes.

Ces méthodes de classification utilisées avec les dataset initiaux (RGB et HSV) ont une bonne précision générale mais passablement de variations entre les différentes couleurs dues au nombre variable d'échantillons pour chaque couleur. Pour pallier ces imprécisions, j'ai utilisé ChatGPT pour générer deux datasets (RGB et HSV) comprenant chacun 500 échantillons par couleur. Ci-dessous, les valeurs de justesse des différents classificateurs.

- **HSV**
  - Random Forest : 0,951
  - SVM : 0,868
  - XGBoost : 0,953
- **RGB**
  - Random Forest : 0,989
  - SVM : 0,973
  - XGBoost : 0,988

Le modèle entraîné avec le jeu de données RGB et le Random Forest Classifier étant le plus précis, il sera utilisé pour le nommage de la couleur dominante.

3.2.4. Analyse du style

Tous les outils d'analyse de style étudiés utilisent des réseaux de neurones convolutifs pour classifier des œuvres selon leur style. Les travaux de Libert (2020), Munir (2019) et Lafay & al. (2021) ont notamment été évalués.

- Munir (2019) : un modèle VGG19 pré-entraîné est utilisé pour la classification des peintures selon leur style artistique. Une attention particulière est accordée à la sélection des styles qui ont suffisamment de données pour assurer un apprentissage robuste.

- Libert (2020) : des architectures de réseaux de neurones convolutifs comme ResNet50 et EfficientNet-B0 sont utilisées pour classer les peintures selon leur style. Ces modèles sont également pré-entraînés sur ImageNet avant d'être affinés pour cette tâche spécifique.

- Lafay & al. (2021) : 8 styles artistiques distincts sont classifiés en utilisant le modèle VGG16 pré-entraîné, appliquant également du transfert d'apprentissage. Le projet utilise des techniques telles que la rotation aléatoire et le zoom aléatoire pour l'augmentation des données.

Mon attention s'est initialement portée sur l'approche de Lafay & al. (2021). Un prototype a été implémenté en suivant les principales étapes proposées : réduction du dataset initial à 8 styles, choix et entraînement du modèle et visualisation des résultats. Pour cette phase de test, les résultats ont été analysés après seulement quelques itérations pour observer la tendance d'apprentissage du modèle et y apporter des modifications : réduction du taux d'apprentissage lorsque la validation stagne, earlystopping pour éviter le surajustement. À la suite de ces ajustements, le prototype a été entraîné sur 12 époques et le résultat, visible sur l'image ci-dessous, montre une tendance prometteuse.

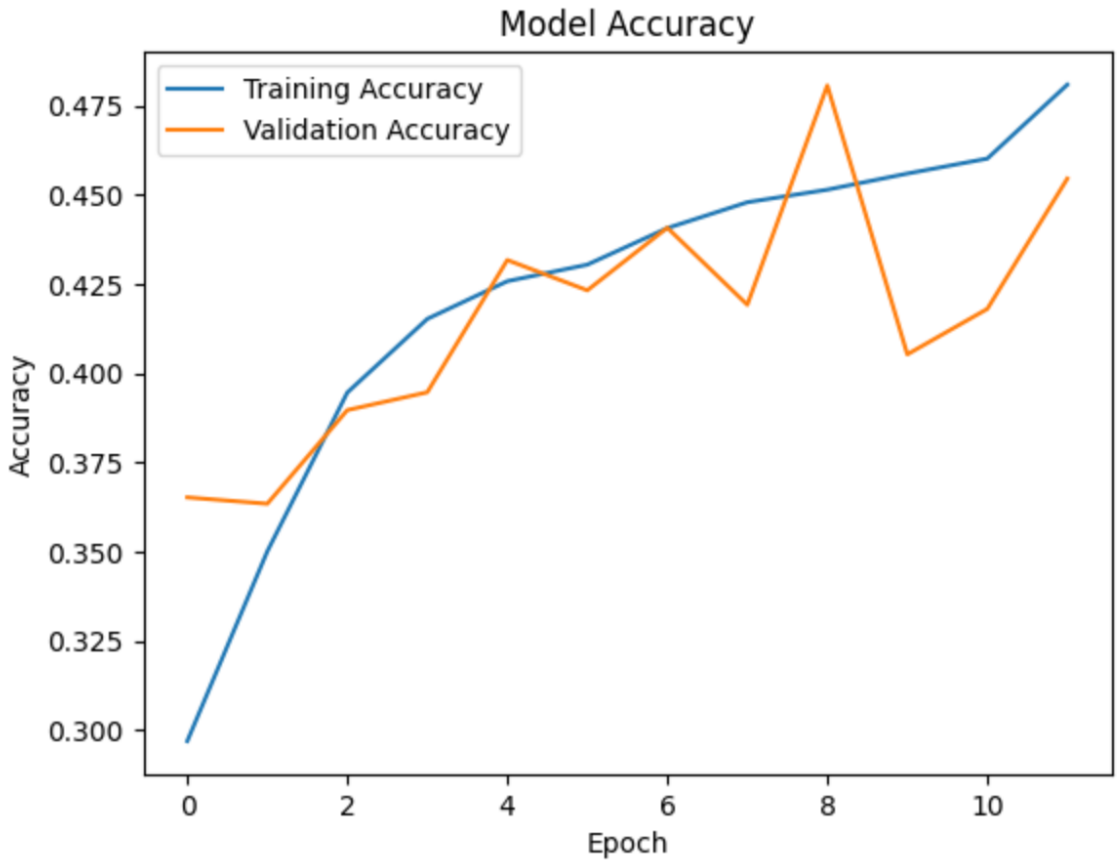


Figure 21 Justesse du prototype sur 12 époques pour les valeurs d'entraînement et de validation

Source : Données de l'auteur

L'implémentation et l'entraînement du prototype étant passablement complexes et chronophages, il a été décidé de ne pas tester les 3 approches. Ce choix est conforté par le fait qu'elles partagent un type d'architecture similaire (réseaux de neurones convolutifs) et ont été pré-entraînés sur de larges bases de données (comme ImageNet).

L'approche de Lafay & al. (2021) sera donc utilisée comme base de travail pour l'implémentation du modèle de classification de styles.

3.3. Choix de l'outil de synthèse sonore

Pour décider de l'outil le plus adapté pour la synthèse sonore, plusieurs critères ont été pris en compte : mes connaissances existantes de l'outil, la compatibilité avec le système d'exploitation du serveur, la facilité de prise en main, les possibilités de lecture et d'interprétation de fichiers (.json, .txt, ou autres), la disponibilité de documentation, l'adhésion avec les besoins du projet et le prix.

Les logiciels envisagés sont CSound, Max/MSP, Native Instruments Reaktor, Symbolic Sound Kyma, SuperCollider et Pure Data. Cette liste comprend les logiciels de programmation les plus connus et

les plus utilisés pour la synthèse sonore. Un premier tri s'est fait en éliminant tous les logiciels payants, soit CSound, Max/MSP, Native Instruments Reaktor & Symbolic Sound Kyma. Seuls SuperCollider et Pure Data sont gratuits et seront donc analysés plus en détail pour décider de l'outil utilisé pour ce projet.

3.3.1. Comparaison des outils

Critère / Logiciel	Pure Data	SuperCollider
Connaissance du logiciel	2	0
Compatibilité avec Linux	5	5
Facilité de prise en main	3	2
Lecture et interprétation de fichiers	3	4
Disponibilité de documentation	2	4
Adhésion aux besoins du projet	3	5
Total	19	**20**

Tableau 2 Comparatif des logiciels de synthèse sonore

L'interface graphique de Pure Data le rend plus accessible que son concurrent, mais ce dernier offre une plus grande flexibilité et une puissance supérieure pour la manipulation de données notamment. Malgré mes connaissances existantes du logiciel Pure Data, SuperCollider se montre être plus adapté aux besoins du projet et sera donc utilisé.

L'annexe 10.2 « Choix du logiciel de synthèse sonore » détaille les notes attribuées dans le tableau ci-dessus.

3.4. Choix de la méthode d'annotation des œuvres (modèle de classification des émotions)

N'ayant pas trouvé de jeu de données avec les annotations requises par ce projet, il a été décidé de récolter ces données primaires via un sondage. Pour décider de la méthode la plus appropriée, trois approches sont ici comparées. Celles-ci ont été choisies car elles sont toutes trois utilisées dans des travaux de recherche portant sur la détection d'émotion dans des images.

- Échelle de Likert : chaque participant évalue chaque image et note l'émotion ressentie sur une échelle de Likert où, par exemple, 1 représente une émotion très négative et 7 une émotion très positive.
    - Avantages : la méthode est facile à prendre en main pour les participants et permet d'obtenir une très bonne précision de la quantification des émotions perçues.
    - Inconvénients : certains biais peuvent influencer les résultats, comme la tendance à éviter les extrêmes.
- Comparaison de paires (TrueSkill) : chaque participant évalue des paires d'images et indique quelle œuvre suscite l'émotion la plus positive parmi les œuvres présentées.
    - Avantages : la méthode est facile à utiliser pour les participants et moins de comparaisons sont nécessaires que dans les autres approches.
    - Inconvénients : l'analyse des données peut être complexe.
- Annotation libre : chaque participant peut décrire librement les émotions suscitées par une œuvre.
    - Avantages : les données récoltées sont riches et détaillées et aucune contrainte n'est imposée aux participants.
    - Inconvénients : l'analyse des données est très complexe et chronophage et il peut y avoir une grande variabilité dans les réponses.

Pour comparer ces méthodes, les critères suivants ont été pris en compte et notés sur 5 :

- Facilité de mise en œuvre
- Richesse des données
- Temps requis pour compléter le sondage
- Simplicité d'utilisation pour les participants
- Précision des annotations
- Facilité de l'analyse des données
- Robustesse aux biais de réponse

Critères / Approches	Echelle de Likert	Comparaison de paires (TrueSkill)	Annotations libres
Facilité de mise en œuvre	5	3	4
Richesse des données	3	4	5
Temps requis	2	4	3
Simplicité pour les participants	5	4	3
Précision des annotations	3	4	4
Analyse des données	5	3	2
Robustesse aux biais de réponse	3	4	3
Score	**26**	**26**	24

Tableau 3 Comparatif des méthodes d'annotation des œuvres

Les résultats de l'analyse montrent que l'échelle de Likert et la comparaison de paires semblent être de bonnes options. Pour les séparer, j'ai décidé de choisir l'approche qui minimise le temps d'annotation requis par participant, soit la méthode TrueSkill.

L'annexe 10.3 « Choix de la méthode d'annotation des œuvres » détaille les notes attribuées dans le tableau ci-dessus.

3.5. Bibliothèques & Frameworks

Les bibliothèques et frameworks suivants seront utilisés pour l'implémentation des divers outils nécessaires au processus de sonification :

OpenCV: OpenCV est une bibliothèque libre spécialisée dans le traitement d'images. Elle propose diverses opérations de traitements d'images, de vidéos ainsi que des algorithmes d'apprentissage artificiel (OpenCV, 2023).

NumPy: NumPy est une bibliothèque pour le langage Python destinée à manipuler et effectuer des calculs mathématiques sur des tableaux et des matrices (NumPy, 2024).

Joblib: Joblib est une bibliothèque Python optimisée pour la persistance, la parallélisation et le chargement d'objets Python de grande taille (Joblib: running Python functions as pipeline jobs, n.d.).

TensorFlow: TensorFlow est un outil d'apprentissage automatique créé par Google (TensorFlow, 2024).

Scikit-image: Scikit-image est une bibliothèque de traitement d'image pour le langage Python (Scikit-image, 2020).

Scikit-learn: Scikit-learn est une bibliothèque d'apprentissage automatique pour le langage Python et comprend notamment de nombreux algorithmes (Scikit-learn, 2024).

Flask: Flask est un framework de développement web en Python, visant à garder un noyau simple, mais extensible (Flask (framework), 2024).

Pandas: Pandas est une bibliothèque de manipulation et d'analyse de données pour le langage Python (Pandas, 2024).

4. Implémentation & Évaluation

4.1. Méthodologie

La méthodologie Agile a été utilisée tout au long de ce travail pour la gestion de projet. Une séance initiale avec M. Henning Müller a permis de clarifier le cadre du travail et de décider de réunions hebdomadaires qui permettront de discuter des progrès réalisés, d'identifier les obstacles éventuels et de planifier les tâches à accomplir avant la prochaine séance.

L'analyse initiale des étapes du projet a conduit à la création du « Product Backlog » qui permet de structurer le travail à travers les différents sprints. Il a été décidé que les sprints dureraient 2 semaines (6 jours de travail à temps partiel).

Dans un second temps, une séance avec M. Henning Müller et Mme Valérie Félix a été organisée. Mme Valérie Félix, historienne de l'art et responsable de la filière HEA de l'EDHEA, a partagé ses connaissances du monde de l'art et suggéré diverses voies qui pourraient être intéressantes à explorer. Cette rencontre a permis de clarifier les objectifs du projet et d'étudier certaines directions potentielles.

Pour maintenir une visibilité constante sur l'avancement des travaux, des « dailys personnels » ont été réalisés au début de chaque journée de travail pour évaluer le travail effectué et prévoir les tâches à réaliser. Au début de chaque nouveau sprint, un temps de « sprint planning » a été réservé pour réévaluer les priorités du « Product Backlog », sélectionner les user stories à traiter, en fonction de leur priorité et de leur nombre de story points, et évaluer le temps de travail nécessaire à la réalisation de chacune de ces tâches.

Tous ces éléments ont permis d'assurer une planification efficace des tâches à réaliser et un déroulement fluide du projet.

4.2. Architecture

Le processus complet de l'application consiste à analyser une œuvre, à en extraire certaines caractéristiques, et à utiliser ces caractéristiques pour générer une composition sonore. L'œuvre à analyser peut être chargée depuis la web app et, une fois la synthèse sonore terminée, un lecteur audio permet d'écouter et de télécharger la composition. La figure 22 illustre ce procédé.

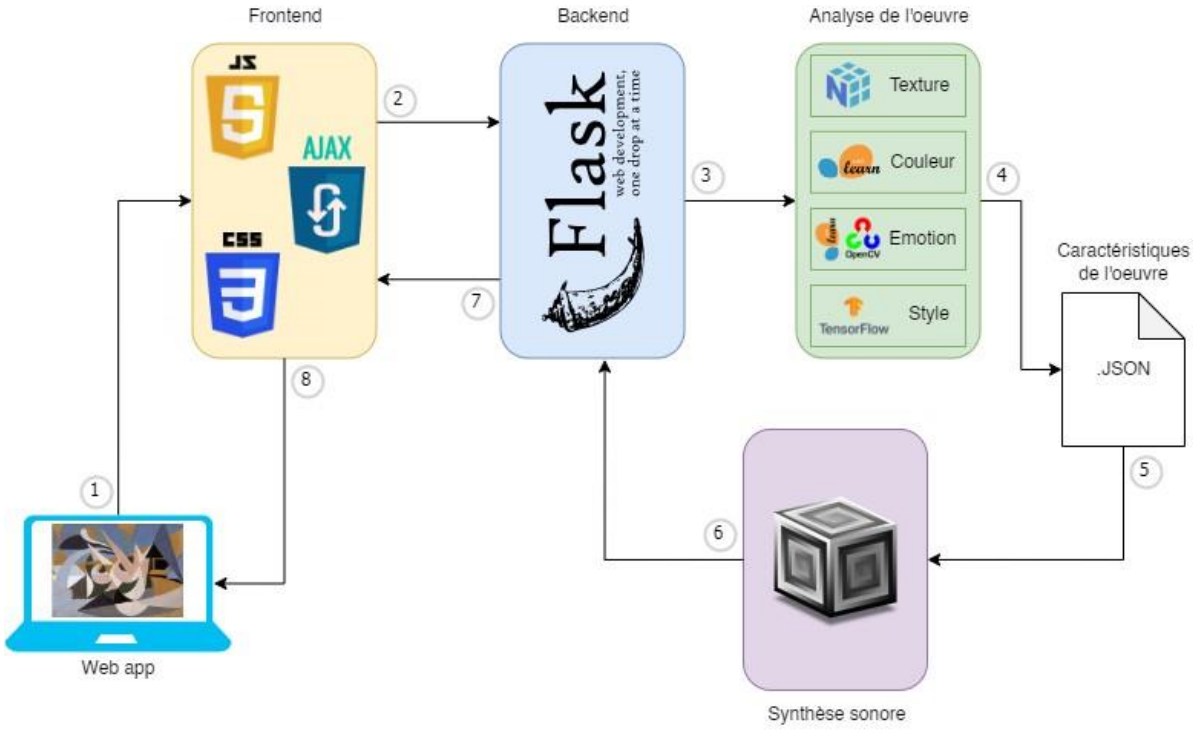


Figure 22 Structure du processus de sonification

Source : Données de l'auteur

Tous les outils nécessaires au bon fonctionnement de ce processus ont été installés sur un serveur Linux (Ubuntu 22.04.3 LTS, Kernel 5.15.0-116-generic, 4 CPUs Intel(R) Xeon(R) Platinum 8358 CPU @ 2.60 GHz, 100 GB de mémoire) accessible via SSH. WinSCP a été utilisé pour accéder au système de fichiers.

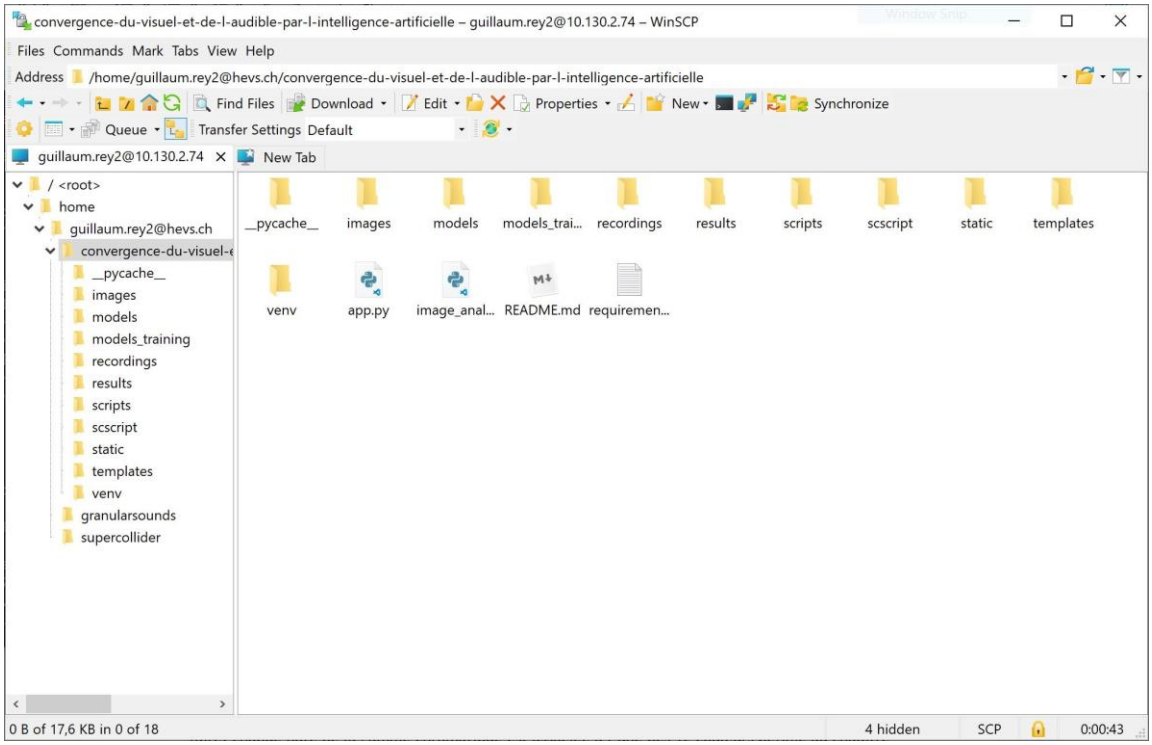


Figure 23 Structure du serveur

Source : Données de l'auteur

Le dossier « convergence-du-visuel-et-de-l-audible-par-l-intelligence-articielle » correspond au projet GitHub dédié à ce travail. Sa structure est la suivante :

- « **images** » est la zone de dépôt des œuvres à sonifier. Les images chargées via la web app y sont déposées.
- « **models** » englobe les différents modèles utilisés lors de l'analyse de l'œuvre.
- « **models_training** » contient les scripts Python utilisés pour entraîner les modèles d'analyse d'image.
- « **recordings** » est le dossier dans lequel sont enregistrés les fichiers audio lorsque la synthèse sonore est terminée.
- « **results** » contient les fichiers JSON générés lors de l'analyse d'images.
- « **scripts** » renferme les scripts Shell utilisés pour l'automatisation du processus de sonification
- « **scscripts** » englobe le script SuperCollider utilisé pour la sonification.
- « **static** » contient les fichiers CSS et JavaScript utilisés par la web app.

- « **templates** » abrite les fichiers HTML utilisés par la web app.
- « **venv** » est l'environnement virtuel Python.
- « **app.py** » est le script de configuration de la web app.
- « **image_analysis.py** » est le script d'analyse d'images qui fait appel aux divers modèles.
- « **requirements.txt** » contient toutes les librairies et frameworks Python nécessaires au projet.

Le dossier « **granularsounds** » contient le son utilisé par le synthétiseur « \granular » dans le patch SuperCollider, et le dossier « **supercollider** » abrite l'installation du logiciel de synthèse sonore. Un serveur audio y a également été installé pour pouvoir enregistrer les fichiers audio générés.

Un environnement virtuel (Python 3.11.9) a été mis en place sur le serveur. Il a permis d'avoir un environnement d'exécution isolé dans lequel toutes les librairies nécessaires ont été installées grâce au fichier « requirements.txt » qui se trouve à la racine du projet.

4.3. Datasets

Le processus de sonification comprenant plusieurs étapes d'analyse et d'extraction, différents jeux de données sont nécessaires pour l'entraînement et l'évaluation des modèles.

4.3.1. Modèle d'analyse des émotions

Pour assurer le bon fonctionnement du modèle sur un large panel d'images, il est nécessaire d'utiliser un jeu d'images annotées (Sartori, Culibrk, Yan, & Sebe, 2015). La diversité des styles est ici importante : abstrait, figuratif, impressionniste, etc.

Deux jeux de données ont été utilisés pour construire le dataset d'entraînement :

- **MART** : MART est une collection d'œuvres abstraites réalisées par 78 artistes professionnels entre 2013 et 2008 (Sartori & al., 2015).
- **ART500K** : ART500K est une large collection de plus de 500'000 œuvres annotées (style, artiste, etc.) (ART500K Dataset, 2019).

Un premier tri de ART500K a été réalisé pour ne garder que des peintures. Ensuite, une sélection aléatoire de 250 œuvres de ART500K et de 250 œuvres du dataset MART a été faite pour construire le jeu d'images final.

Pour l'annotation du jeu d'images, la méthode « TrueSkill » a été utilisée. Pour ce faire, une web app a été créée sur laquelle les participants devaient choisir, parmi deux images, laquelle évoquait l'émotion la plus agréable.

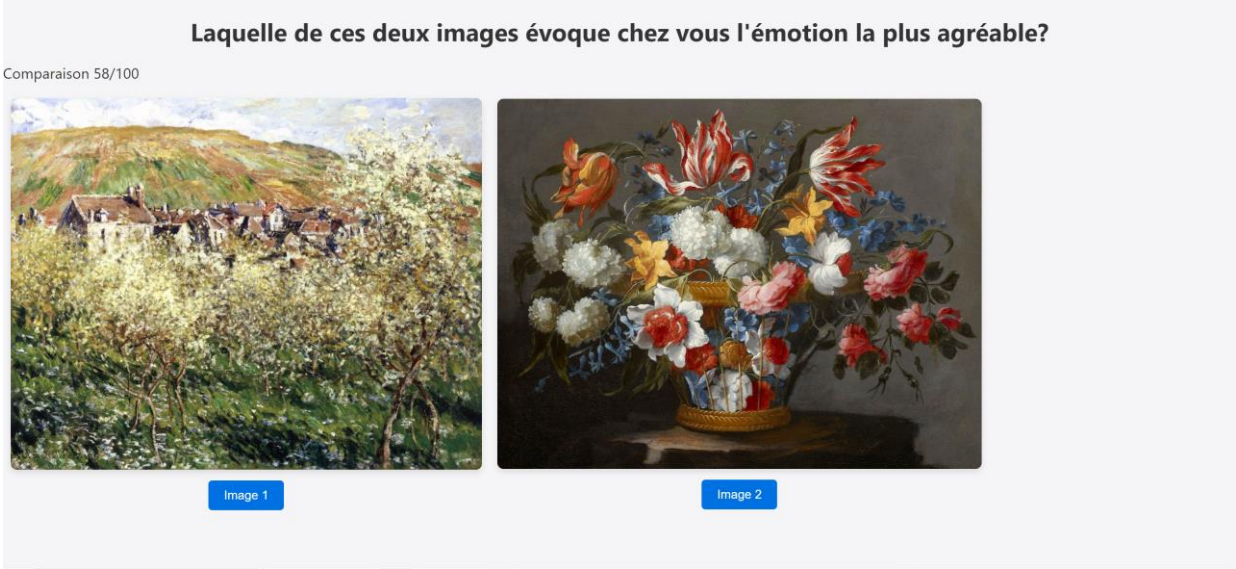


Figure 24 Web App pour l'annotation des images pour l'analyse des émotions

Source : Données de l'auteur

Toutes les images ont été initialisées avec les valeurs suivantes, recommandées par Microsoft (TrueSkill™ Ranking System, 2005): mu = 25,0 et sigma = 8,33. Pour rappel, mu correspond à la compétence et sigma à l'incertitude. Ces valeurs ont été mises à jour après chaque vote et le nombre de comparaisons de chaque image a également été enregistré dans la base de données. Pour s'assurer que toutes les images aient un nombre de comparaisons équitable, les nouvelles images ont été sélectionnées aléatoirement parmi celles qui avaient l'incertitude la plus haute.

Au total, 3153 comparaisons ont été effectuées par 37 participants (24 hommes et 13 femmes) âgés de 21 à 75 ans, et chaque image a été comparée entre 12 et 13 fois. Une fois les données exportées, un seuil de classification a été défini selon la médiane des valeurs de « mu » calculée à 24,996. Celle valeur a été choisie pour assurer une répartition équilibrée des données destinées à l'entraînement du modèle de classification d'émotion.

Au final, les valeurs de sigma se trouvent entre 2,49 et 4,14, ce qui indique que l'incertitude pour toutes les images a été réduite de façon conséquente.

Il est important de préciser que l'approche utilisée pour annoter ce jeu de données comporte certains biais. Si pour les œuvres abstraites seules les formes et les couleurs influencent réellement l'émotion perçue, les œuvres figuratives peuvent être plus difficiles à évaluer « objectivement ». Le vécu du participant peut fortement influencer la perception de l'image et l'émotion perçue. Le

nombre de personnes ayant participé à ce sondage n'est largement pas suffisant pour compenser ce biais.

4.3.2. Modèle de nommage des couleurs

Pour l'implémentation finale du modèle de nommage des couleurs, le jeu de données a été modifié par rapport à celui décrit au chapitre 3.2.3. Ces modifications ont été apportées suite à l'évaluation de certaines teintes à la frontière entre deux couleurs. Ainsi, certains échantillons correspondant à du « gris bleuté », par exemple, ont été relabellisés en bleu clair pour mieux correspondre à une perception humaine des couleurs.

Le dataset final contient 886 échantillons répartis comme suit : 7 échantillons de noir, 9 échantillons de blanc, 85 échantillons d'orange, 86 échantillons de rouge, de bleu clair et de bleu foncé, 87 échantillons de vert et 88 échantillons de gris, brun, rose, jaune et violet.

4.3.3. Modèle de classification du style

Le jeu d'images utilisé pour entraîner le modèle de classification du style a été construit sur la base du dataset WikiArt composé de 57'025 données d'entraînement et de 24'421 données de validation, dans 27 styles différents.

Suite aux premiers tests d'entraînements du modèle sur le dataset réduit aux 8 styles utilisés dans le cadre de ce travail, le nombre variable d'images par style s'est avéré être un problème. Pour réduire le temps d'entraînement et obtenir des résultats similaires pour tous les styles, le dataset a été réduit.

Le jeu d'images final est composé d'environ 4'000 données d'entraînement et de validation, avec une répartition plus équitable entre les divers styles artistiques.

4.3.4. Test de l'application

Les images utilisées pour tester l'application ont été sélectionnées parmi les images du dataset WikiArt ne faisant pas partie des jeux de données d'entraînement.

4.4. Lexique

Le lexique de sonification détaille toutes les relations entre les caractéristiques visuelles des œuvres d'art analysées et les caractéristiques sonores de la composition.

Utilisation de l'émotion

L'outil de reconnaissance d'émotion sépare les œuvres en deux catégories : émotion positive, ou agréable, et émotion négative. Bien que cette division binaire soit réductrice au vu de la complexité et du nombre d'émotions que l'être humain peut expérimenter, elle permet de diriger la composition vers l'un des deux principaux modes de la musique occidentale, à savoir le mode majeur généralement considéré comme « joyeux » et le mode mineur naturel souvent associé à la tristesse. Ainsi, une œuvre catégorisée comme « positive » engendra une composition majeure, alors qu'une œuvre « négative » fera appel au mode mineur naturel lors de la sonification.

Utilisation de la texture

Les 5 caractéristiques extraites grâce à l'analyse de la texture (il a été décidé de ne pas utiliser la linéarité) sont utilisées pour contrôler et moduler certains paramètres lors de la synthèse sonore. La granularité, ou « coarseness », influence le volume d'un module de synthèse granulaire, le contraste définit la quantité d'effet appliquée aux divers synthétiseurs, la directionnalité modifie le placement de certains éléments dans l'espace sonore, la régularité module l'enveloppe de volume des synthétiseurs et définit le nombre d'accords joués, et enfin la grossièreté, ou « roughness », définit un niveau de « désaccordage » et influe sur la complexité des accords joués. Les valeurs choisies pour ces variables ont été définies suite à l'analyse de diverses œuvres et à l'évaluation des résultats.

```
// EFFECTS AND MODULATION AMOUNT
granularSynthVolume = 0.008 + 2*(coarseness / 1000);
reverbAmount = (1 - contrast) + fx;
delayAmount = contrast/2 ;
detuneAmount = detune + (roughness / 4);
envModAmount = regularity * 2;
panningRange = 1 - directionality;
rate = ((1 - coarseness) * 4) + 1 ;
```

Figure 25 Utilisation de certaines caractéristiques de texture

Source : Données de l'auteur

Utilisation des couleurs

La couleur dominante de l'œuvre est utilisée pour définir la tonalité de la composition. Pour ce faire, chaque note de la gamme chromatique est associée à l'une des 12 couleurs définies au chapitre 3.2.3. Voici les relations choisies :

Note	Couleur	Numéro de note MIDI
Do (C)	Noir	60
Do dièse (C#)	Brun	61
Ré (D)	Bleu foncé	62
Mi bémole (Eb)	Violet	63
Mi (E)	Rouge	64
Fa (F)	Orange	65
Fa dièse (F#)	Vert	66
Sol (G)	Gris	67
La bémole (Ab)	Bleu clair	68
La (A)	Rose	69
Si bémole (Bb)	Jaune	70
Si (B)	Blanc	71

Tableau 4 Correspondance entre les notes, les couleurs et les valeurs MIDI

Ce choix ne peut être justifié de manière totalement logique et objective. D'une part, les plages fréquentielles des couleurs et des notes sont très différentes, une relation fréquentielle directe ne peut donc pas être établie. Une autre approche pourrait consister à faire correspondre, par exemple, une couleur sombre à une note basse, mais cette même note peut être jouée à une octave supérieure et la correspondance n'a alors plus lieu d'être. Il a également été discuté au chapitre 3.2.3 que l'étude de la synesthésie ne permet pas de définir une échelle, car les synesthètes musique-couleur ne s'accordent pas sur la couleur perçue pour un son donné.

Ainsi, les relations couleur-note ont été décidées selon la luminosité de chacune des couleurs, de la plus sombre à la plus claire : noir, brun, bleu foncé, violet, rouge, orange, vert, gris, bleu clair, rose, jaune, blanc. Une fois cet ordre établi, il a été arbitrairement choisi que le noir (première couleur de ma liste) correspondrait au Do (première note de la gamme telle qu'elle est généralement définie).

Utilisation du style

Pour ce projet, 8 styles artistiques sont utilisés. Pour rappel, ce choix a été fait en fonction des jeux de données annotés disponibles et de la littérature existante sur l'analyse du style dans les œuvres d'art. Les 8 styles sont l'art abstrait, le cubisme, la peinture en champs de couleur, le réalisme, le romantisme, le symbolisme, l'expressionnisme et l'impressionnisme. Un « patch » de synthèse sonore unique a été créé pour chaque style en prenant en compte certaines particularités de chacun de ces mouvements artistiques. Pour définir ces caractéristiques, je me suis principalement basé sur les descriptifs du magazine « Artland » qui a produit une liste extensive décrivant les nombreux mouvements et styles artistiques et leurs particularités. Ces attributs sont discutés ci-dessous et quelques adjectifs sont attribués à chaque style. Ces adjectifs seront considérés lors du développement du script de synthèse sonore pour assurer une cohérence esthétique entre l'image et le son.

Art abstrait : ce mouvement abandonne la figuration en faveur de l'abstraction. La spontanéité, l'expression d'émotions et les thèmes universels sont au cœur de ce mouvement (Abstract Expressionism, n.d.).

Adjectifs : *abstrait, spontané, énergique, dynamique*

Cubisme : le rejet de la perspective traditionnelle est au centre du cubisme. Si beaucoup de styles antérieurs tentent de créer l'illusion d'un espace tridimensionnel, la bidimensionnalité de la toile est ici accentuée (Cubism, n.d.).

Adjectifs : *bidimensionnel, géométrique, fragmenté, abstrait*

Peinture en champs de couleur : les compositions de ce mouvement préfèrent des compositions plates, sans profondeur illusoire. Ces peintures utilisent généralement des formes simples et de grandes étendues de couleurs unies (Art Movement : Color Field Painting, n.d.).

Adjectifs : *épuré, monolithique, vibrant, minimaliste, immersif*

Réalisme : au centre de ce mouvement se trouve l'envie de représenter des situations banales, des scènes de la vie courante sans idéalisation ou embellissement. L'apparition de la photographie a joué un rôle dans l'envie des artistes réalistes de représenter la réalité avec une précision accrue (Réalisme (peinture), 2024).

Adjectifs : *brut, précis, radical, détaillé*

Romantisme : le romantisme privilégie l'émotion et l'imagination sur la raison et la rigueur classique, et les thèmes fantastiques, les rêves et le mysticisme sont omniprésents (Peinture romantique, 2024).

Adjectifs : sombre, mystique, fantastique

Symbolisme : le symbolisme s'oppose au réalisme, préférant l'expression subjective des émotions. L'âme humaine, les rêves et le subconscient sont des thèmes récurrents et des significations cachées sont souvent exprimées à travers des métaphores ou allégories (Art movement : Symbolism, n.d.).

Adjectifs : *onirique, mystérieux, subtil, rêveur, fantastique*

Expressionisme : dans l'expressionnisme, la réalité physique est mise de côté au profit de l'expression d'émotions profondes. Ces aspects émotionnels sont accentués par l'utilisation de couleurs vives et la représentation distordue de formes et de figures humaines (Art movement : Expressionism, n.d.).

Adjectifs : *intense, vibrant, brut, mystique, émotionnel, distordu, exagéré*

Impressionnisme : l'impressionnisme est né en réaction à l'art académique traditionnel, et les artistes impressionnistes préféraient travailler en extérieur, dans la nature, pour capturer l'aspect changement de la lumière. Pour représenter cet effet transitoire, les coups de pinceaux sont généralement visibles et rapides (Art Movement : Impressionism, n.d.).

Adjectifs : *lumineux, dynamique, expressif, subversif, innovant*

4.5. Analyse d'image

Nous allons dans ce chapitre détailler l'implémentation des divers outils utilisés pour l'analyse d'une œuvre. Le pipeline final est fait de plusieurs étapes. Dans un premier temps, les modèles entraînés et l'œuvre à analyser sont chargés.

```
# Set the model path relative to the script location
MODEL_PATH = os.path.join(os.path.dirname(__file__), 'models')

# Load trained models and resources
color_model = joblib.load(os.path.join(MODEL_PATH, 'color_classifier_model.pkl'))
color_encoder = joblib.load(os.path.join(MODEL_PATH, 'label_encoder.pkl'))
emotion_model = joblib.load(os.path.join(MODEL_PATH, 'emotion_classifier_model.pkl'))
color_palette = np.load(os.path.join(MODEL_PATH, 'color_palette.npy'))
art_style_model = tf.keras.models.load_model(os.path.join(MODEL_PATH, 'art_style_model.h5'))
```

Figure 26 Chargement des modèles d'analyse d'image

Source : Données de l'auteur

```
print(f"Analyzing image: {image_path}")
image = cv2.imread(image_path)
if image is None:
    print("Error: Image not found.")
    return

base_filename = os.path.splitext(os.path.basename(image_path))[0]
```

Figure 27 Chargement de l'œuvre

Source : Données de l'auteur

Les caractéristiques de l'œuvre sont ensuite extraites selon l'ordre suivant : analyse de la texture, extraction de la couleur dominante, nommage de la couleur dominante, classification de l'émotion dominante, classification du style de la peinture et enregistrement des caractéristiques dans un fichier JSON.

Les sous-chapitres suivants détaillent chacune de ces étapes.

4.5.1. Texture

L'analyse de la texture de l'œuvre est réalisée en plusieurs étapes décrites ci-après. Les méthodes de calcul adoptées ici sont fortement inspirées de l'implémentation mise à disposition par Lee (2019). La bibliothèque NumPy (« np » dans le code) est utilisée pour la majorité des calculs impliquant des tableaux et des matrices.

Prétraitement

Un prétraitement est appliqué pour simplifier les calculs et standardiser l'analyse de texture. Dans un premier temps, l'image est convertie en niveaux de gris, puis redimensionnée (en conservant le ratio de l'image originale), et enfin floutée avec un flou gaussien pour réduire le bruit et les détails trop fins. Ces étapes permettent de faciliter et d'accélérer l'analyse.

```python
# Function to preprocess the image for texture analysis
def tamura_preprocess_image(image, target_size=(256, 256)):
    gray_img = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    h, w = gray_img.shape[:2]
    target_w, target_h = target_size
    aspect_ratio = w / h

    if aspect_ratio > 1:  # Width is greater, resize by width
        new_w = target_w
        new_h = int(new_w / aspect_ratio)
    else:  # Height is greater, resize by height
        new_h = target_h
        new_w = int(new_h * aspect_ratio)

    resized_img = cv2.resize(gray_img, (new_w, new_h))
    smooth_img = cv2.GaussianBlur(resized_img, (5, 5), 0)
    return smooth_img
```

Figure 28 Prétraitement de l'image pour l'analyse de texture

Source : Données de l'auteur

Coarseness

La granularité de la texture est mesurée en calculant la moyenne des niveaux de gris sur différentes tailles de fenêtres, puis en évaluant les variations horizontales et verticales de ces moyennes (Brown, et al., 2021). Les gradients horizontaux et verticaux, qui sont des vecteurs mesurant les changements d'intensité des pixels, sont ensuite normalisés et utilisés pour déterminer la taille de fenêtre la plus significative pour chaque pixel . La granularité est finalement obtenue en moyennant ces valeurs sur l'ensemble de l'image. Cette méthode permet de quantifier à quel point les variations de texture sont grossières ou fines.

```python
# Function to calculate coarseness
def coarseness(image, kmax):
    w, h = image.shape
    kmax = min(kmax, int(np.log2(w)), int(np.log2(h)))
    average_gray = np.zeros([kmax, w, h])
    horizon = np.zeros([kmax, w, h])
    vertical = np.zeros([kmax, w, h])
    Sbest = np.zeros([w, h])

    for k in range(kmax):
        window = np.power(2, k)
        for wi in range(window, w-window):
            for hi in range(window, h-window):
                average_gray[k, wi, hi] = np.mean(image[wi-window:wi+window+1, hi-window:hi+window+1])
        for wi in range(window, w-window):
            for hi in range(window, h-window):
                horizon[k, wi, hi] = average_gray[k, wi+window, hi] - average_gray[k, wi-window, hi]
                vertical[k, wi, hi] = average_gray[k, wi, hi+window] - average_gray[k, wi, hi-window]
        horizon[k] *= (1.0 / (2 * window))
        vertical[k] *= (1.0 / (2 * window))

    for wi in range(w):
        for hi in range(h):
            h_max = np.max(horizon[:, wi, hi])
            v_max = np.max(vertical[:, wi, hi])
            Sbest[wi, hi] = np.power(2, np.argmax(horizon[:, wi, hi] >= v_max))

    fcrs = np.mean(Sbest)
    return fcrs
```

Figure 29 Calcul de la granularité (coarseness)

Source : Données de l'auteur

Contraste

Le contraste de l'image est évalué en utilisant la distribution des niveaux de gris. « np.mean((image – np.mean(image))**4) » calcule la moyenne des valeurs de pixels, puis évalue la déviation de chaque pixel par rapport à cet moyenne. Cette déviation est élevée à la puissance 4 pour obtenir le quatrième moment centré (qui permet de quantifier la distribution des valeurs d'une variable autour de sa moyenne) (Brown, et al., 2021). La valeur de « alfa4 » est un indicateur de forme : une valeur élevée indique une distribution leptokurtique, alors qu'une valeur basse met en lumière une distribution platykurtique (Leptokurtique examen de la fonction generatrice de moments et de son role, 2024).

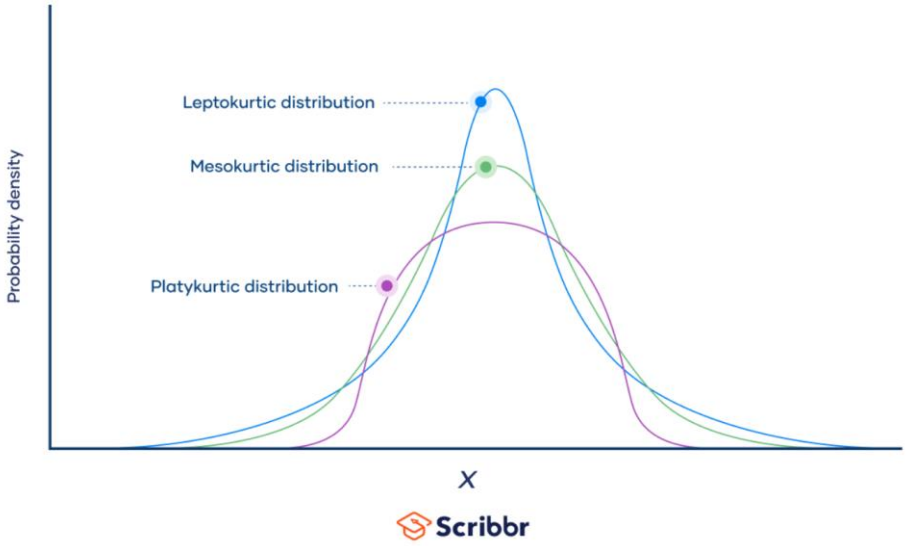


Figure 30 Types de kurtosis

Source : https://www.scribbr.com/statistics/kurtosis/

Ainsi, la valeur de « alfa4 » permet d'affiner la mesure du contraste en tenant compte de la variante des intensités, mais également de la forme de la distribution.

```python
# Function to calculate contrast
def contrast(image):
    image = np.array(image, dtype=np.float32).flatten()
    m4 = np.mean((image - np.mean(image))**4)
    v = np.var(image)
    std = np.sqrt(v)

    if v == 0:
        alfa4 = 0
    else:
        alfa4 = m4 / (v**2)

    if alfa4 == 0:
        fcon = 0
    else:
        fcon = std / (np.power(alfa4, 0.25))

    return fcon
```

Figure 31 Calcul du contraste

Source : Données de l'auteur

Directionnalité

La directionnalité est évaluée en utilisant les filtres de Sobel pour calculer les gradients de l'image dans les directions x et y (Brown, et al., 2021). Le gradient d'une image à un point donné est un vecteur qui indique la direction du plus grand changement d'intensité lumineuse (Brown, et al., 2021). Un histogramme est construit pour capturer la distribution des orientations dans l'image. Ces orientations sont regroupées en 36 « bins », où chaque bin représente 5 degrés, et pondérées par la magnitude. L'orientation dominante est dérivée de cet histogramme en évaluant le « bin » avec la valeur maximale.

```python
# Function to calculate directionality
def directionality(image):
    Gx = cv2.Sobel(image, cv2.CV_64F, 1, 0, ksize=5)
    Gy = cv2.Sobel(image, cv2.CV_64F, 0, 1, ksize=5)
    magnitude = np.sqrt(Gx**2 + Gy**2)
    orientation = np.arctan2(Gy, Gx) * (180 / np.pi) % 180
    histogram, bins = np.histogram(orientation, bins=36, range=(0, 180), weights=magnitude)

    max_index = np.argmax(histogram)
    direction_strength = histogram[max_index] / np.sum(histogram) if np.sum(histogram) != 0 else 0

    return direction_strength
```

Figure 32 Calcul de la directionnalité

Source : Données de l'auteur

Régularité

La régularité est mesurée en divisant l'image en blocs et en calculant la granularité et le contraste de chaque bloc. La régularité est ensuite définie comme la moyenne des écarts-types de ces valeurs sur l'ensemble des blocs. Cela permet de quantifier à quel point les variations de texture sont uniformes ou irrégulières.

```python
# Function to calculate regularity
def regularity(image, block_size=32):
    scores = []
    for i in range(0, image.shape[0], block_size):
        for j in range(0, image.shape[1], block_size):
            block = image[i:i+block_size, j:j+block_size]
            if block.shape[0] == block_size and block.shape[1] == block_size:
                scores.append((coarseness(block, 5), contrast(block)))
    if scores:
        scores = np.array(scores)
        std_devs = np.std(scores, axis=0)
        return np.mean(std_devs)
    else:
        return 0
```

Figure 33 Calcul de la régularité

Source : Données de l'auteur

Roughness

La « grossièreté » est simplement la somme de la granularité et du contraste.

```
# Function to calculate roughness
def roughness(fcrs, fcon):
    return fcrs + fcon
```

Figure 34 Calcul de la grossièreté

Source : Données de l'auteur

Normalisation

Finalement, toutes valeurs des caractéristiques sont mises à l'échelle afin qu'elles se situent dans une plage de valeurs commune (entre 0 et 1), facilitant ainsi leur utilisation au moment de la synthèse sonore. Pour définir les plages de valeur initiales, 500 œuvres ont été analysées et les résultats suivants ont été observés :

```
# Normalization ranges based on data analysis
COARSENESS_RANGE = (1, 4)
CONTRAST_RANGE = (0, 100)
DIRECTIONALITY_RANGE = (0, 0.5)
ROUGHNESS_RANGE = (0, 100)
REGULARITY_RANGE = (0, 30)
```

Figure 35 Plages de valeur des caractéristiques de texture

Source : Données de l'auteur

Si une nouvelle image a une valeur inférieure ou supérieure à celles définies par la plage, cette valeur est rognée.

4.5.2. Couleur

Comme il a été discuté au chapitre 3.2.3, l'algorithme KMeans est utilisé pour extraire la couleur dominante de l'œuvre. Ce processus d'extraction fait appel aux bibliothèques OpenCV, NumPy, Pandas, Joblib et Scikit-learn (silhouette_score, KMeans, RandomForestClassifier et train_test_split).

Ainsi, pour l'extraction de la couleur dominante de l'œuvre, les étapes suivantes ont été implémentées :

Prétraitement

Comme vu précédemment pour l'analyse de texture, un prétraitement permet d'optimiser l'analyse. Dans un premier temps l'image est convertie de l'espace de couleur BGR (utilisé par défaut par OpenCV) à l'espace de couleur RGB. Ensuite, l'image est floutée à l'aide d'un filtre gaussien pour réduire le bruit et les petites variations de couleur, rendant ainsi les clusters de couleurs plus distincts. Pour accélérer le processus, l'image est redimensionnée pour réduire le nombre de pixels à traiter sans perdre significativement les informations de couleur.

```python
# Function to preprocess the image for color analysis
def color_preprocess_image(image):
    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    blurred_image = cv2.GaussianBlur(image_rgb, (5, 5), 0)
    resized_image = cv2.resize(blurred_image, (250, 250))
    return resized_image
```

Figure 36 Prétraitement pour l'extraction de la couleur dominante

Source : Données de l'auteur

Filtrage des pixels noirs

Les pixels noirs peuvent souvent représenter des arrière-plans ou des éléments non pertinents dans l'analyse de couleurs dominantes, et donc, avant d'appliquer l'algorithme de clustering, il est important de considérer la proportion de pixels noirs dans l'image. Cependant, dans certains cas, le noir peut être une couleur importante dans l'œuvre. Pour gérer cette situation, la proportion de pixels noirs est calculée, et si elle est inférieure à un certain seuil, défini ici à 30 %, ces pixels sont filtrés. Ce filtrage conditionnel garantit que les pixels noirs sont pris en compte uniquement lorsqu'ils représentent une part significative de l'image.

```python
# Function to filter out dark pixels
def filter_dark_pixels(pixel_values, dark_threshold=30, black_threshold=0.3):
    dark_pixels = np.all(pixel_values < [dark_threshold, dark_threshold, dark_threshold], axis=1)
    dark_proportion = np.sum(dark_pixels) / pixel_values.shape[0]
    print(f"Proportion of dark pixels: {dark_proportion}")

    if (dark_proportion < black_threshold):
        pixel_values = pixel_values[~dark_pixels]

    return pixel_values
```

Figure 37 Calcul de la proportion de pixels noirs

Source : Données de l'auteur

La variable « dark_threshold » correspond au seuil utilisé pour évaluer si une valeur RGB est noire ou non.

Détermination du nombre de clusters

Le choix du nombre de clusters pour l'algorithme KMeans est une étape critique pour garantir une segmentation de couleur optimale. Dans le prototype initial, ce nombre était fixe, ce qui n'est pas idéal pour toutes les œuvres, en fonction du nombre de couleurs qui les compose et de leur complexité. Dans le pipeline final, le nombre de clusters et déterminé dynamiquement en utilisant le score de silhouette, qui évalue la cohésion des points au sein des clusters et la séparation entre les différents clusters (Selecting the number of clusters with silhouette analysis on KMeans clustering, n.d.). En itérant sur un éventail de valeurs possibles (de 2 à un maximum prédéfini), le score de silhouette est évalué pour chaque valeur. Il peut être calculé par *(b-a) / max (a, b)* où « a » est la distance moyenne intra-cluster, soit la distance moyenne entre un échantillon et tous les autres points du cluster, et « b » est la distance moyenne au cluster le plus proche (silhouette_score, n.d.).

Le nombre de clusters qui maximise le score de silhouette est choisi comme optimal. Ce processus assure que le clustering reflète fidèlement la distribution des couleurs dans l'image.

```python
# Function to find the optimal number of clusters for KMeans
def find_optimal_clusters(pixel_values, max_k=10):
    best_k = 2
    best_score = -1
    for k in range(2, max_k + 1):
        kmeans = KMeans(n_clusters=k, random_state=0).fit(pixel_values)
        labels = kmeans.labels_
        score = silhouette_score(pixel_values, labels)
        if score > best_score:
            best_score = score
            best_k = k
    return best_k
```

Figure 38 Calcul du nombre de clusters optimal

Source : Données de l'auteur

Application de l'algorithme KMeans

Une fois le nombre optimal de clusters déterminé, l'algorithme KMeans est appliqué pour segmenter les pixels de l'image. Chaque cluster représente une couleur dominante potentielle et chaque pixel est attribué à l'un d'entre eux en minimisant la distance entre le pixel et le centre du cluster. Les centres de clusters finaux représentent les couleurs dominantes dans l'image. Ces couleurs sont triées en fonction de la taille de leur cluster, c'est-à-dire du nombre de pixels assignés à chacun

d'entre eux. La couleur avec le plus grand nombre de pixels est identifiée comme la couleur dominante.

Le seuil de pixels noirs, le nombre de clusters maximal et d'autres paramètres ont été définis suite à des tests effectués sur une petite batterie d'images dans des styles variés.

Entraînement du modèle de nommage

L'ensemble des données est regroupé dans un fichier CSV qui est chargé en utilisant la bibliothèque Pandas. Le fichier CSV contient des valeurs RGB pour les 12 couleurs qui nous intéressent ainsi que leurs étiquettes correspondantes. Les colonnes représentant les valeurs RGB sont extraites pour constituer les caractéristiques (X), tandis que la colonne des étiquettes de couleur est utilisée comme variable cible (y). Les labels sont également encodés.

Les données sont divisées en ensembles d'entraînement et de test pour permettre d'entraîner le modèle sur un sous-ensemble des données et de l'évaluer sur un autre sous-ensemble, évitant ainsi le surapprentissage et fournissant une estimation plus fiable de la performance du modèle. Le ratio de 80 % pour l'entraînement et de 20 % pour le test est utilisé.

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2, random_state=42)
```

Figure 39 Séparation des données en données d'entraînement et de test

Source : Données de l'auteur

Un modèle Random Forest est ensuite utilisé et initié avec une valeur de 100 pour « n_estimators » qui correspond au nombre d'arbres dans la forêt, est une valeur de « random_state » est précisée pour garantir la reproductibilité des résultats (RandomForestClassifier, n.d.).

```
# Initialize and train the Random Forest classifier
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
```

Figure 40 Initialisation du classificateur Random Forest

Source : Données de l'auteur

Utilisation du modèle de nommage

Le modèle entraîné est appelé, avec comme paramètre la valeur RGB extraite à l'étape de détection de la couleur dominante. Une fois le processus terminé, l'encodeur de label est utilisé pour convertir la valeur prédite en une chaîne de caractères correspondant au nom de la couleur.

```python
# Function to predict the color label
def predict_color_label(dominant_color):
    print("Predicting color label")
    dominant_color_rgb = np.uint8([dominant_color]).reshape(1, -1)
    print(f"RGB value sent for prediction: {dominant_color_rgb}")
    prediction = color_model.predict(dominant_color_rgb)
    color_label = color_encoder.inverse_transform(prediction)
    return color_label[0]
```

Figure 41 Utilisation du modèle de nommage de couleurs

Source : Données de l'auteur

4.5.3. Émotion

Il a été décidé au chapitre 3.2.1 d'utiliser la publication « Who's afraid of Itten: Using the art theory of color combination to analyze emotions in abstract paintings » de Sartori & al. (2015a) comme base de travail pour l'implémentation du modèle de classification des émotions. Ce travail repose sur l'utilisation du SGL pour l'étape de classification. Après différentes tentatives infructueuses d'intégration de la bibliothèque « group-lasso » existante, deux alternatives ont été testées : la régression logistique et l'arbre de décision. La seconde approche fournissant de meilleurs résultats, c'est elle qui a été utilisée dans le pipeline final.

Ce processus fait appel aux bibliothèques OpenCV, NumPy, Pandas, Scikit-learn et Joblib.

Entraînement du modèle

Pour commencer, toutes les valeurs RGB des images contenues dans le jeu de données sont évaluées. Un sous-échantillonnage est effectué si le nombre total de valeurs RGB dépasse 500'000. Cela permet de réduire la charge computationnelle tout en conservant une bonne représentation des couleurs présentes dans le dataset.

```python
# Function to extract RGB values from images in a dataset
def extract_rgb_values(dataset_path, sample_size=500000):
    print("Extracting RGB values from images...")
    all_rgb_values = []
    for filename in os.listdir(dataset_path):
        if filename.lower().endswith(('.png', '.jpg', '.jpeg')):
            image_path = os.path.join(dataset_path, filename)
            image = cv2.imread(image_path)
            if image is not None:
                image = augment_image(image)
                all_rgb_values.append(image.reshape(-1, 3))
            else:
                print(f"Warning: Image {filename} could not be loaded.")
    all_rgb_values = np.vstack(all_rgb_values) if all_rgb_values else np.empty((0, 3))
    if all_rgb_values.shape[0] > sample_size:
        indices = np.random.choice(all_rgb_values.shape[0], sample_size, replace=False)
        all_rgb_values = all_rgb_values[indices]
    return all_rgb_values
```

Figure 42 Extraction des valeurs RGB du jeu de données

Source : Données de l'auteur

Les valeurs RGB sont ensuite regroupées en clusters en utilisant l'algorithme KMeans. Cela permet de réduire la complexité des données en identifiant les couleurs dominantes dans le dataset, formant ainsi une palette de 180 couleurs, soit le nombre maximal de couleurs dans le modèle d'Itten (Sartori & al., 2015a). La palette obtenue est sauvegardée pour être utilisée ultérieurement lors de la classification de nouvelles images.

```python
# Function to extract a color palette using K-means clustering
def extract_color_palette(rgb_values, n_clusters=180):
    print("Starting K-means clustering...")
    if not rgb_values.size:
        return np.empty((0, 3))
    kmeans = KMeans(n_clusters=n_clusters, random_state=0)
    kmeans.fit(rgb_values)
    print("K-means clustering completed.")
    return kmeans.cluster_centers_
```

Figure 43 Extraction de la palette de couleur

Source : Données de l'auteur

Cette palette est utilisée pour repeindre les œuvres analysées. Les images repeintes sont ensuite segmentées. Ici, l'implémentation existante de l'algorithme de Felzenszwalb est utilisée pour segmenter l'image en régions homogènes (en termes de couleur). Chaque segment représente une région de l'image avec des caractéristiques similaires, facilitant ainsi l'extraction d'attributs pertinents pour le modèle d'analyse d'émotion. La variable « sigma » définit ici la largeur du noyau gaussien utilisé pour le prétraitement, et « min_size » définit la taille minimale des segments. Ceux

qui sont plus petits que cette valeur seront fusionnés avec leurs voisins. Ces valeurs sont celles suggérées par Sartori & al. (2015a).

```python
# Function to segment an image using the Felzenszwalb method
def segment_image(image):
    print("Segmenting image...")
    segments = felzenszwalb(image, scale=100, sigma=0.8, min_size=50)
    print("Segmentation completed.")
    return segments
```

Figure 44 Segmentation de l'image par l'algorithme Felzenszwalb

Source : Données de l'auteur

De ces segments sont extraites des caractéristiques (couleur moyenne et taille du segment notamment) qui sont par la suite normalisées pour assurer une échelle similaire entre tous ces attributs. Ces propriétés encapsulent des informations visuelles pertinentes qui influencent la perception émotionnelle des images et sont donc cruciales à ce processus. Pour chaque image, toutes les caractéristiques sont « aplaties » en un seul vecteur.

```python
# Function to extract adjacent color features from segmented image
def extract_adjacent_color_features(image, segments):
    print("Extracting adjacent color features...")
    num_segments = np.max(segments) + 1
    feature_vector_size = 10
    feature_vectors = np.zeros((num_segments, feature_vector_size))
    for seg_id in range(num_segments):
        mask = segments == seg_id
        if np.any(mask):
            segment_color = np.mean(image[mask], axis=0)
            segment_area = np.sum(mask)
            feature_vectors[seg_id, :3] = segment_color
            feature_vectors[seg_id, 3] = segment_area
    return feature_vectors.flatten() if feature_vectors.size else np.zeros(feature_vector_size * num_segments)
```

Figure 45 Extraction des caractéristiques des segments

Source : Données de l'auteur

Finalement, les vecteurs de caractéristiques et de labels (annotations du jeu de données) sont stockés dans des listes qui sont ensuite converties en tableaux NumPy. Ces tableaux sont stockés dans des blocs de mémoire voisins, facilitant ainsi l'accès en lecture et offrant une solution plus optimale que des listes, par exemple. Un modèle de classification d'arbre de décision est initialisé et entraîné sur les données préalablement divisées en données d'entraînement et de test.

```python
# Pad feature vectors to have the same length
max_features_length = max(len(f) for f in features_list)
features_list = [np.pad(f, (0, max_features_length - len(f)), 'constant') for f in features_list]

# Convert lists to numpy arrays
X = np.array(features_list)
y = np.array(labels_list)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Figure 46 Préparation des données pour l'entraînement du modèle

Source : Données de l'auteur

Les premiers résultats du modèle entraîné sur le dataset décrit au chapitre 4.3.1 n'étant pas satisfaisants, une étape d'augmentation de données a été ajoutée.

```python
# Function to apply data augmentation on an image
def augment_image(image):
    print("Applying data augmentation...")
    if np.random.rand() > 0.5:
        # Randomly rotate the image
        angle = np.random.randint(-15, 15)
        (h, w) = image.shape[:2]
        center = (w // 2, h // 2)
        M = cv2.getRotationMatrix2D(center, angle, 1.0)
        image = cv2.warpAffine(image, M, (w, h))
    if np.random.rand() > 0.5:
        # Randomly flip the image horizontally
        image = cv2.flip(image, 1)
    if np.random.rand() > 0.5:
        # Randomly flip the image vertically
        image = cv2.flip(image, 0)
```

Figure 47 Exemples de méthodes d'augmentation

Source : Données de l'auteur

Ces augmentations sont appliquées de manière probabiliste, chacune d'entre elles n'est donc pas appliquée à chaque image. Les augmentations suivantes peuvent être employées : rotation aléatoire, retournement horizontal, retournement vertical, ajout de bruit gaussien ou ajustement des valeurs HSV.

Utilisation du modèle entraîné

Dans le processus final, les étapes de préparation de l'image discutées ci-dessus sont appliquées à l'œuvre, puis un vecteur de caractéristiques, généré en segmentant l'image puis en calculant les couleurs moyennes et la taille des segments, est passé au modèle entraîné.

```
repainted_image = repaint_image_with_palette(image, color_palette)
segments = segment_image(repainted_image)
features = extract_adjacent_color_features(repainted_image, segments)
expected_feature_length = 39880
if len(features) < expected_feature_length:
    features = np.pad(features, (0, expected_feature_length - len(features)), 'constant')
else:
    features = features[:expected_feature_length]

features = features.reshape(1, -1)

predicted_emotion = emotion_model.predict(features)
emotion_label = 'positive' if predicted_emotion == 1 else 'negative'
```

Figure 48 Préparation de l'image et utilisation du modèle de classification d'émotions

Source : Données de l'auteur

4.5.4. Style

Pour rappel, il a été décidé au chapitre 3.2.4 d'utiliser le travail de Lafay & al. (2021) comme base de travail pour notre modèle de classification de style.

Ce processus fait appel aux bibliothèques NumPy, Pandas, TensorFlow (Keras) et Scikit-learn.

Entraînement du modèle

Pour commencer, les données d'entraînement et de validation sont chargées. Le dataset original n'a pas de noms de colonnes, « .columns » est donc utilisé pour assigner un nom à la colonne qui contient le nom du style.

```python
# Load training and validation data
train_df = pd.read_csv(os.path.join(csv_directory, 'small_style_train.csv'), header=None)
val_df = pd.read_csv(os.path.join(csv_directory, 'small_style_val.csv'), header=None)

# Set column names
train_df.columns = ['filename', 'style']
val_df.columns = ['filename', 'style']

# Convert style labels to string type
train_df['style'] = train_df['style'].astype(str)
val_df['style'] = val_df['style'].astype(str)
```

Figure 49 Chargement des données et définition du nom des colonnes

Source : Données de l'auteur

Une série d'augmentations est ensuite définie pour chaque image analysée. Pour les données de validation, les œuvres sont simplement redimensionnées. Pour celles d'entraînement, des rotations, des décalages et des zooms sont utilisés.

```python
# Define data augmentation for training images
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

# Define data augmentation for validation images (only rescaling)
val_datagen = ImageDataGenerator(rescale=1./255)
```

Figure 50 Définition des augmentations de données

Source : Données de l'auteur

Le VGG16 pré-entraîné sur ImageNet est chargé. Seules les couches convolutives sont utilisées (include_top=False), mais différentes couches personnalisées sont ajoutées par la suite (tf.keras.applications.VGG16, 2024). Les poids des différentes couches, soit les paramètres que le modèle apprend, ne sont pas mis à jour pendant l'entraînement pour ne pas altérer les caractéristiques apprises sur ImageNet, mais diverses couches denses sont ajoutées. Ces dernières prennent les caractéristiques apprises sur ImageNet et les utilisent pour faire des prédictions sur les nouvelles classes. « Dropout » permet d'éviter le surapprentissage en désactivant aléatoirement

certaines neurones et le « softmax » est utilisé pour calculer les probabilités de chaque classe (tf.keras.applications.VGG16, 2024).

```
# Load the VGG16 base model without the top layer
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

# Freeze the layers of the base model
for layer in base_model.layers:
    layer.trainable = False

# Add custom layers on top of the base model
x = Flatten()(base_model.output)
x = Dense(512, activation='relu')(x)
x = Dropout(0.5)(x)
x = Dense(256, activation='relu')(x)
x = Dropout(0.3)(x)
x = Dense(128, activation='relu')(x)
predictions = Dense(len(train_generator.class_indices), activation='softmax')(x)
```

Figure 51 Chargement du modèle VGG16 et ajout des couches personnalisées

Source : Données de l'auteur

Le modèle est compilé en utilisant l'optimiseur « Adam » qui ajuste les poids des couches en fonction des erreurs de classification calculées (Adam, n.d.). Deux callbacks, « EarlyStopping » et « ReduceLROnPlateau » sont utilisés respectivement pour arrêter l'entraînement si la perte de validation ne s'améliore pas durant un nombre d'époques défini, et pour réduire le taux d'apprentissage lorsque la perte de validation stagne (tf.keras.callbacks.EarlyStopping, 2024) (tf.keras.callbacks.ReduceLROnPlateau, 2024).

```
# Compile the model
model.compile(optimizer=Adam(learning_rate=0.0001), loss='categorical_crossentropy', metrics=['accuracy'])

# Define callbacks for early stopping and learning rate reduction
early_stopping = EarlyStopping(monitor='val_loss', patience=20, restore_best_weights=True)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=10, min_lr=0.00001, verbose=1)
```

Figure 52 Compilation du modèle et définition des callbacks

Source : Données de l'auteur

Enfin, le modèle est entraîné sur un maximum de 100 époques et il est ensuite sauvegardé. L'entraînement du modèle final a été stoppé après 62 époques par le callback « EarlyStopping ».

```python
# Train the model
history = model.fit(
    train_generator,
    epochs=100,
    validation_data=val_generator,
    verbose=2,
    callbacks=[early_stopping, reduce_lr]
)

# Save the trained model
model.save(model_save_path)
```

Figure 53 Entraînement et sauvegarde du modèle de classification de style

Source : Données de l'auteur

Utilisation du modèle

Dans le pipeline, l'œuvre à analyser doit subir quelques modifications pour correspondre au format attendu par le modèle entraîné.

```python
keras_image = load_img(image_path, target_size=(224, 224))
keras_image = img_to_array(keras_image)
keras_image = np.expand_dims(keras_image, axis=0)
keras_image /= 255.0
```

Figure 54 Prétraitement de l'image pour le modèle de classification de style

Source : Données de l'auteur

L'image est redimensionnée et convertie en un tableau NumPy. Les modèles de deep learning opèrent avec des valeurs numériques, la fonction « img_to_array » permet donc de convertir l'image dans un format utilisable par le modèle entraîné, où chaque élément du tableau correspond à la valeur d'un pixel. La fonction « np.expand_dims » est utilisée pour ajouter une dimension au tableau NumPy (numpy.expand_dims, n.d.). Cette dimension indique au modèle la taille du lot (le nombre d'images dans notre cas).

L'œuvre est ensuite passée au modèle qui prédit les probabilités pour les 8 styles, trouve l'index (le style) avec la probabilité la plus haute et le convertit en chaîne de caractères.

```python
style_prediction = art_style_model.predict(keras_image)
style_label_index = np.argmax(style_prediction)
style_label = style_map_reverse[style_label_index]
```

Figure 55 Prédiction du style et conversion en chaîne de caractères

Source : Données de l'auteur

4.6. Synthèse sonore

Il a été décidé au chapitre 3.3 d'utiliser le logiciel SuperCollider pour l'étape de synthèse sonore. SuperCollider utilise son propre langage de programmation orienté objet, « SuperCollider Language » ou « sclang », qui est utilisé pour créer des routines, des synthétiseurs et manipuler différentes données sonores.

Le script SuperCollider a été construit autour d'éléments basiques du langage musical : la mélodie, l'harmonie et le rythme (Vannet, 2007). Ainsi, dans chaque composition, l'harmonie est amenée par l'utilisation d'accords plus ou moins complexes, et une ligne monophonique amène de la mélodie et du rythme (en fonction des intervalles entre les notes). Ce choix de structure – une mélodie simple accompagnant une suite d'accord – a été fait pour plusieurs raisons. Tout d'abord, SuperCollider étant un langage très « verbeux », la création d'un script passablement simple nécessite une quantité de code importante. De plus, l'implémentation d'un prototype minimaliste a permis d'évaluer rapidement les différences réelles faites par les variations de caractéristiques extraites des images.

L'un des aspects clés pour amener de la diversité dans les compositions, a priori passablement minimalistes, a été de définir une série d'accords de complexités différentes. Les travaux du compositeur autrichien Arnold Schönberg ont été étudiés pour catégoriser ces accords. Selon son livre « Theory of Harmony », divers intervalles produisent différents niveaux de consonance ou de dissonance (Schönberg, 1948). Ceux-ci peuvent être définis comme suit :

1. Intervalles à consonance parfaite : octave et quinte parfaite
2. Intervalles à consonance imparfaite : tierce majeure, tierce mineure, sixième majeure et sixième mineure
3. Intervalles dissonants : septième majeure, septième mineure et neuvième

Ces listes ne sont pas exhaustives mais contiennent les intervalles qui sont utilisés dans le cadre de ce projet. Nous reviendrons sur ces aspects dans les chapitres suivants qui détaillent la structure du script SuperCollider.

4.6.1. Extraction du contenu du fichier JSON & définition des variables

La première partie du script est construite autour de l'extraction des caractéristiques de l'œuvre analysée. Les données du fichier JSON sont extraites et associées à une ou plusieurs variables utilisées à différentes étapes de la synthèse sonore. Ces relations ont été discutées au chapitre 4.3, mais nous allons ici préciser certaines d'entre elles.

```
// KEY & SCALE
keyMapping = (
    black: 60, brown: 61, dark_blue: 62, purple: 63,
    red: 64, orange: 65, green: 66, grey: 67,
    light_blue: 68, pink: 69, yellow: 70, white: 71
);

key = keyMapping[dominantColor.asSymbol];
scale = if(emotion == "positive") { Scale.major } { Scale.minor }; // Scale definition
scaleNotesMidi = scale.degrees + key; // Extract MIDI values from the scale
scaleNotesFreq = scale.degrees.collect { |degree| (degree + key).midicps }; // Extract Hz values for the notes
```

Figure 56 Mapping Couleur-Note MIDI

Source : Données de l'auteur

Comme le montre la figure ci-dessus, un dictionnaire est créé pour associer le nom des couleurs à des numéros de note MIDI. Ces numéros de notes MIDI sont ensuite utilisés, avec la valeur de la variable « emotion », pour déterminer les notes MIDI de la gamme qui sera utilisée pour créer la mélodie. Cette gamme est également exploitée pour créer les divers accords qui pourront être utilisés, comme le montre l'image ci-dessous.

```
// Consonant chords
tonicPerfectFifth = [scaleNotesMidi[0] - 36, scaleNotesMidi[4] - 12, scaleNotesMidi[0], scaleNotesMidi[4], scaleNotesMidi[0] + 12];
secondPerfectFifth = [scaleNotesMidi[1] - 36, scaleNotesMidi[5] - 12, scaleNotesMidi[1], scaleNotesMidi[5], scaleNotesMidi[1] + 12];
thirdPerfectFifth = [scaleNotesMidi[2] - 36, scaleNotesMidi[6] - 12, scaleNotesMidi[2], scaleNotesMidi[6], scaleNotesMidi[2] + 12];
fourthPerfectFifth = [scaleNotesMidi[3] - 36, scaleNotesMidi[0] - 12, scaleNotesMidi[3], scaleNotesMidi[0], scaleNotesMidi[3] + 12];
fifthPerfectFifth = [scaleNotesMidi[4] - 36, scaleNotesMidi[1] - 12, scaleNotesMidi[4], scaleNotesMidi[1], scaleNotesMidi[4] + 12];
sixthPerfectFifth = [scaleNotesMidi[5] - 36, scaleNotesMidi[2] - 12, scaleNotesMidi[5], scaleNotesMidi[2], scaleNotesMidi[5] + 12];
consonants = [tonicPerfectFifth, secondPerfectFifth, thirdPerfectFifth, fourthPerfectFifth, fifthPerfectFifth, sixthPerfectFifth]; // Array of consonant chords

// Slightly dissonant chords
tonicTriad = [scaleNotesMidi[0] - 24, scaleNotesMidi[0], scaleNotesMidi[2], scaleNotesMidi[4], scaleNotesMidi[4] - 12];
secondTriad = [scaleNotesMidi[1] - 24, scaleNotesMidi[1], scaleNotesMidi[3], scaleNotesMidi[5], scaleNotesMidi[5] - 12];
fourthTriad = [scaleNotesMidi[2] - 24, scaleNotesMidi[2], scaleNotesMidi[4], scaleNotesMidi[6], scaleNotesMidi[6] - 12];
thirdTriad = [scaleNotesMidi[3] - 24, scaleNotesMidi[3], scaleNotesMidi[5], scaleNotesMidi[0], scaleNotesMidi[0] - 12];
fifthTriad = [scaleNotesMidi[4] - 24, scaleNotesMidi[4], scaleNotesMidi[6], scaleNotesMidi[1], scaleNotesMidi[1] - 12];
sixthTriad = [scaleNotesMidi[5] - 24, scaleNotesMidi[5], scaleNotesMidi[0], scaleNotesMidi[2], scaleNotesMidi[2] - 12];
sDissonants = [tonicTriad, secondTriad, thirdTriad, fourthTriad, fifthTriad, sixthTriad]; // Array of slightly dissonant chords

// Dissonant chords
diminishedTriad = [scaleNotesMidi[1], scaleNotesMidi[3], scaleNotesMidi[3] - 12, scaleNotesMidi[6], scaleNotesMidi[6] - 24];
tonicSeventh = [scaleNotesMidi[0] - 24, scaleNotesMidi[0], scaleNotesMidi[2], scaleNotesMidi[4], scaleNotesMidi[6]];
secondSeventh = [scaleNotesMidi[1] - 24, scaleNotesMidi[1], scaleNotesMidi[3], scaleNotesMidi[5], scaleNotesMidi[0]];
thirdSeventh = [scaleNotesMidi[2] - 24, scaleNotesMidi[2], scaleNotesMidi[4], scaleNotesMidi[6], scaleNotesMidi[1]];
fourthSeventh = [scaleNotesMidi[3] - 24, scaleNotesMidi[3], scaleNotesMidi[5], scaleNotesMidi[0], scaleNotesMidi[2]];
fifthSeventh = [scaleNotesMidi[4] - 24, scaleNotesMidi[4], scaleNotesMidi[6], scaleNotesMidi[1], scaleNotesMidi[3]];
sixthSeventh = [scaleNotesMidi[5] - 24, scaleNotesMidi[5], scaleNotesMidi[0], scaleNotesMidi[2], scaleNotesMidi[4]];
seventhSeventh = [scaleNotesMidi[6] - 24, scaleNotesMidi[6], scaleNotesMidi[1], scaleNotesMidi[3], scaleNotesMidi[5]];
dissonants = [diminishedTriad, tonicSeventh, secondSeventh, thirdSeventh, fourthSeventh, fifthSeventh, sixthSeventh, seventhSeventh]; // Array of dissonant chords
```

Figure 57 Définition des accords et de leur complexité

Source : Données de l'auteur

Dans un second temps, diverses variables sont initialisées avec des valeurs en lien avec les caractéristiques de texture de l'œuvre analysée. Comme discuté au chapitre 4.3, elles permettent de moduler certains aspects de la composition comme la quantité d'effets appliquée ou la complexité des accords joués.

```
// 'Roughness' defines the complexity of the chords, i.e. consonant, slightly dissonant, dissonant
chordComplexity = if (roughness <= 0.3) {
    1
} {
    if (roughness > 0.3 && roughness <= 0.6) {
        2
    } {
        3
    }
};

// Create array of array based on 'nbChords' and 'chordComplexity'
chords = if (chordComplexity == 1) {
    Array.fill(nbChords, { consonants.choose })
} {
    if (chordComplexity == 2) {
        Array.fill(nbChords, { sDissonants.choose }) // Choose from sDissonants
    } {
        Array.fill(nbChords, { dissonants.choose }) // Choose from dissonants
    }
};
```

Figure 58 Choix du nombre d'accords et de leur complexité

Source : Données de l'auteur

4.6.2. Définition des synthétiseurs

La synthèse soustractive a été utilisée pour tous les synthétiseurs. Pour rappel, cette technique consiste à adoucir le signal généré par une onde en utilisant des filtres fréquentiels (Synthèse sonore soustractive, 2024). Cette technique a pour avantage d'être facile à implémenter et peu gourmande en ressources computationnelles, mais ne permet qu'un champ limité de sonorités.

Trois synthétiseurs sont utilisés pour générer les accords et les mélodies, chacun utilisant un type d'onde différent : une onde carrée, une onde sinusoïdale et une onde en dents de scie. Ces trois synthétiseurs ont une structure identique et contiennent une série d'éléments qui peuvent être définis ou modulés en fonction du style de l'œuvre d'art. L'enveloppe de volume, la plage de fréquence, la modulation de cette plage, le taux de « désaccordage » ou la résonance du filtre appliqué peuvent ainsi changer permettant plus de variation dans les compositions.

```
SynthDef(\saw, {
    arg atk = 2, sus = 0, rel = 3, c1 = 1, c2 = (-1),
    freq = 500, detune = 0.2, pan = 0, cfhzmin = 0.1, cfhzmax = 0.3,
    cfmin = 100, cfmax = 2000, rqmin = 0.1, rqmax = 0.2,
    lsf = 200, ldb = 0, amp = 1, out = 0, reverb, delay;
    var sig, env;
    env = EnvGen.kr(Env([0, 1, 1, 0], [atk, sus, rel], [c1, 0, c2]), doneAction: 2);
    sig = Saw.ar(freq * { LFNoise1.kr(0.5, detune).midiratio }!2);
    sig = BPF.ar(
        sig,
        { LFNoise1.kr(LFNoise1.kr(4).exprange(cfhzmin, cfhzmax)).exprange(cfmin, cfmax) }!2,
        { LFNoise1.kr(0.1).exprange(rqmin, rqmax) }!2
    );
    sig = BLowShelf.ar(sig, lsf, 0.5, ldb);
    sig = Balance2.ar(sig[0], sig[1], pan);
    sig = sig * env * amp;
    Out.ar(reverb, sig); // Output to reverb bus
    Out.ar(delay, sig);  // Output to delay bus
}).add;
```

Figure 59 Synthétiseur à onde en dents de scie

Source : Données de l'auteur

Deux effets ont également été implémentés : un module de réverbération et un écho. Ceux-ci sont appliqués en parallèle, c'est-à-dire qu'un signal « traité », composé seulement de l'effet, est mixé au signal « sec » (sans effet). La quantité d'effet peut ainsi être variée en modulant le volume du synthétiseur de réverbération.

```
SynthDef(\reverb, {
    arg in, predelay = 0.1, revtime = 1.8, lpf = 4500, mix = 0.5, amp = 1, out = 0;
    var dry, wet, temp, sig;
    dry = In.ar(in, 2);
    temp = In.ar(in, 2);
    wet = 0;
    temp = DelayN.ar(temp, 0.2, predelay);
    16.do {
        temp = AllpassN.ar(temp, 0.05, { Rand(0.001, 0.05) }!2, revtime);
        temp = LPF.ar(temp, lpf);
        wet = wet + temp;
    };
    sig = XFade2.ar(dry, wet, mix * 2 - 1, amp);
    Out.ar(out, sig);
}).add;
```

Figure 60 Synthétiseur de réverbération

Source : Données de l'auteur

Enfin, un dernier module permet d'ajouter du « grain » à la composition. La synthèse granulaire est utilisée ici : un son existant est découpé en échantillons qui sont ensuite combinés pour créer un signal sonore.

```
SynthDef(\granular, {
    arg bufnum, amp = 0.5, out = 0, reverb, dur = 1, rate = 5;
    var sig, env;
    sig = GrainBuf.ar(
        2,
        Impulse.ar(50),
        dur,
        bufnum,
        rate,
        (
            Phasor.ar(0, 1 * BufRateScale.ir(bufnum), 0, BufSamples.ir(bufnum) - 1) + LFNoise1.ar(100).bipolar(0.01 * SampleRate.ir)
        ) / BufSamples.ir(bufnum),
        5,
        0,
        -1,
        512
    ) * amp;
    Out.ar(reverb, sig);
}).add;
```

Figure 61 Synthétiseur granulaire

Source : Données de l'auteur

4.6.3. Styles artistiques

Chaque style de peinture ayant ses spécificités, lorsque les synthétiseurs sont créés, diverses variables sont initialisées avec des valeurs qui visent à refléter certaines caractéristiques du style de l'œuvre. De plus, différents styles vont faire appel à des synthétiseurs utilisant différents types d'ondes. La figure 62 présentée ci-après illustre l'initialisation de certaines variables pour le cubisme.

```
cubismDetune = 0.1;
cubismFx = 0.1;
cubismAtt = 0.5;
cubismRel = 2;
cubismFtrLo = 50;
cubismFtrHi = 1000;
cubismFtrModLo = 0.1;
cubismFtrModHi = 0.2;
cubismFtrResLow = 0.3;
cubismFtrResHi = 0.5;
cubismDurLo = 4;
cubismDurHi = 7;
```

Figure 62 Initialisation des variables pour le cubisme

Source : Données de l'auteur

Les adjectifs définis au chapitre 4.3 ont été utilisés pour définir ces valeurs, et des ajustements ont été faits suite à l'analyse d'images de test.

4.6.4. Enregistrement du fichier audio

La dernière étape du processus de synthèse sonore consiste à enregistrer et exporter un fichier audio au format WAV. Il a été décidé de générer une composition de 60 secondes pour chaque œuvre. Les synthétiseurs sont stoppés après 50 secondes, ce qui laisse le temps aux divers sons de « mourir », et l'enregistrement est arrêté après 60 secondes. Suite à l'arrêt de l'enregistrement, le serveur est complètement libéré : tous les modules sont détruits dans la mémoire vive).

```
// Stop recording and free server after 30 seconds
Routine({
    60.wait;
    s.stopRecording;
    {
        file = File(recordPath ++ ".done", "w");
        file.openWrite;
        file.write("done");
        file.close;
        "Done file created.".postln;
    }.defer;
    s.freeAll;
    "Recording stopped and server freed.".postln;
    s.quit;
    "Quit server".postln;
}).play(SystemClock);
```

Figure 63 Processus d'enregistrement du fichier audio

Source : Données de l'auteur

4.7. Automatisation du pipeline

Tout le processus de sonification, de l'analyse de l'œuvre à l'exportation d'un fichier audio, a été automatisé sur le serveur. Pour ce faire, deux scripts ont été créés : l'un qui observe un dossier « images » dans lequel l'œuvre à analyser doit être déposée, et l'autre qui observe un dossier « results » dans lequel est sauvegardé le fichier JSON généré lors de l'analyse de l'œuvre.

Le script watch_images.sh fait appel au script Python « image_analysis.py » lorsqu'une image est déposée dans le dossier observé. La structure complète et le contenu de ce script Pyhton ont été discutés au chapitre 4.4.

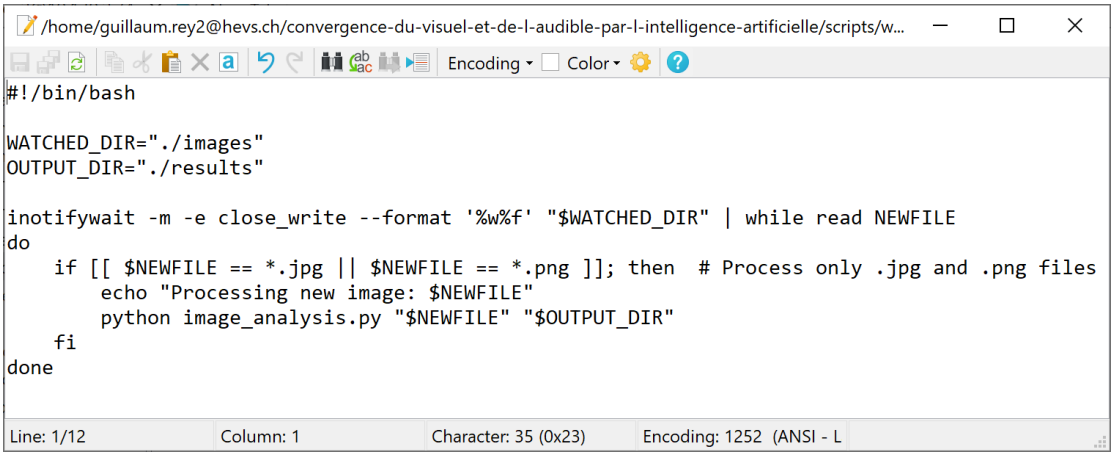


Figure 64 watch_images.sh

Source : Données de l'auteur

Le script watch_results.sh fait appel quant à lui au script SuperCollider, sound_synthesis.scd, qui initialise le serveur SuperCollider, analyse le fichier JSON et génère et enregistre une composition sonore. Le chapitre précédent explique en détail la structure de ce script. Le fichier WAV est enregistré dans le dossier « recordings ».

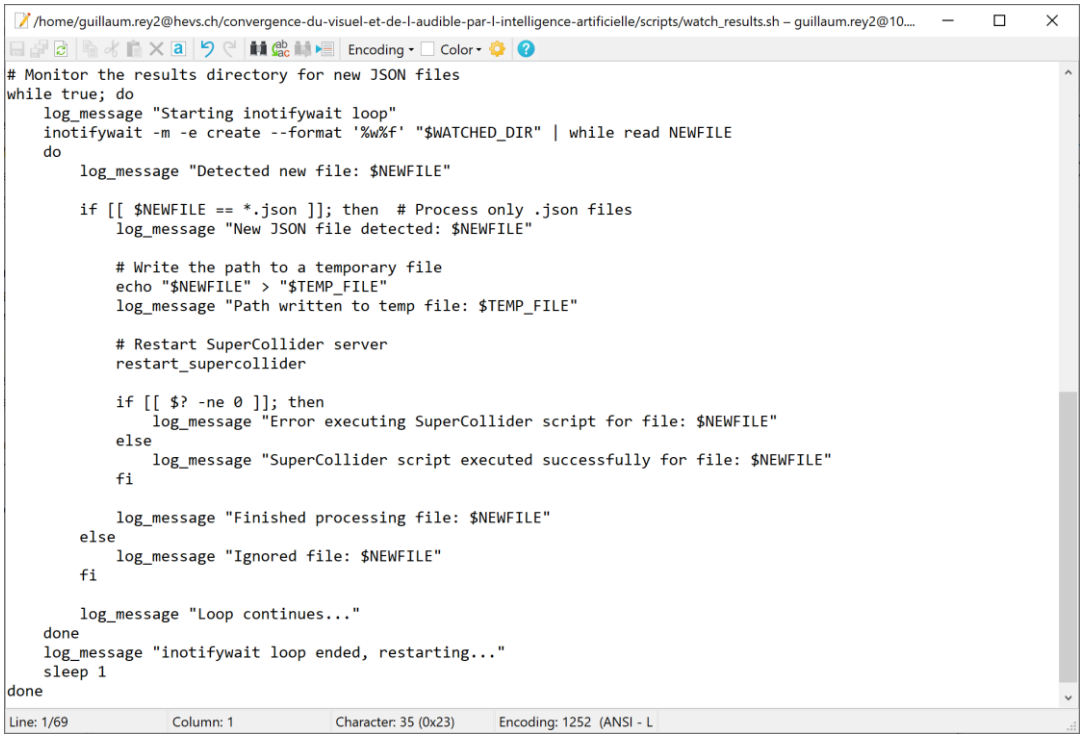


Figure 65 Extrait de watch_results.sh

Source : Données de l'auteur

Dans ce prototype, le serveur n'a pas été configuré pour gérer plusieurs requêtes simultanément.

4.8. Développement du backend

Le Framework Flask a été utilisé pour sa facilité d'utilisation et sa mise en place rapide. Diverses fonctions et classes Flask ont été importées pour la gestion des routes, des requêtes HTTP, des réponses JSON et de l'envoi des fichiers. Suite à l'initialisation de l'app, les dossiers utilisés pour l'enregistrement des œuvres à analyser et des fichiers audio ont été définis, et les extensions autorisées ont été précisées.

```
# Initialize the Flask application
app = Flask(__name__)

# Configure upload and recordings folders and allowed extensions
app.config['UPLOAD_FOLDER'] = '/home/guillaum.rey2@hevs.ch/convergence-du-
app.config['RECORDINGS_FOLDER'] = '/home/guillaum.rey2@hevs.ch/convergence
app.config['ALLOWED_EXTENSIONS'] = {'png', 'jpg', 'jpeg'}
```

Figure 66 Initialisation de l'app Flask & configuration des dossiers

Source : Données de l'auteur

Les routes vers les différentes pages sont ensuite définies. Celles-ci définissent les routes d'affichage de la page d'accueil, de la page de chargement de l'image, de la page d'attente lorsque l'analyse est en cours et de la page d'affichage de l'œuvre avec le lecteur audio lorsque la sonification est terminée.

Il y a également une route « check_status » qui fait appel à une requête AJAX qui vérifie si le fichier audio est prêt. Si ce n'est pas le cas, une nouvelle requête est envoyée après 5 secondes.

```
// Function to check the status of a file
function checkStatus(filename) {
    // Make an AJAX request to check the status of the file
    $.getJSON('/check_status/' + filename, function(data) {
        if (data.ready) {
            // If the file is ready, redirect to the recordings page
            window.location.href = '/recordings/' + filename;
        } else {
            // If the file is not ready, check again after 5 seconds
            setTimeout(() => checkStatus(filename), 5000);
        }
    }).fail(function() {
        // Log an error message if the AJAX request fails
        console.log("Error checking status");
        // Check again after 5 seconds
        setTimeout(() => checkStatus(filename), 5000);
    });
}
```

Figure 67 Fonction de vérification du statut du fichier audio

Source : Données de l'auteur

La complétion de la génération de la composition sonore est indiquée par la création d'un fichier « .wav.done » portant le nom de l'œuvre analysée. Le système est donc informé que la redirection vers la page d'écoute de la composition peut être faite.

```python
# Route to check the status of a file
@app.route('/check_status/<filename>')
def check_status(filename):
    done_flag_path = os.path.join(app.config['RECORDINGS_FOLDER'], filename.rsplit('.', 1)[0] + '.wav.done')
    if os.path.isfile(done_flag_path):
        return jsonify({'ready': True})
    return jsonify({'ready': False})
```

Figure 68 Définition de la fonction "check_status"

Source : Données de l'auteur

La communication entre le client et le serveur est facilitée par les méthodes HTTP GET et POST. Elles sont respectivement utilisées pour afficher le formulaire de téléchargement « loadImage.html » et pour traiter le fichier téléchargé.

```python
# Route to load an image
@app.route('/loadImage', methods=['GET', 'POST'])
def loadImage():
    if request.method == 'POST':
        file = request.files.get('file')
        if file and allowed_file(file.filename):
            filename = secure_filename(file.filename)
            filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)
            file.save(filepath)
            logging.debug("File saved successfully")
            return redirect(url_for('loading', filename=filename))
    return render_template('loadImage.html')
```

Figure 69 Route de chargement d'une image

Source : Données de l'auteur

4.9. Développement du frontend

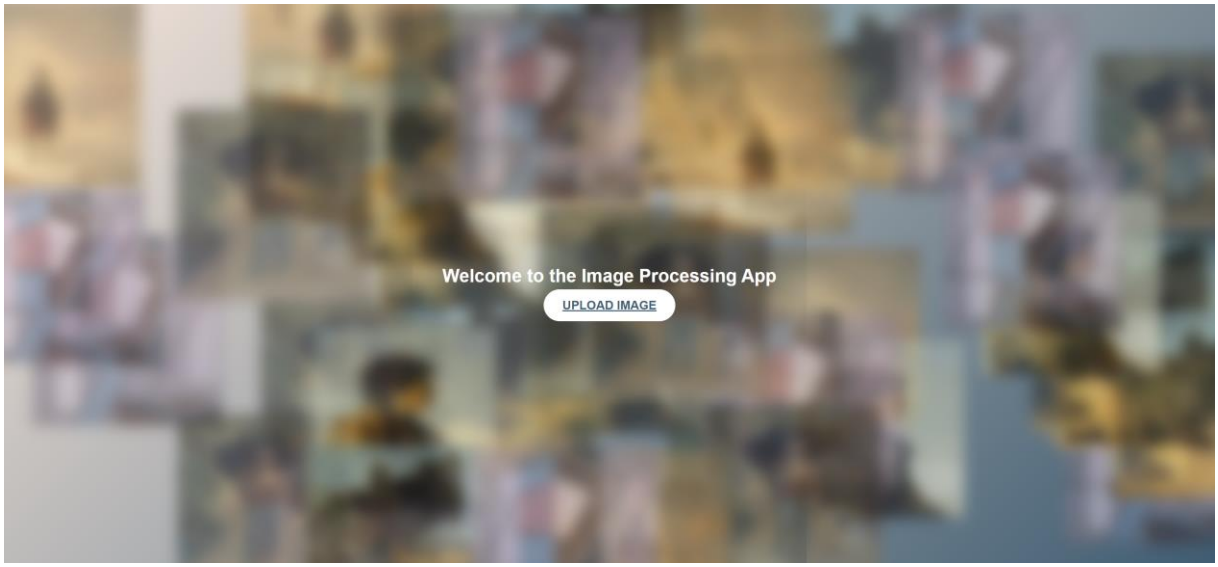


Figure 70 Web app - Page d'accueil

Source : Données de l'auteur

Les styles CSS pour le body et html définissent le style de base de l'application. « Flexbox » est utilisé pour centrer le contenu à la fois horizontalement et verticalement, les propriétés « overflow » permettent le défilement vertical mais pas horizontal, et un dégradé linéaire est appliqué à l'arrière-plan.

```css
body, html {
    height: 100%;
    margin: 0;
    padding: 0;
    overflow-y: auto; /* Allow vertical scrolling */
    overflow-x: hidden; /* Prevent horizontal scrolling */
    font-family: 'Poppins', sans-serif; /* Set font family */
    display: flex; /* Use flexbox for centering */
    justify-content: center; /* Center horizontally */
    align-items: center; /* Center vertically */
    background: linear-gradient(135deg, #F7E7DC, #405D72); /* Background gradient */
    color: #fff; /* Text color */
}
```

Figure 71 Style CSS pour le body et le html

Source : Données de l'auteur

Tous les boutons partagent un style commun : couleur de fond blanche, texte en couleur contrastante, coins arrondis et effet de survol.

```css
/* Common styling for links, submit buttons, and file input buttons */
a, input[type="submit"], input[type="file"]::file-selector-button {
    background-color: #ffffff; /* Background color */
    color: #405D72; /* Text color */
    padding: 12px 25px; /* Padding around text */
    border: none; /* No border */
    border-radius: 25px; /* Rounded corners */
    text-transform: uppercase; /* Uppercase text */
    transition: background-color 0.3s ease; /* Smooth background color transition */
    font-weight: bold; /* Bold text */
    margin-top: 20px; /* Top margin */
    cursor: pointer; /* Pointer cursor on hover */
}
```

Figure 72 Style CSS des boutons

Source : Données de l'auteur

En arrière-plan, des images floutées flottent à une vitesse et dans une direction aléatoire. Cet effet a été généré à l'aide du script JavaScript de la figure 73.

```javascript
// Loop to create and append 50 floating images
for (let i = 0; i < 50; i++) {
    const img = document.createElement('img'); // Create a new img element
    img.src = `/static/images/${imageNames[i % imageNames.length]}`; // Set the source of the image
    img.className = 'floating-img'; // Assign a class name to the image
    img.style.position = 'absolute'; // Set the image position to absolute
    img.style.top = `${Math.random() * 100}%`; // Randomize the top position
    img.style.left = `${Math.random() * 100}%`; // Randomize the left position
    img.style.transform = `translate(-50%, -50%) scale(${Math.random() * 0.5 + 3})`; // Apply random scale transformation
    img.style.opacity = 0; // Set initial opacity to 0

    container.appendChild(img); // Append the image to the container

    // Fade in the image with a slight delay based on its index
    setTimeout(() => img.style.opacity = 0.6, 10 + 100 * i);

    // Randomize the speed of movement in both X and Y directions
    const speedX = Math.random() * 0.1 - 0.05;
    const speedY = Math.random() * 0.1 - 0.05;

    // Move the image in random directions at regular intervals
    setInterval(() => {
        const topVal = parseFloat(img.style.top); // Get the current top position
        const leftVal = parseFloat(img.style.left); // Get the current left position
        img.style.top = `${(topVal + speedY + 100) % 100}%`; // Update the top position
        img.style.left = `${(leftVal + speedX + 100) % 100}%`; // Update the left position
    }, 100); // Interval of 100 milliseconds
}
```

Figure 73 Génération et affichage des images pour l'arrière-fond de la web app

Source : Données de l'auteur

Finalement, une transition fait apparaitre les images d'arrière-plan lorsque la page est chargée, en modifiant la valeur d'opacité.

```css
/* Styling for floating images container */
.floating-images {
    position: absolute; /* Absolute positioning */
    top: 0; /* Top of the viewport */
    left: 0; /* Left of the viewport */
    width: 100%; /* Full width */
    height: 100%; /* Full height */
    pointer-events: none; /* Allow clicks to pass through */
    overflow: hidden; /* Hide overflow */
}

/* Styling for individual floating images */
.floating-img {
    position: absolute; /* Absolute positioning */
    width: 10vmin; /* Width relative to viewport */
    height: 10vmin; /* Height relative to viewport */
    opacity: 0.6; /* Semi-transparent */
    filter: blur(3px); /* Blur effect */
    transition: opacity 2s ease-in; /* Smooth opacity transition */
}
```

Figure 74 Style CSS des images d'arrière-plan

Source : Données de l'utilisateur

5. Résultats

Nous allons, dans ce chapitre, étudier les résultats des divers modèles et outils utilisés pour l'analyse des œuvres. Pour chacun des modèles, la justesse, ou « accuracy » sera discutée.

La justesse est calculée en divisant le nombre de prédictions correctes par le nombre total de prédictions (Machine Learning : Cours de base, n.d.). Pour la classification binaire (par exemple pour notre modèle d'analyse des émotions), la justesse peut être mesurée comme la somme des vrais positifs et des vrais négatifs divisée par la somme des vrais positifs (TP), des vrais négatifs (TN), des faux positifs (FP) et des faux négatifs (FN), comme l'illustre la figure ci-dessous (Machine Learning : Cours de base, n.d.).

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Figure 75 Calcul de la justesse d'un modèle

Source : https://developers.google.com/machine-learning/crash-course/classification/accuracy?hl=fr

Une matrice de confusion permet de représenter la justesse d'un modèle en indiquant le nombre de vrais positifs, de vrais négatifs, de faux positifs et de faux négatifs.

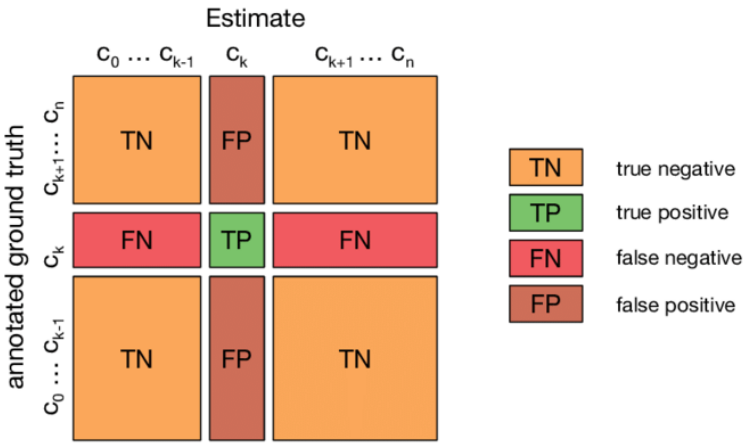


Figure 76 Matrice de confusion pour une classification multi-classe

Source : https://www.researchgate.net/figure/Confusion-matrix-for-multi-class-classification-The-confusion-matrix-of-a_fig7_314116591

Pour certains modèles, nous observerons également le score F1 qui est une moyenne pondérée de la précision et du rappel. La précision est le ratio de vrais positifs par rapport au nombre total de prédictions positives (Fonctionnement de l'outil Calculer la précision pour la détection d'objets, n.d.). Le rappel correspond au ratio de vrais positifs par rapport au nombre total d'objets pertinents (Fonctionnement de l'outil Calculer la précision pour la détection d'objets, n.d.). Le score F1 permet donc de fournir une image plus juste de la performance du modèle lorsqu'il y a des déséquilibres entre certaines classes.

5.1. Modèle de classification des émotions

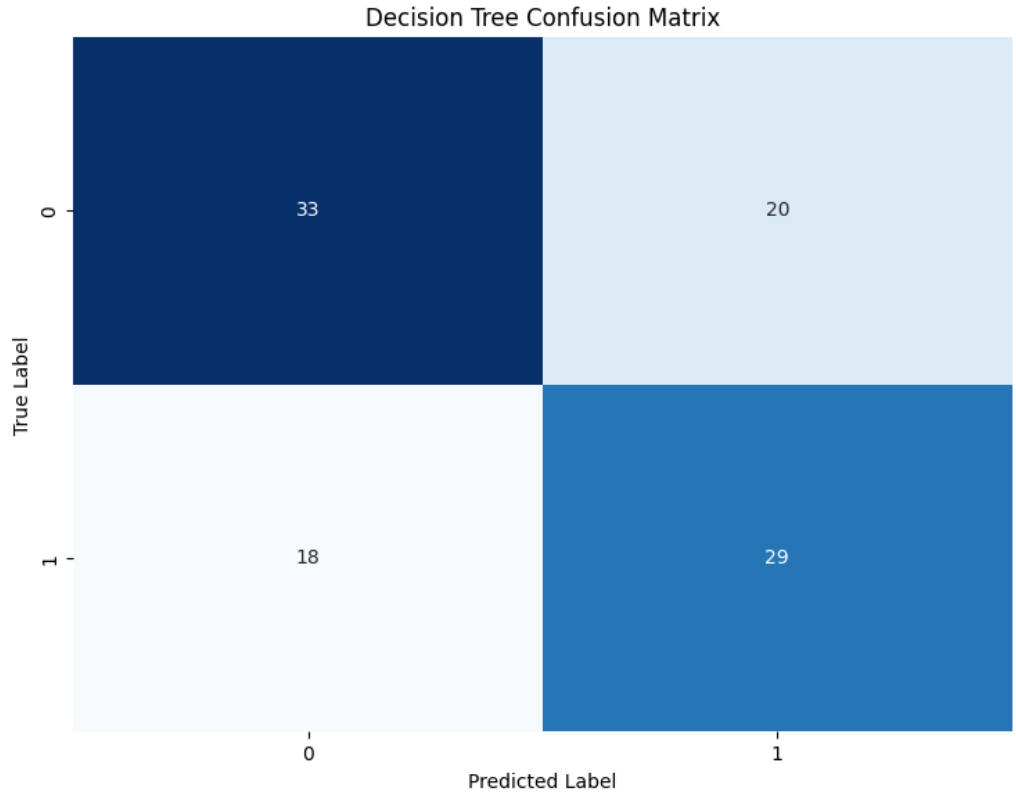


Figure 77 Matrice de confusion - Modèle de classification des émotions

Source : Données de l'auteur

La justesse du modèle est de 62 %, et les scores F1 sont de 0,51 pour les deux classes.

5.2. Modèle de nommage des couleurs

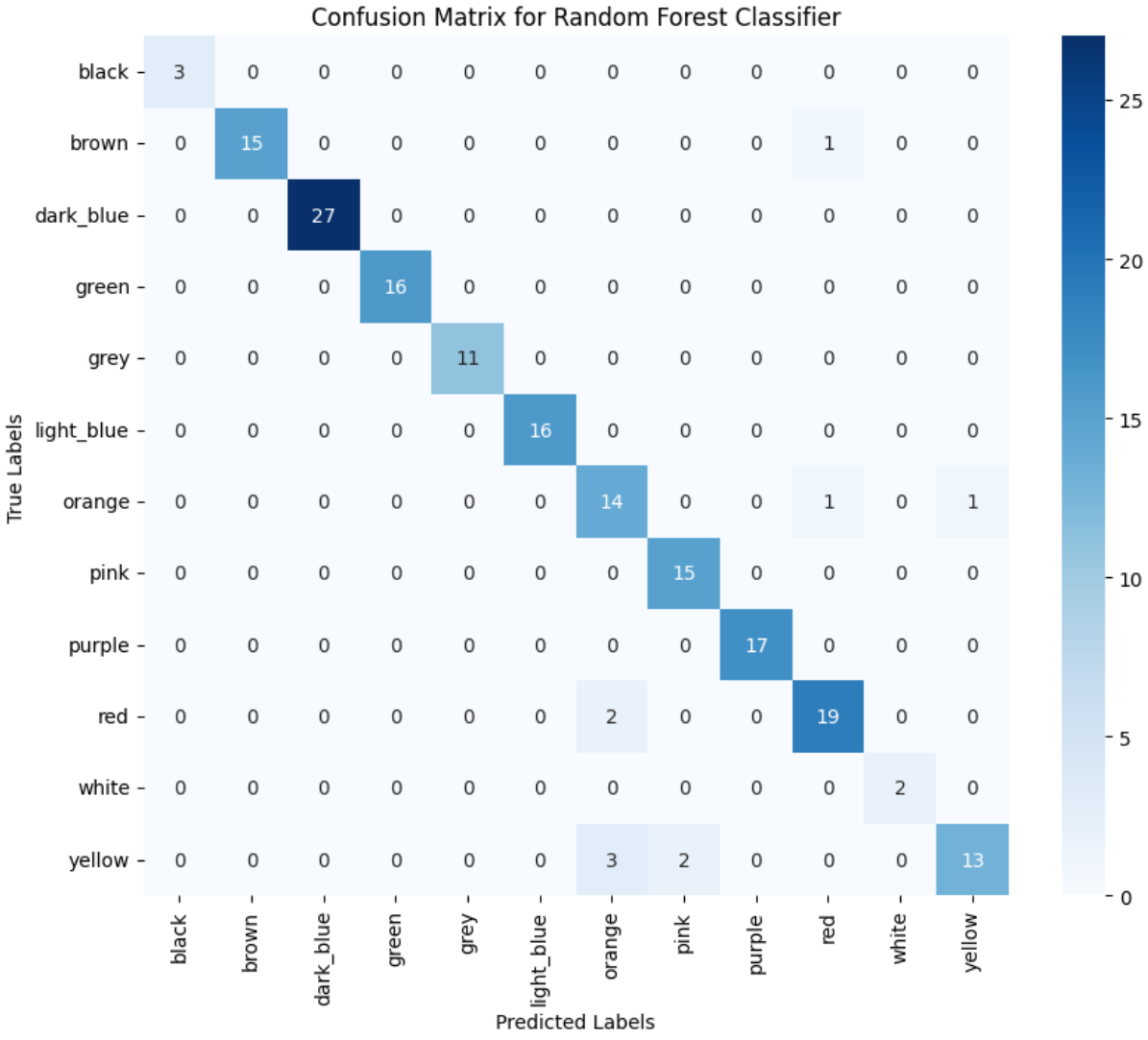


Figure 78 Matrice de confusion - Modèle de nommage des couleurs

Source : Données de l'auteur

La justesse du modèle est de 94 %, et les scores F1 sont les suivants :

- Noir, bleu foncé, vert, gris, bleu clair, violet et blanc : 1,00
- Brun : 0,97
- Orange : 0,80
- Rose : 0,94
- Rouge : 0,90

- Jaune : 0,81

5.3. Modèle de reconnaissance du style

L'entraînement du modèle a été stoppé après 62 époques. La figure 79 montre la justesse du modèle sur les données d'entraînement et sur les données de validation, et la perte sur les données d'entraînement et de validation.

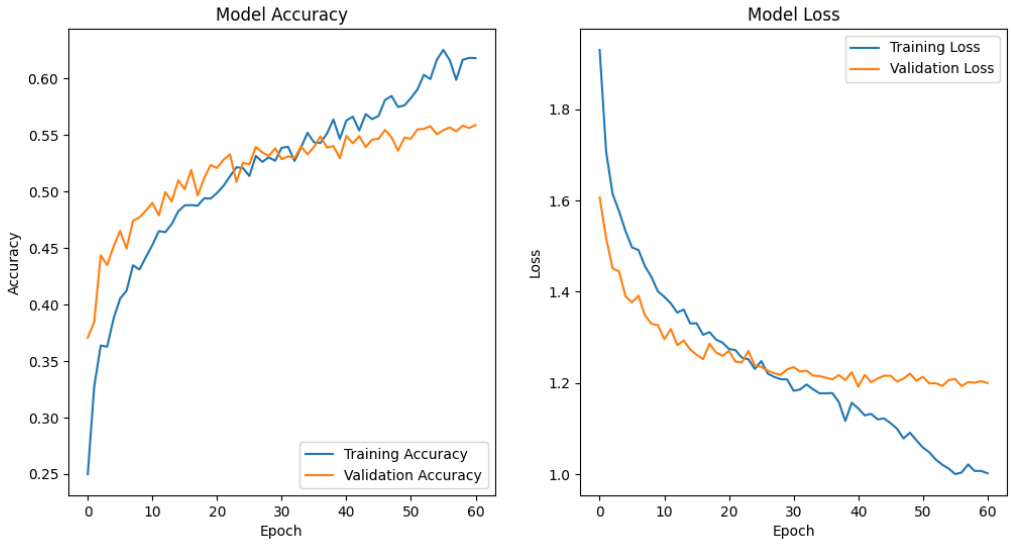


Figure 79 Justesse du modèle de détection du style

Source : Données de l'auteur

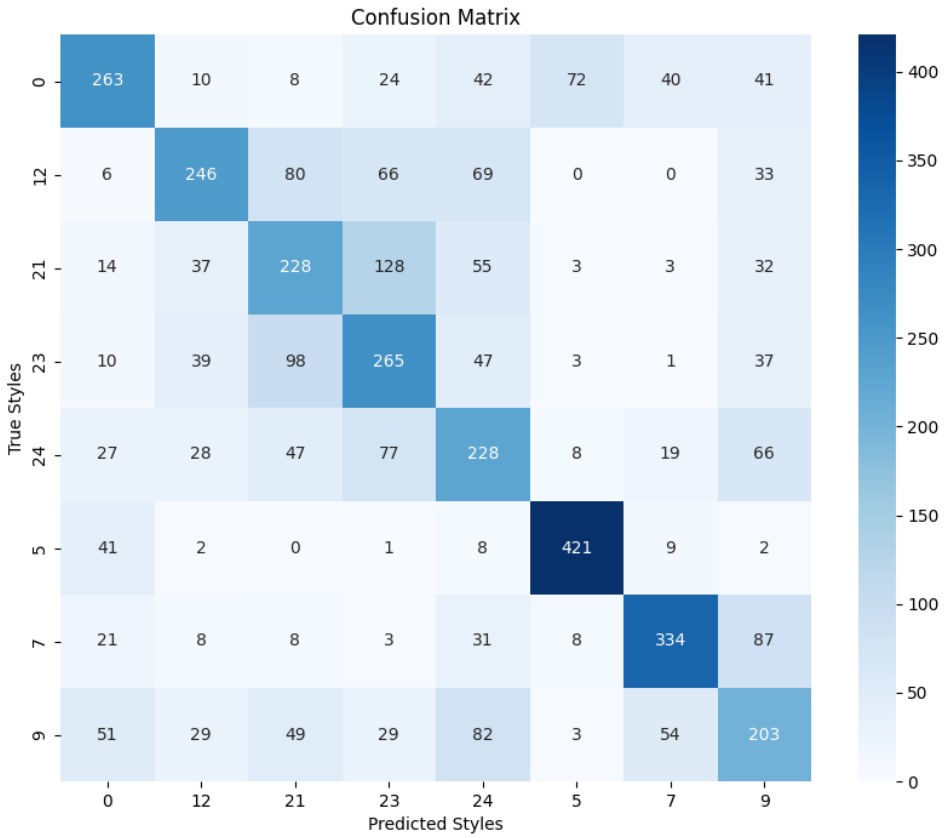


Figure 80 Matrice de confusion - Modèle de détection du style

Source : Données de l'auteur

Afin de clarifier les légendes de la matrice ci-dessus, 0 correspond à l'art abstrait, 12 à l'impressionnisme, 21 au réalisme, 23 au romantisme, 24 au symbolisme, 5 à la peinture en champs de couleurs, 7 au cubisme et 9 à l'expressionisme.

La justesse du modèle est de 55 %, et les scores F1 sont les suivants :

- Art abstrait : 0,56
- Impressionnisme : 0,55
- Réalisme : 0,45
- Romantisme : 0,48
- Symbolisme : 0,43

- Peinture en champs de couleurs : 0,84
- Cubisme : 0,70
- Expressionnisme : 0,41

6. Discussion des résultats

Le dossier « Résultats » joint à ce travail contient 16 dossiers correspondant aux résultats de l'analyse de 16 œuvres, soit 2 œuvres par style. Dans chacun de ces dossiers se trouvent une image de l'œuvre, un fichier JSON contenant les caractéristiques extraites lors de l'analyse, et un fichier audio résultant du processus de sonification.

L'annexe 10.4 contient les résultats de l'analyse des 16 œuvres (sans les fichiers audio), avec un détail de toutes les caractéristiques extraites.

6.1. Analyse de l'œuvre

Avant de discuter précisément des résultats des différentes étapes du processus de sonification, nous allons partager une appréciation globale de l'analyse de certaines œuvres.

Figure 81 Résultats de l'analyse de l'œuvre « Swiss landscape with flowering apple tree » de Gustave Courbet, labellisée "réalisme"

Source : Données de l'auteur

Figure 82 Résultats de l'analyse d'une œuvre de Aki Kuroda, labellisée "peinture en champs de couleur"

Source : Données de l'auteur

Figure 83 Résultats de l'analyse de l'œuvre « A lane in Headingley, Leeds » de John Atkinson Grimshaw, labellisée "romantisme"

Source : Données de l'auteur

Comme indiqué au chapitre 4.3.4, les œuvres utilisées pour tester l'application ont été sélectionnées dans le dataset « WikiArt », parmi celles qui n'ont pas été utilisées pour entraîner le modèle de prédiction. Ainsi, pour les 16 œuvres analysées, le style a été correctement prédit pour 11 images (selon les annotations du jeu de données).

Là où la prédiction ne correspond pas au label du dataset, le modèle fait généralement ressortir une probabilité plus ou moins élevée pour plusieurs styles. Pour la figure 83, par exemple, le modèle donne une probabilité de 41,51 pour le réalisme et de 30,37 pour le romantisme, qui est le style annoté. Il semble donc que dans la plupart des cas, les mauvaises prédictions sont dues à une incapacité à distinguer deux ou plusieurs styles qui partagent certaines caractéristiques.

Nous nous sommes rendu compte lors de cette analyse de résultats qu'il aurait été judicieux d'inclure les images utilisées pour tester l'application aux sondages réalisés pour annoter les émotions et évaluer les méthodes d'extraction de la couleur dominante. Cela aurait permis d'avoir des données de style, d'émotion et de couleur pour toutes ces œuvres et une discussion plus précise des résultats aurait pu être faite.

Pour contrer ce manquement et évaluer l'émotion détectée, les images utilisées pour tester le pipeline ont été comparées à diverses images annotées (au minimum 3) partageant des similarités de couleur, mais également de style et de structure. La figure 84 montre, au centre, l'œuvre utilisée pour tester le processus, et autour des images « similaires » dont l'émotion a été observée pour évaluer l'émotion la plus probable pour l'œuvre centrale.

Figure 84 Analyse de l'émotion d'œuvres similaires

Source : Données de l'auteur

Ainsi, selon cette appréciation, l'émotion dominante extraite dans les images de test est « cohérente » dans 62,5 % des cas (10 œuvres sur 16), ce qui est bien aligné avec les résultats du modèle vus au chapitre précédent.

Émotion

Le modèle de classification des émotions, avec une justesse de 62 %, manque ostensiblement de précision, et ce malgré l'augmentation de données utilisée lors de la phase d'entraînement.

Dans un modèle de cette complexité, l'ajustement de certains paramètres peut amener à des changements importants dans les résultats. Ainsi, une révision du modèle pourrait permettre d'améliorer la justesse de celui-ci. En outre, le modèle actuel ne prend pas en compte la valeur de certitude associée à chaque image, calculée lors de l'annotation des œuvres. La considération de cet aspect lors de l'entraînement du modèle permettrait certainement d'obtenir une meilleure classification. Une image ayant une valeur sigma très basse devrait avoir plus de poids qu'une œuvre dont l'incertitude est haute.

Texture

Pour obtenir une plus grande précision et une meilleure granularité dans les valeurs des différentes caractéristiques de texture, les plages de valeur pourraient être évaluées sur un jeu de données plus important et diversifié.

Pour les 16 œuvres analysées, les plages de valeurs sont les suivantes :

- Granularité (coarseness) : la valeur minimale est 0,28 et la valeur maximale 0,63. La moyenne est de 0,419 et l'écart-type est de 0,1060.
- Contraste : la valeur minimale est 0,09 et la valeur maximale 0,54. La moyenne est de 0,349 et l'écart-type est de 0,1355.
- Directionnalité : la valeur minimale est 0,07 et la valeur maximale 0,88. La moyenne est de 0,241 et l'écart-type est de 0,2460.
- Régularité : la valeur minimale est 0,09 et la valeur maximale 0,32. La moyenne est de 0,171 et l'écart-type est de 0,0716.
- Grossièreté (roughness) : la valeur minimale est 0,11 et la valeur maximale 0,57. La moyenne est de 0,371 et l'écart-type est de 0,1365.

Hormis pour la directionnalité, les écart-types sont tous passablement petits, ce qui indique une faible dispersion autour de la moyenne. L'étude de seulement 16 images, bien que très différentes,

ne permet pas de tirer une conclusion définitive, mais met en lumière une tendance qui peut indiquer que les plages de valeur utilisées pour la normalisation ne sont certainement pas optimisées. Il se peut que les valeurs maximales et minimales qui ont permis de définir cette plage soient des « cas extrêmes » et ne soient pas représentatives de la masse.

Ces valeurs étant inhérentes à la définition de certaines propriétés de la composition musicale, une évaluation plus affinée de ces caractéristiques permettrait d'obtenir plus de différentiation lors de la synthèse sonore.

Couleurs

Bien qu'il soit difficile de quantifier la justesse de l'outil d'extraction de la couleur dominante, les résultats du sondage réalisé (et discuté au chapitre 3.2.3) montrent que dans certains cas, la couleur dominante extraite ne correspond pas à celle perçue par les participants. La détection d'objets saillants ne s'est pas montrée très prometteuse (selon les résultats du sondage), et cela est dû en partie aux variations de structure des œuvres. La détection est aisée dans une composition simple, un portrait par exemple, mais peut être faussée ou inexacte pour une œuvre plus complexe ou abstraite. La figure 85 illustre diverses œuvres plus ou moins « complexes » pour la détection de couleur dominante.

Figure 85 Diverses œuvres dans des styles variés

Source : Données de l'auteur

Figure 86 Résultats de l'analyse de l'œuvre ?? de ??

Source : Données de l'auteur

En prenant cela en considération, un modèle de détection de la couleur dominante entraîné sur un dataset d'œuvres annotées par un large panel de participants pourrait être une piste intéressante à explorer et certainement engendrer une extraction plus alignée à la manière dont les couleurs sont perçues par l'humain.

En ce qui concerne le modèle de nommage des couleurs, la justesse est excellente et les scores F1 sont tous bons, voire très bons. Si les couleurs sont, dans la plupart des cas, nommées correctement, la complexité de ce modèle réside dans la gestion des « cas de bord », ceux où la couleur perçue par un humain peut différer de la nomenclature choisie par le modèle. La figure 87 est un bon exemple d'une telle situation. Les valeurs RGB de la couleur dominante mesurée sont [188, 92, 35], et un échantillon de cette couleur est illustré par la figure 88. Cette teinte, prédite comme « brun », pourrait être interprétée comme du orange.

Figure 87 Résultat de l'œuvre ?? de ??

Source : Données de l'auteur

Figure 88 Echantillon de couleur dominante

Source : Données de l'auteur

Plusieurs révisions du dataset ont été faites pour mieux définir les frontières entre certaines couleurs, mais il serait judicieux de réaliser un sondage auprès d'une batterie importante de participants pour obtenir des données qui permettraient d'entraîner un modèle correspondant mieux à la manière dont les couleurs sont perçues par l'œil humain.

Style

En ce qui concerne le modèle de reconnaissance du style artistique, la justesse de 55 % n'est évidemment pas très satisfaisante. Plusieurs modifications ont été faites pour améliorer le modèle et arriver à ce résultat - révision du dataset, augmentation des données, modification du nombre d'époques, modification de la valeur de patience – mais la marge de progression est encore grande. Les grandes différences de précision entre les styles pourraient être minimisées en faisant une révision

approfondie du jeu d'images utilisé pour l'entraînement, et éviter ainsi un déséquilibre entre les classes.

En outre, la matrice de confusion montre que de nombreuses œuvres réalistes ont été classifiées comme œuvres romantiques. Ce type d'erreurs peut en partie être dû aux similarités que l'on peut retrouver entre certains styles artistiques. La robustesse du modèle pourrait être améliorée en utilisant d'autres données, comme le nom des artistes ou la date de création de l'œuvre.

6.2. Synthèse sonore

Il est évidemment impossible d'évaluer une composition musicale de manière quantitative et son évaluation qualitative est sujette à de la subjectivité. Ainsi, les points discutés ci-après révèlent une appréciation personnelle, nécessairement subjective, et non une vérité scientifique.

L'une des intentions de ce travail était d'obtenir des résultats « musicaux » et cet objectif a été atteint. Les compositions générées respectent les gammes définies par les caractéristiques extraites de l'œuvre. Cependant, l'utilisation de peu d'éléments musicaux (accords, mélodie et texture granulaire) limite les débouchés. Les diverses compositions générées partagent ainsi certaines similitudes soniques qui ne représentent pas toujours les divergences et les particularités des œuvres d'art. Voici un détail de la conclusion résultant de l'analyse des 16 compositions générées.

Tout d'abord, étant donné que l'émotion dominante extraite de l'image (positive ou négative) influence le mode de la composition (majeur ou mineur), il y a, dans certains cas, un décalage perceptible entre la "gaieté" perçue dans l'œuvre et la "tristesse" transmise par la musique, notamment lorsque l'émotion est mal classifiée. La mauvaise classification du style créé également parfois une discordance entre l'image et le son.

Parmi les 8 styles artistiques, certains tendent à produire des compositions passablement dissonantes ou désaccordées, notamment l'art abstrait, l'expressionnisme et l'impressionnisme. Le taux de désaccordage est influencé par une caractéristique de texture, mais la dissonance est importante même lorsque cette valeur est faible. Une révision supplémentaire des paramètres initiaux pour ces styles semble donc être nécessaire.

Généralement, les compositions des œuvres dont le style est le même partagent beaucoup de similitudes sonores. C'est notamment le cas des compositions "abstract1.wav", "colorfield1.wav" et "colorfield2.wav" (voir figure 89), toutes classifiées comme œuvres abstraites, pour lesquelles la simplicité de la synthèse sonore ne permet pas de créer la même différenciation dans les compositions que dans les œuvres. Cet aspect est inévitable du fait que, dans le script de synthèse sonore, certains paramètres sont initialisés en fonction du style détecté. C'est toutefois un problème lorsque les images sont très divergentes.

Figure 89 Exemples d'œuvres pour lesquelles les compositions sont similaires

Source : Données de l'auteur

Globalement, toute la logique nécessaire à une synthèse sonore basée sur les caractéristiques d'une œuvre est en place, mais une complexification et une optimisation des modules utilisés mènerait à des résultats plus satisfaisants. Cette amélioration pourrait, dans un premier temps, se faire en se basant sur les paramètres utilisés pour les styles dont les compositions reflètent bien l'œuvre, ou en utilisant des méthode de synthèse sonore plus complexes. Pour obtenir d'encore meilleurs résultats, il pourrait être intéressant de demander à un groupe d'individus d'évaluer les compositions selon des critères définis tels que la balance des volumes, la qualité de la mélodie ou l'adéquation avec les caractéristiques perçues dans l'œuvre, et d'utiliser ces retours pour guider le développement du script SuperCollider.

Pour terminer sur une note positive, certains styles et certaines compositions générées reflètent très bien l'ambiance de l'œuvre. Parmi les 16 œuvres analysées, celles classifiées en peinture en champs de couleurs, réalisme et symbolisme ont donné lieu à de bons résultats. Les 3 œuvres de la figure 90 en sont un exemple :

Figure 90 Exemples d'œuvres pour lesquelles les compositions reflètent l'ambiance de l'image

Source : Données de l'auteur

Les fichiers audio liés à ces œuvres sont "abstract2.wav", "realism2.wav" et "symbolism1.wav". Pour l'image du haut, « abstract2.jpg », l'horizontalité des formes de l'œuvre se retrouve dans le mouvement des accords joués, et le minimalisme de l'image est également reflété dans la composition. Pour l'œuvre qui se trouve en bas à gauche, « realism2.jpg », on retrouve dans la musique l'atmosphère vaste et sereine du tableau. Enfin, la composition de la dernière image, « symbolism1.jpg », incarne le calme du paysage peint, et reflète son mouvement et ses courbes.

Sur l'aspect technique, un léger bug a été observé de manière sporadique. Dans certains cas, lors de l'arrêt de l'enregistrement, certains synthétiseurs ne sont pas stoppés entraînant ainsi une coupure nette à la fin du fichier audio. Cette erreur est certainement liée à une défaillance intermittente du serveur SuperCollider et n'a pas pu être corrigée.

6.3. Difficultés rencontrées

Plusieurs problématiques ont, par moments, freiné l'avancée de ce travail. En voici un détail :

Modèle de détection d'émotion

L'implémentation du modèle de détection d'émotion a été semée d'embûches. La méthode discutée au chapitre 3.2.1 impliquait l'utilisation du Group Lasso pour la classification des émotions.

Nous avons vu dans le détail de l'implémentation du modèle qu'un arbre de décision a finalement été utilisé. Ce changement de direction s'est fait après de nombreuses tentatives d'implémentations du Group Lasso qui ont toutes menées à des classifications erronées : erreurs dans le nombre de groupes utilisés, classification des œuvres dans une seule classe, erreurs liées à des incohérences de la taille de certains tableaux, etc. Il a donc été décidé en fin de projet de tester d'autres méthodes de classification plus basiques telles que la régression logistique et l'arbre de décision.

SuperCollider

La prise en main de SuperCollider a également été compliquée. Son langage de programmation, sclang, a une syntaxe très lourde et des messages d'erreur qui sont parfois très imprécis, nécessitant un temps de recherche et de résolution passablement long. Cette difficulté a exercé une influence sur certains choix liés à la complexité du patch de synthèse sonore dont les conséquences ont été discutées au chapitre précédent. L'implémentation de SuperCollider dans le pipeline s'est également avérée ardue. Le serveur n'étant pas équipé d'une carte son physique, il a fallu y installer un serveur audio (jack2) et le manque de documentation à ce sujet a rendu la tâche complexe.

Manque de connaissances

L'entraînement et l'utilisation de modèles d'intelligence artificielle était totalement nouveau pour moi. Ce manque de connaissance a imposé un important travail de recherche et de compréhension pour que ces modèles soient correctement implémentés et utilisés, et ce temps d'apprentissage a dû être utilisé au détriment parfois de l'optimisation de certains paramètres de ces mêmes modèles.

Ces diverses barrières ont mené à un manque de temps qui aurait été précieux pour l'optimisation des modèles de machine learning et du script de synthèse sonore.

6.4. Améliorations et perspectives

Nous allons ici compléter les quelques idées d'améliorations mentionnées aux chapitres 6.1 et 6.2, et proposer quelques perspectives qui pourraient étendre le cadre actuel de ce travail.

Datasets et modèles

Tout d'abord, afin d'obtenir des résultats plus précis lors de l'analyse des œuvres, la plupart des jeux de données utilisés pour l'entraînement des modèles pourraient être revus. Cette amélioration pourrait être amenée par l'agrandissement des datasets, l'optimisation de l'équilibre des classes, l'augmentation des données au moment de l'entraînement et, dans certains cas, l'utilisation de plus d'annotations (style artistique, date, certitude, etc.).

Compositions musicales

La complexité des compositions pourrait également être accrue. Cela pourrait se faire grâce à l'extraction de plus de caractéristiques de l'œuvre et à l'extension du lexique de sonification. Le patch de synthèse sonore devrait lui aussi être complexifié pour utiliser les nouvelles caractéristiques : un panel plus large d'émotions pourrait être utilisé, des séquences rythmiques pourraient complémenter les compositions actuelles et la quantité et la complexité des accords pourraient être étendues. De plus, le patch actuel n'utilise que des méthodes de synthèse passablement simples. Parmi les techniques discutées au chapitre 2.3, plusieurs pourraient être implémentées pour étendre les possibilités de composition et permettre des sonorités plus variées (et moins électroniques).

Pipeline

Enfin, le processus de sonification dans son ensemble pourrait être plus robuste. Le temps d'analyse de l'œuvre pourrait certainement être réduit en optimisant toutes les étapes du traitement et en étudiant la possibilité d'effectuer certaines tâches en parallèle par exemple. De plus, le prototype présenté dans ce travail ne permet pas d'effectuer plusieurs sonifications simultanément. Si la web app devait être utilisée en production, il serait nécessaire d'implémenter une gestion de threads et une priorisation des tâches. En outre, une gestion erreur plus complète devrait être mise en place. L'attention a été portée sur les erreurs qui apparaissaient lors des tests du pipeline, mais il y a vraisemblablement des erreurs qui n'ont pas été traitées.

Perspectives

Hormis ces améliorations directes, un travail de développement supplémentaire permettrait d'étendre l'approche purement artistique étudiée dans ce travail. Grâce à une extraction plus ciblée et complète des caractéristiques d'une œuvre, une narration décrivant l'image pourrait, par exemple, être générée. Un tel ajout permettrait à des personnes malvoyantes ou non-voyantes d'expérimenter des œuvres visuelles grâce à une couche musicale accompagnée d'une narration explicative. Une fonctionnalité comme celle-ci pourrait possiblement attiser l'intérêt de musées ou d'autres partenaires culturels.

7. Utilisation de l'IA

ChatGPT et Microsoft Copilot ont été une aide à diverses étapes de ce travail. Premièrement, pour la revue de la littérature, ces outils se sont montrés précieux pour résumer certains travaux de recherche complexes et ainsi rapidement estimer si leur contenu était cohérent pour mon travail.

ChatGPT a également été d'une grande aide lors de la création des prototypes et lors de l'implémentation des modèles finaux. À nouveau, l'utilisation de « Large Language Models » (LLM) a permis un gain de temps considérable pour le débogage de morceaux de code complexes. Les suggestions d'amélioration et d'optimisations faites par ChatGPT notamment se sont montrées très utiles.

L'IA a aussi été utilisée pour commenter le code utilisé dans le pipeline. Une révision des commentaires a été réalisée dans certains cas, mais la majeure partie des commentaires ont été générés automatiquement.

« SuperCollider Language », le langage de programmation utilisé par SuperCollider, m'était totalement inconnu. Certaines spécificités de ce langage ont pu être apprivoisées plus facilement grâce aux suggestions faites par les LLM, qui m'ont permis de comprendre l'architecture de base de ce langage et de guider mes recherches sur les différents forums dédiés à SuperCollider.

Enfin, en complément des relectures réalisées par mes proches, ChatGPT a été utilisé pour parcourir le rapport et relever des fautes d'orthographe ou syntaxiques.

8. CONCLUSION

Apports personnels

L'idée de ce travail est née de l'envie de mettre en commun les connaissances acquises lors de ma formation antérieure en ingénierie du son et production musicale, et celles apprises durant les 4 années passées à la HES-SO Valais/Wallis. Les fondements musicaux acquis lors de ma précédente formation n'ont pas été détaillés dans ce document, mais je crois avoir rendu les explications liées aux concepts musicaux suffisamment claires pour qu'elles puissent être comprises par tous.

Ce projet m'a permis d'acquérir de nouvelles compétences, tant sur le plan technique que méthodologique. L'utilisation de nouveaux langages de programmation et de logiciels spécialisés a non seulement enrichi mes connaissances techniques, mais a également alimenté un intérêt accru pour le machine learning et plus largement pour l'intelligence artificielle.

La complexité de prise en main de ces nouvelles technologies a parfois été source de frustration, notamment lorsque certaines facettes du projet ne pouvaient être explorées plus en profondeur en raison des contraintes de temps. Ce sentiment a mis en lumière la difficulté de devoir jongler entre le désir d'approfondir certains aspects et la nécessité de respecter un calendrier serré.

Sur le plan méthodologique, une planification rigoureuse ainsi que l'évaluation réaliste du temps de travail et la priorisation des tâches ont fait de ce travail une expérience très formatrice.

Avec du temps et de la persévérance, tous les obstacles initiaux ont pu être surmontés. Chaque difficulté rencontrée a été l'occasion de renforcer mes compétences et d'adopter une approche plus résiliente face aux défis.

Respect des objectifs

L'objectif principal de ce travail était d'étudier les possibilités de sonification d'œuvres d'art, plus précisément de peintures, en créant un outil capable d'extraire des propriétés inhérentes à une œuvre et de les utiliser pour générer une composition musicale propre à l'œuvre en question. Cette approche artistique visait à offrir une appréciation nouvelle, une perspective supplémentaire pour la personne qui fait l'expérience de l'œuvre.

Pour ce faire, un travail de recherche sur les méthodes d'analyse d'image et de synthèse sonore existantes a permis de définir un lexique de sonification qui met en correspondance les particularités d'une peinture et les caractéristiques musicales de la composition, ainsi que de faire les divers choix technologiques qui ont guidé ce projet. Il est important de mentionner que le système tonal utilisé

dans ce lexique est basé sur un système musical occidental et ne peut donc prétendre à une compréhension universelle.

Les résultats discutés dans les chapitres précédents mettent en lumière divers aspects à améliorer mais permettent d'évaluer la faisabilité d'un tel processus. Il était question, dans la définition initiale du travail, de créer une forme d'appréciation d'œuvres visuelles accessible aux personnes malvoyantes. Le choix a été fait lors du lancement du projet de prendre un chemin plus artistique. Ainsi, les résultats en l'état ne permettraient pas à une personne atteinte de cécité de distinguer avec précision les caractéristiques qui composent une peinture. Toutefois, ces résultats permettent de saisir les possibilités, les challenges et les étapes qui pourraient être nécessaires à la réalisation d'une transformation plus complète et détaillée des caractéristiques visuelles d'une œuvre en attributs sonores.

" L'art défie la technologie, et la technologie inspire l'art." — John Lasseter

9. RÉFÉRENCES

Abstract Expressionism. (n.d., n.d. n.d.). Consulté le Juin 21, 2024, sur Artland: https://magazine.artland.com/abstract-expressionism/

Adam. (n.d., n.d. n.d.). Consulté le Août 13, 2024, sur Keras: https://keras.io/api/optimizers/adam/

Alameda-Pineda, X., Ricci, E., Yan, Y., & Sebe, N. (2016). Recognizing Emotions from Abstract Paintings. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Analyse sémantique latente probabiliste. (2018, Mai 9). Consulté le Juillet 26, 2024, sur Wikipedia: https://fr.wikipedia.org/wiki/Analyse_s%C3%A9mantique_latente_probabiliste

Angelova, A., & Zhu, S. (2013). Efficient object detection and segmentation for fine-grained recognition. *IEEE Conference on Computer Vision and Pattern Recognition*.

Art Movement : Color Field Painting. (n.d., n.d. n.d.). Consulté le Juin 21, 2024, sur Artland: https://magazine.artland.com/art-movement-colour-field-painting/

Art movement : Expressionism. (n.d., n.d. n.d.). Consulté le Juin 21, 2024, sur Artland: https://magazine.artland.com/art-movement-expressionism/

Art Movement : Impressionism. (n.d., n.d. n.d.). Consulté le Juin 21, 2024, sur Artland: https://magazine.artland.com/art-movement-impressionism/

Art movement : Symbolism. (n.d., n.d. n.d.). Consulté le Juin 21, 2024, sur Artland: https://magazine.artland.com/art-movement-symbolism/

ART500K Dataset. (2019, Juillet 12). Consulté le Juillet 31, 2024, sur Deepart Hkust Edu: https://deepart.hkust.edu.hk/ART500K/art500k.html

Back-End Définition. (n.d., n.d. n.d.). Consulté le Août 12, 2024, sur Centre Européen de Formation: https://www.centre-europeen-formation.fr/blog/informatique/back-end-definition/

Berlin, B., & Kay, P. (1971). Basic Color Terms: Their Universality and Evolution. *International Journal of American Linguistics, Vol. 37, No. 4*.

Bharati, M. H., Liu, J., & MacGregor, J. F. (2004). Image Texture Analysis: Methods and Comparisons. *Chemometrics and Intelligent Laboratory Systems 72*, 57-71.

Boshernitsan, M., & Downes, M. (1997). Visual Programming Languages: A Survey.

Brown, R., DuTell, V., Walter, B., Rosenholtz, R., Shirley, P., McGuire, M., & Luebke, D. (2021). Efficient Dataflow Modeling of Peripheral Encoding in the Human Visual System. *n.d.*

Burnett, M. M. (1994). Visual Programming. *ACM SIGPLAN OOPS Messenger*, 127-129.

Canic, E., & Pachur, T. (2014, Août 22). Serial-position effects in preference construction: a sensitivity analysis of the pairwise-competition model. *Frontiers in Psychology*. Consulté le Juin 22, 2024, sur Frontiers: https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2014.00902/full#B11

Carson, C., Belongie, S., Greenspan, H., & Malik, J. (2002). Blobworld: Image segmentation using Expectation-Maximization and its.

Chavan, A. (2023, Mai 26). *RGB-Color-Classifier-with-Deep-Learning-using-Keras-and-Tensorflow*. Consulté le Juillet 22, 2024, sur Github: https://github.com/AjinkyaChavan9/RGB-Color-Classifier-with-Deep-Learning-using-Keras-and-Tensorflow/tree/master

Chavda, S., & M Mahesh, G. (2019). Recent evaluation on Content Based Image Retrieval. *International Journal of Computer Sciences and Engineering 7*, 325-329.

Cheng, M.-M., Zhang, G.-X., Mitra, N. J., & Huang, X. (2011). Global Contrast Based Salient Region Detection. *Computer Vision and Pattern Recognition (CVPR), 2011*.

Clavier à lumières. (2024, Janvier 5). Consulté le Juillet 26, 2024, sur Wikipedia: https://en.wikipedia.org/wiki/Clavier_%C3%A0_lumi%C3%A8res

Comment éviter les biais de sondage. (n.d., n.d. n.d.). Consulté le Juillet 26, 2024, sur Survey Monkey: https://fr.surveymonkey.com/learn/survey-best-practices/how-to-avoid-common-types-survey-bias/

Cubism. (n.d., n.d. n.d.). Consulté le Juin 21, 2024, sur Artland: https://magazine.artland.com/cubism/

De Poli, G. (2023). Sound models for synthesis: a structural viewpoint. *Music / Technology*.

Debnath, S., Aman, & Changder, S. (2018). Automatic Detection of Regular Geometrical Shapes in Photograph using Machine Learning Approach. *Tenth International Conference on Advanced Computing (ICoAC)*.

Deguerre, B., Chatelain, C., & Gasso, G. (2021). Object Detection in the DCT Domain: is Luminance the Solution? *2020 25th International Conference on Pattern Recognition (ICPR)*.

Distance euclidienne. (2024, Août 6). Consulté le Août 12, 2024, sur Wikipédia: https://fr.wikipedia.org/wiki/Distance_euclidienne

Editors, A. ((s.d.), (s.d.) (s.d.)). *The Most Famous Art Movements and Styles*. Consulté le Juin 29, 2024, sur Artland Magazine: https://magazine.artland.com/

El Abbadi, N., & Al Saadi, L. (2013). Automatic Detection and Recognize Different. *International Journal of Computer Science Issues, Vol. 10, Issue 6, No 1*.

El Korchi, A., & Ghanou, Y. (2020). 2D geometric shapes dataset – For machine learning and pattern recognition. *Data in Brief 32*.

Flask (framework). (2024, Février 3). Consulté le Juillet 31, 2024, sur Wikipédia: https://fr.wikipedia.org/wiki/Flask_(framework)

Fonctionnement de l'outil Calculer la précision pour la détection d'objets. (n.d., n.d. n.d.). Consulté le Juillet 27, 2024, sur ArcGIS Pro: https://pro.arcgis.com/fr/pro-app/latest/tool-reference/image-analyst/how-compute-accuracy-for-object-detection-works.htm#:~:text=Par%20exemple%2C%20si%20le%20mod%C3%A8le%20a%20correctement%20d%C3%A9tect%C3%A9%2075%20arbres,est%20de%2075%20pour%20cent.&te

Front-End. (n.d., n.d. n.d.). Consulté le Août 12, 2024, sur Link Web: https://linkweb.fr/creation-site-internet-toulouse/front-end/#:~:text=Qu'est%2Dce%20que%20le,ou%20une%20application%20web%20mobile).

Gandhi, M., Kamdar, J., & Shah, M. (2020). Preprocessing of Non-symmetrical Images for Edge Detection. *Augmented Human Research 5*.

Gresham-Lancaster, S. (2012). Relationships of sonification to music and sound art. *AI & SOCIETY 27*.

Groves, A. M., Singh, Y., Dempsey, E., Molnar, Z., Austin, T., El-Khuffash, A., & de Boode, W. P. (2018). Introduction to neonatologist-performed echocardiography. *Pediatric Research volume 84*, 1-12.

Image Processing Using Machine Learning. (n.d., n.d. n.d.). Consulté le Août 14, 2024, sur Javatpoint: https://www.javatpoint.com/image-processing-using-machine-learning#:~:text=By%20training%20algorithms%20on%20large,for%20image%20analysis%20and%20processing.

Itten, J. (1961). *The Art of Color : The subjective experience and objective rationale of color*.

Joblib: running Python functions as pipeline jobs. (n.d., n.d. n.d.). Consulté le Juillet 31, 2024, sur Joblib: https://joblib.readthedocs.io/en/stable/

Jono. (2019). *des-Objets-Reconnus*. Consulté le Juin 22, 2024, sur Jono: https://www.jono.fyi/des-Objets-Reconnus

Karayev, S., Trentacoste, M., Han, H., Agarwala, A., Darrell, T., Hertzmann, A., & Winnemoeller, H. (2014). Recognizing Image Style. *Proc. British Machine Vision Conference*.

Kassel, R. (2020, Juin 22). *Machine Learning & Clustering : Focus sur l'algorithme CAH*. Consulté le Août 12, 2024, sur DataScientest: https://datascientest.com/machine-learning-clustering-focus-sur-algorithme-cah#:~:text=Le%20clustering%20est%20une%20discipline,homog%C3%A8nes%20ayant%20des%20caract%C3%A9ristiques%20communes.

Krajnc, A. (2024, Mai 3). *VGG, Modèle de Transfert Learning*. Consulté le Août 12, 2024, sur Jedha: https://www.jedha.co/formation-ia/vgg-transfert-learning#:~:text=VGG16%20est%20un%20r%C3%A9seau%20neuronal,une%20am%C3%A9lioration%20du%20VGG%2D16.

Kramer, G., Walker, B., Bonebright, T., Cook, P., Flowers, J. H., Miner, N., & Neuhoff, J. (2010). Sonification Report: Status of the Field and Research Agenda. *Faculty Publications, Department of Psychology - University of Nebraska*.

Lafay, G., Sustrac, Q., Mas, P., & Lavallee, A. (2021). *neural-art*. Consulté le Juin 18, 2024, sur Github: https://github.com/gregoirelafay/neural-art/tree/master

Lecoutre, A., Negrevergne, B., & Yger, F. (2017). Recognizing Art Style Automatically in Painting with Deep Learning. *Proceedings of the Ninth Asian Conference on Machine Learning*, (pp. 327-342).

Lee, M. (2019, Février 27). *Tamura-In-Python*. Consulté le Juillet 31, 2024, sur GitHub: https://github.com/MarshalLeeeeee/Tamura-In-Python

Leptokurtique examen de la fonction generatrice de moments et de son role. (2024, Juin 2). Consulté le Août 13, 2024, sur FasteCapital: https://fastercapital.com/fr/contenu/Leptokurtique---examen-de-la-fonction-generatrice-de-moments-et-de-son-role.html#:~:text=Les%20distributions%20leptokurtiques%20pr%C3%A9sentent%20des,qui%20affecte%20directement%20l'aplatissement.

Libert, C. (2020). *Art_Classifying_Project*. Consulté le Juin 14, 2024, sur Github: https://github.com/Camillelib/Art_Classifying_Project

Liu, T., Sun, J., Zheng, N.-N., & Tang, X. (2007). Learning to Detect A Salient Object. *Computer Vision and Pattern Recognition, 2007. CVPR '07.*

Machine Learning : Cours de base. (n.d., n.d. n.d.). Consulté le Juillet 26, 2024, sur Developers Google: https://developers.google.com/machine-learning/crash-course/classification/accuracy?hl=fr

Manjunath, B. S., Ohm, J., V. Vasudevan, V., & Yamada, A. (2001). Color and Texture Descriptors. *IEEE Transactions on Circuits and Systems for Video Technology 11*, 703-715.

Marchenko, Y., Chua, T.-S., & Aristarkhova, I. (2005). Analysis and Retrieval of Paintings Using Artistic Color Concepts. *IEEE International Conference on Multimedia and Expo*.

Martinez-Zorrilla, D. (2008). *Synthesizers: A Brief Introduction.*

matplotlib.pyplot.hist. (n.d., n.d. n.d.). Consulté le Août 14, 2024, sur Matplotlib: https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.hist.html

Matrix completion. (2024, January 31). Consulté le Juillet 26, 2024, sur Wikipedia: https://en.wikipedia.org/wiki/Matrix_completion

McKee, A. (2023, Juin n.d.). *Seeing Like a Machine: A Beginner's Guide to Image Analysis in Machine Learning*. Consulté le Août 14, 2024, sur Datacamp: https://www.datacamp.com/tutorial/seeing-like-a-machine-a-beginners-guide-to-image-analysis-in-machine-learning?dc_referrer=https%3A%2F%2Fwww.google.com%2F

Mean shift. (2023, Septembre 6). Consulté le Août 12, 2024, sur Wikipédia: https://en.wikipedia.org/wiki/Mean_shift

Minca, T., Zaykov, Y., & Tims, J. (2005, Novembre 18). *TrueSkill™ Ranking System*. Consulté le Juin 6, 2024, sur Microsoft: https://www.microsoft.com/en-us/research/project/trueskill-ranking-system/

Mirmehdi, M., Xie, X., & Suri, J. (2008). *Handbook of Texture Analysis.*

Munir, S. (2019). *Classifying-Painting-Art-Style-With-Deep-Learning*. Consulté le Juin 15, 2024, sur Github: https://github.com/samiramunir/Classifying-Painting-Art-Style-With-Deep-Learning

Musique. (2024, Juin 4). Consulté le Juin 27, 2024, sur Wikipedia: https://fr.wikipedia.org/wiki/Musique#Musique_stochastique

Neil Harbisson. (2024, Juin 20). Consulté le Juin 27, 2024, sur Wikipedia: https://fr.wikipedia.org/wiki/Neil_Harbisson

Newjazz. (2014, Septembre 30). *Les différentes formes d'ondes en synthèse sonore*. Consulté le Juin 27, 2024, sur AudioFanzine: https://fr.audiofanzine.com/synthese-sonore-acoustique/editorial/dossiers/les-ondes-en-pleine-forme.html

NumPy. (2024, Février 12). Consulté le Juillet 31, 2024, sur Wikipédia: https://fr.wikipedia.org/wiki/NumPy

numpy.expand_dims. (n.d., n.d. n.d.). Consulté le Août 13, 2024, sur NumPy: https://numpy.org/doc/stable/reference/generated/numpy.expand_dims.html

OpenCV. (2023, Octobre 23). Consulté le Juillet 31, 2024, sur Wikipédia: https://fr.wikipedia.org/wiki/OpenCV

O'Toole, P., Glowinski, D., Pitt, I., & Mancini, M. (2021). When Emotions are Triggered by Single Musical Notes: Revealing the Underlying Factors of Auditory-Emotion Associations. *ICMI '21: INTERNATIONAL CONFERENCE ON MULTIMODAL INTERACTION*.

Ou, L.-C., Luo, M. R., Woodcock, A., & Wright, A. B. (2004). A study of colour emotion and colour preference. Part II: Colour emotions for two-colour combinations. *Color Research & Application 29*, 292-298.

Pandas. (2024, Avril 26). Consulté le Juillet 31, 2024, sur Wikipédia: https://fr.wikipedia.org/wiki/Pandas

Peinture romantique. (2024, Juin 9). Consulté le Juin 21, 2024, sur Wikipédia: https://fr.wikipedia.org/wiki/Peinture_romantique

Pipeline : qu'est-ce que c'est ? (n.d., n.d. n.d.). Consulté le Août 13, 2024, sur Futura: https://www.futura-sciences.com/tech/definitions/informatique-pipeline-2597/

Pošćić, A., & Kreković, G. (2018). Ecosystems of Visual Programming Languages for Music Creation: A Quantitative Study. *Journal of the Audio Engineering Society. Audio Engineering Society 66*, 486-494.

Pramod, O. (2024, Mars 6). *Callbacks: Your Secret Weapon in Machine Learning*. Consulté le Août 13, 2024, sur Medium: https://medium.com/@ompramod9921/callbacks-your-secret-weapon-in-

machine-learning-b08ded5678f0#:~:text=Callbacks%20in%20machine%20learning%20are,for%20greater%20control%20and%20flexibility.

RandomForestClassifier. (n.d., n.d. n.d.). Consulté le Juillet 22, 2024, sur Scikit-learn: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Rao, N. S., Nowak, R., Cox, C. R., & Rogers, T. T. (2015). Classification With the Sparse Group Lasso. *IEEE Transactions on Signal Processing 64*, 1-1.

Réalisme (peinture). (2024, Mai 16). Consulté le Juin 21, 2024, sur Wikipédia: https://fr.wikipedia.org/wiki/R%C3%A9alisme_(peinture)

Sainui, J., & Tongsamrit, M. (2020). Color Naming for Describing Object in Image using Different Classification Algorithms and Color Spaces.

Sartori, A., Culibrk, D. R., Yan, Y., & Sebe, N. (2015). Who's afraid of Itten: Using the art theory of color combination to analyze emotions in abstract paintings. *Proceedings of the 23rd ACM international conference on Multimedia (MM'15)*.

Sartori, A., Yanulevskaya, V., Akdag Salah, A., Uijling, J., & Bruni, S. (2015). Affective Analysis of Professional and Amateur Abstract Paintings Using Statistical Analysis and Art Theory. *The ACM Transactions on Interactive Intelligent Systems 5*, 1–27.

Schauerte, B., & Stiefelhagen, R. (2012). Learning robust color name models from web images. *(ICPR), 2012 21st International Conference on Pattern Recognition*.

Schönberg, A. (1948). *Theory of Harmony*.

Scikit-image. (2020, Août 27). Consulté le Juillet 31, 2024, sur Wikipédia: https://fr.wikipedia.org/wiki/Scikit-image

Scikit-learn. (2024, Avril 23). Consulté le Juillet 31, 2024, sur Wikipédia: https://fr.wikipedia.org/wiki/Scikit-learn

Selecting the number of clusters with silhouette analysis on KMeans clustering. (n.d., n.d. n.d.). Consulté le Juillet 22, 2024, sur Scikit-learn: https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html

Serra, M.-H. (1993). Stochastic Composition and Stochastic Timbre: GENDY3 by Iannis Xenakis. *Perspectives of New Music, Vol. 31, No. 1 (Winter, 1993)*, 236-257.

Serra, X. (2007). State of the Art and Future Directions in Musical Sound Synthesis. *Multimedia Signal Processing 2007*.

silhouette_score. (n.d., n.d. n.d.). Consulté le Juillet 31, 2024, sur Scikit-learn: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html

Siraudin, A. (2022, Mai 4). *Color Naming with Python*. Consulté le Juillet 25, 2024, sur Medium: https://medium.com/@antoine.siraudin/color-naming-with-python-dbfa51d55a4c

Sonochromatism. (2023, Juillet 8). Consulté le Juin 27, 2024, sur Wikipedia: https://en.wikipedia.org/wiki/Sonochromatism#:~:text=The%20Sonochromatic%20Music%20Scale%20is,light%20frequencies%20to%20sound%20frequencies.

Synesthésie. (2024, Juin 15). Consulté le Juin 29, 2024, sur Wikipédia: https://fr.wikipedia.org/wiki/Synesth%C3%A9sie

Synthèse sonore. ((s.d.), (s.d.) (s.d.)). Consulté le Juin 27, 2024, sur Larousse: https://www.larousse.fr/encyclopedie/musdico/synth%C3%A8se_sonore/170270

Synthèse sonore. (2024, Mars 14). Consulté le Juin 27, 2024, sur Wikipedia: https://fr.wikipedia.org/wiki/Synth%C3%A8se_sonore

Synthèse sonore additive. (2023, Février 18). Consulté le Août 12, 2024, sur Wikipédia: https://fr.wikipedia.org/wiki/Synth%C3%A8se_sonore_additive

Synthèse sonore soustractive. (2024, Mars 14). Consulté le Août 13, 2024, sur Wikipédia: https://fr.wikipedia.org/wiki/Synth%C3%A8se_sonore_soustractive#:~:text=La%20synth%C3%A8se%20sonore%20soustractive%20est,l'aide%20de%20filtres%20fr%C3%A9quentiels.

Tamura, H., Mori, S., & Yamawaki, T. (1978). Textural Features Corresponding to Visual Perception. *IEEE Transactions on Systems, Man, and Cybernetics*, 460-473.

TensorFlow. (2024, Juillet 13). Consulté le Juillet 31, 2024, sur Wikipédia: https://fr.wikipedia.org/wiki/TensorFlow

tf.keras.applications.VGG16. (2024, Juin 7). Consulté le Août 8, 2024, sur TensorFlow: https://www.tensorflow.org/api_docs/python/tf/keras/applications/VGG16

tf.keras.callbacks.EarlyStopping. (2024, Juin 7). Consulté le Août 13, 2024, sur TensorFlow: https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/EarlyStopping

tf.keras.callbacks.ReduceLROnPlateau. (2024, Juin 7). Consulté le Août 13, 2024, sur TensorFlow: https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/ReduceLROnPlateau

Transformation de Fourrier rapide. (2023, Octobre 23). Consulté le Juillet 26, 2024, sur Wikipedia: https://fr.wikipedia.org/wiki/Transformation_de_Fourier_rapide

TrueSkill™ Ranking System. (2005, Novembre 18). Consulté le Juillet 31, 2024, sur Microsoft: https://www.microsoft.com/en-us/research/project/trueskill-ranking-system/#:~:text=The%20TrueSkill%20value%20is%20then,%2D3*8.333%20%3D%200.

Valdez, P., & Mehrabian, A. (1994). Effects of Color on Emotions. *Journal of Experimental Psychology General 123*, 394-409.

van de Weijer, J., Schmid, C., & Verbeek, J. (2007). Learning Color Names from Real-World Images. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*. Minneapolis.

Vannet, J. (2007, Juillet 14). *La composition musicale*. Consulté le Juin 28, 2024, sur JackGuitar: https://jackguitar.com/la-composition-musicale/#:~:text=Les%20%C3%A9l%C3%A9ments%20du%20langage%20musical,le%20caract%C3%A8re%20et%20le%20style.&text=Une%20m%C3%A9lodie%20d%C3%A9signe%20l'impression,de%20sons%20de%20diff%C3%A9rentes%20hauteurs.

Weingerl, P., Hladnik, A., & Javoršek, D. (2020). Development of a machine learning model for extracting image prominent colors. *Color Research & Application 45*.

Yanulevskaya, V., Uijlings, J., Bruni, E., & Sartori, A. (2012). In the eye of the beholder: Employing statistical analysis and eye tracking for analyzing abstract paintings. *Proceedings of the 20th ACM international conference on Multimedia*.

Yapi, L.-A. (n.d., n.d. n.d.). *La régression linéaire : Ridge, Lasso et Elastic Net (1/2)*. Consulté le Août 13, 2024, sur Alliage: https://www.alliage-ad.com/data-science/la-regression-lineaire-ridge-lasso-et-elastic-net/#:~:text=La%20R%C3%A9gression%20Lasso%20nous%20permet,donc%20%C3%AAtre%20retir%C3%A9es%20du%20mod%C3%A8le.

Zheng, Y., Zhou, X. S., Georgescu, B., & Zhou, S. K. (2006). Example Based Non-rigid Shape Detection. *Computer Vision - ECCV 2006, 9th European Conference on Computer Vision*.

10. ANNEXES

10.1. Choix de la méthodes d'analyse des émotions

NLMC

1. Adaptation aux besoins (4/5) : Cette méthode utilise une approche avancée pour capturer les émotions complexes dans les œuvres d'art, ce qui est essentiel pour un projet de sonification.

2. Facilité de réalisation (3/5) : La complétion de matrice non linéaire est une technique sophistiquée qui nécessite une expertise avancée en machine learning et en optimisation.

3. Bibliothèques existantes (3/5) : Il existe des bibliothèques pour la complétion de matrice, mais elles ne sont pas aussi courantes que celles pour les SVM ou le Group Lasso.

4. Articles détaillés (4/5) : Les articles fournissent une base théorique solide et des détails sur l'implémentation.

5. Performance et Précision (4/5) : Cette méthode est très performante pour la classification des émotions en utilisant des relations multi-label.

6. Flexibilité (4/5) : Capable de gérer plusieurs types d'étiquettes et de relations complexes.

7. Coût computationnel (3/5) : Demande des ressources computationnelles importantes pour entraîner et exécuter.

8. Disponibilité de jeux de données annotés (3/5) : Besoin de jeux de données annotés spécifiques, ce qui peut limiter les options disponibles.

Sparse Group Lasso

1. Adaptation aux besoins (4/5) : Utilise des caractéristiques de couleur et de texture qui sont bien adaptées pour capturer les émotions dans les œuvres d'art.

2. Facilité de réalisation (4/5) : Méthode relativement simple à implémenter avec des bibliothèques disponibles.

3. Bibliothèques existantes (4/5) : Le Lasso et Group Lasso sont bien supportés par des bibliothèques comme Scikit-learn.

4. Articles détaillés (5/5) : Les articles fournissent des détails clairs et pratiques pour l'implémentation.

5. Performance et Précision (4/5) : Bonne précision pour la classification des émotions, démontrée par les résultats expérimentaux.

6. Flexibilité (4/5) : Flexible pour s'adapter à différents types d'œuvres d'art.

7. Coût computationnel (4/5) : Moins coûteuse computationnellement comparée à des méthodes plus complexes comme la complétion de matrice.

8. Disponibilité de jeux de données annotés (3/5) : Jeux de données existants ou faciles à créer pour l'entraînement du modèle.

SVM

1. Adaptation aux besoins (3/5) : Utilise des descripteurs standard qui peuvent ne pas capturer toutes les nuances des émotions dans les œuvres d'art abstraites.

2. Facilité de réalisation (4/5) : Méthode simple à mettre en œuvre avec des outils courants.

3. Bibliothèques existantes (5/5) : Les SVM sont largement supportés par des bibliothèques comme Scikit-learn.

4. Articles détaillés (4/5) : Les articles sont bien documentés et faciles à suivre.

5. Performance et Précision (4/5) : Bonne précision de classification démontrée par les études expérimentales.

6. Flexibilité (3/5) : Moins flexible pour capturer des nuances complexes d'émotions.

7. Coût computationnel (4/5) : Efficace en termes de ressources computationnelles.

8. Disponibilité de jeux de données annotés (4/5) : Facile à trouver des jeux de données annotés pour l'entraînement.

SVM + texte

1. Adaptation aux besoins (4/5) : Combine des descripteurs visuels et textuels pour une analyse plus riche des émotions.

2. Facilité de réalisation (3/5) : Intégrer des données textuelles peut ajouter de la complexité à l'implémentation.

3. Bibliothèques existantes (4/5) : Les SVM et les outils de traitement de texte sont bien supportés par des bibliothèques existantes.

4. Articles détaillés (4/5) : Les articles fournissent des détails utiles pour l'implémentation.

5. Performance et Précision (4/5) : Bonne précision grâce à l'intégration des descripteurs textuels et visuels.

6. Flexibilité (4/5) : Flexible pour capturer des nuances émotionnelles complexes grâce à l'ajout de données textuelles.

7. Coût computationnel (3/5) : Plus coûteuse en ressources en raison de l'intégration des données textuelles.

8. Disponibilité de jeux de données annotés (4/5) : Les jeux de données annotés incluant des descriptions textuelles peuvent être plus rares mais sont trouvables.

10.2. Choix du logiciel de synthèse sonore

Pure Data

1. Connaissances du logiciel (2/5) : Petite expérience des éléments basiques de l'application

2. Compatibilité avec le système d'exploitation du serveur (5/5) : Pure Data est largement compatible avec les systèmes Linux, y compris Ubuntu. Il est régulièrement mis à jour et bien supporté.

3. Facilité de prise en main (3/5) : Pure Data est un environnement de programmation graphique qui peut être plus intuitif pour les débutants en programmation sonore. Cependant, la complexité de certains concepts peut nécessiter du temps pour une maîtrise complète.

4. Possibilités de lecture et d'interprétation de fichiers (JSON, TXT ou autres) (3/5) : Pure Data peut lire des fichiers texte et certains formats de données via des objets externes, mais la manipulation de JSON nécessite des extensions spécifiques et peut être moins directe.

5. Disponibilité de documentation (2/5) : Il existe une documentation extensive, des tutoriels et une communauté active autour de Pure Data. Cependant, la qualité et la clarté peuvent varier.

6. Adhésion aux besoins du projet (4/5) : Pure Data est bien adapté à la sonification interactive, surtout avec une interface graphique. Cependant, la manipulation avancée de données externes peut être moins directe sans extensions.

SuperCollider

1. Connaissances du logiciel (2/5) : Petite expérience des éléments basiques de l'application

2. Compatibilité avec le système d'exploitation du serveur (5/5) : SuperCollider fonctionne également très bien sous Ubuntu et autres distributions Linux. Il bénéficie d'une bonne intégration dans ces environnements.

3. Facilité de prise en main (2/5) : SuperCollider utilise un langage de programmation textuel (sclang), qui peut être moins accessible pour les débutants, surtout ceux sans expérience en programmation. Cependant, il offre une grande flexibilité et puissance pour les utilisateurs expérimentés.

4. Possibilités de lecture et d'interprétation de fichiers (JSON, TXT ou autre) (4/5) : SuperCollider a de bonnes capacités de manipulation de fichiers, y compris la lecture de fichiers JSON et d'autres formats, grâce à sa bibliothèque de classes et d'extensions. Cela rend la lecture et l'interprétation des données plus flexibles et intégrées.

5. Disponibilité de documentation (4/5) : SuperCollider dispose également d'une documentation complète et d'une communauté engagée. Les guides et tutoriels sont nombreux et bien détaillés, ce qui aide à surmonter la courbe d'apprentissage initiale.

6. Adhésion aux besoins du projet (4/5) : SuperCollider est extrêmement puissant pour la génération et la manipulation audio basée sur des données, y compris la sonification. Sa capacité à traiter des fichiers JSON et sa flexibilité en programmation en font un outil très pertinent pour ce type de projet.

10.3. Choix de la méthode d'annotation des œuvres

Échelle de Likert

1. Facilité de mise en œuvre (5/5) : Très facile à mettre en œuvre, nécessite peu de préparation.
2. Richesse des données (3/5) : Données quantitatives limitées à une échelle fixe.
3. Temps requis par participant (2/5) : Chaque participant doit évaluer chaque image, ce qui peut être long et fatigant.
4. Simplicité pour les participants (5/5) : Simple, direct et intuitif.
5. Précision des annotations (3/5) : Moyenne, dépend des interprétations individuelles de l'échelle.
6. Analyse des données (5/5) : Facile à analyser quantitativement.
7. Robustesse aux biais de réponse (3/5) : Sensible à certains biais de réponse.

TrueSkill

1. Facilité de mise en œuvre (3/5) : Passablement complexe à organiser et nécessite des outils pour le traitement des comparaisons.
2. Richesse des données (4/5) : Donne des relations ordinales riches.
3. Temps requis par participant (4/5) : Moins d'évaluations individuelles nécessaires pour obtenir un classement global.
4. Simplicité pour les participants (4/5) : Simple mais répétitif.
5. Précision des annotations (4/5) : Précision relative élevée.
6. Analyse des données (3/5) : Analyse plus complexe nécessitant des méthodes spécifiques.
7. Robustesse aux biais de réponse (4/5) : Peu sensible aux biais cognitifs.

Annotations libres

1. Facilité de mise en œuvre (4/5) : Très facile à collecter.
2. Richesse des données (5/5) : Données qualitatives très riches.
3. Temps requis par participant (3/5) : Variable, mais généralement modéré.
4. Simplicité pour les participants (3/5) : Plus difficile à articuler des réponses libres.
5. Précision des annotations (4/5) : Précision élevée mais subjective.
6. Analyse des données (2/5) : Analyse textuelle complexe.
7. Robustesse aux biais de réponse (3/5) : Variabilité élevée due aux réponses libres.

10.4. Résultats de l'analyse d'œuvres

Couleur dominante : brun

Style labellisé : art abstrait

Style détecté : art abstrait

Probabilité par style : abstrait: 55,50 - impressionnisme: 0,44 - réalisme: 0,59 - romantisme: 0,36 - symbolisme: 1,41 - peinture en champs de couleur: 36,24 - cubisme: 2,37 - expressionnisme: 3,09

Émotion détectée : négative

Couleur dominante : bleu foncé

Style labellisé : art abstrait

Style détecté : peinture en champs de couleur

Probabilité par style : abstrait: 14,08 - impressionnisme: 0,22 - réalisme: 0,19 - romantisme: 0,23 - symbolisme: 4,67 - peinture en champs de couleur: 79,43 - cubisme: 0,59 - expressionnisme: 0,59

Émotion détectée : négative

Couleur dominante : rose

Style labellisé : peinture en champs de couleur

Style détecté : peinture en champs de couleur

Probabilité par style : abstrait: 1,29 - impressionnisme: 0,00 - réalisme: 0,00 - romantisme: 0,00 - symbolisme: 0,10 - peinture en champs de couleur: 98,43 - cubisme: 0,14 - expressionnisme: 0,02

Émotion détectée : positive

Couleur dominante : brun

Style labellisé : peinture en champs de couleur

Style détecté : peinture en champs de couleur

Probabilité par style : abstrait: 0,55 - impressionnisme: 3,99 - réalisme: 8,10 - romantisme: 2,25 - symbolisme: 0,00 - peinture en champs de couleur: 99,42 - cubisme: 0,03 - expressionnisme: 0,00

Émotion détectée : négative

Couleur dominante : brun

Style labellisé : cubisme

Style détecté : cubisme

Probabilité par style : abstrait: 2,56 - impressionnisme: 7,67 - réalisme: 9,70 - romantisme: 1,42 - symbolisme: 0,00 - peinture en champs de couleur: 0,30 - cubisme: 96,74 - expressionnisme: 0,39

Émotion détectée : négative

Couleur dominante : gris

Style labellisé : cubisme

Style détecté : cubisme

Probabilité par style : abstrait: 6,35 - impressionnisme: 5,25 - réalisme: 5,30 - romantisme: 5,80 - symbolisme: 0,00 - peinture en champs de couleur: 0,92 - cubisme: 92,26 - expressionnisme: 0,47

Émotion détectée : négative

Couleur dominante : brun

Style labellisé : expressionnisme

Style détecté : symbolisme

Probabilité par style : abstrait: 10,53 - impressionnisme: 7,59 - réalisme: 7,23 - romantisme: 8,24 - symbolisme: 29,31 - peinture en champs de couleur: 2,16 - cubisme: 11,36 - expressionnisme: 23,58

Émotion détectée : négative

Couleur dominante : vert

Style labellisé : expressionnisme

Style détecté : expressionnisme

Probabilité par style : abstrait: 0,63 - impressionnisme: 0,50 - réalisme: 0,31 - romantisme: 0,48 - symbolisme: 34,66 - peinture en champs de couleur: 0,00 - cubisme: 25,87 - expressionnisme: 37,54

Émotion détectée : négative

Couleur dominante : vert

Style labellisé : impressionnisme

Style détecté : impressionnisme

Probabilité par style : abstrait: 19,75 - impressionnisme: 58,34 - réalisme: 8,33 - romantisme: 4,24 - symbolisme: 5,34 - peinture en champs de couleur: 0,84 - cubisme: 1,14 - expressionnisme: 2,01

Émotion détectée : positive

Couleur dominante : gris

Style labellisé : impressionnisme

Style détecté : impressionnisme

Probabilité par style : abstrait: 0,38 - impressionnisme: 73,11 - réalisme: 11,13 - romantisme: 5,36 - symbolisme: 7,01 - peinture en champs de couleur: 0,00 - cubisme: 0,41 - expressionnisme: 2,60

Émotion détectée : positive

Couleur dominante : brun

Style labellisé : réalisme

Style détecté : impressionnisme

Probabilité par style : abstrait: 0,34 - impressionnisme: 32,21 - réalisme: 11,74 - romantisme: 11,28 - symbolisme: 16,77 - peinture en champs de couleur: 0,01 - cubisme: 1,87 - expressionnisme: 25,77

Émotion détectée : négative

Couleur dominante : brun

Style labellisé : réalisme

Style détecté : réalisme

Probabilité par style : abstrait: 0,02 - impressionnisme: 10,53 - réalisme: 54,99 - romantisme: 30,25 - symbolisme: 3,92 - peinture en champs de couleur: 0,00 - cubisme: 0,00 - expressionnisme: 0,28

Émotion détectée : positive

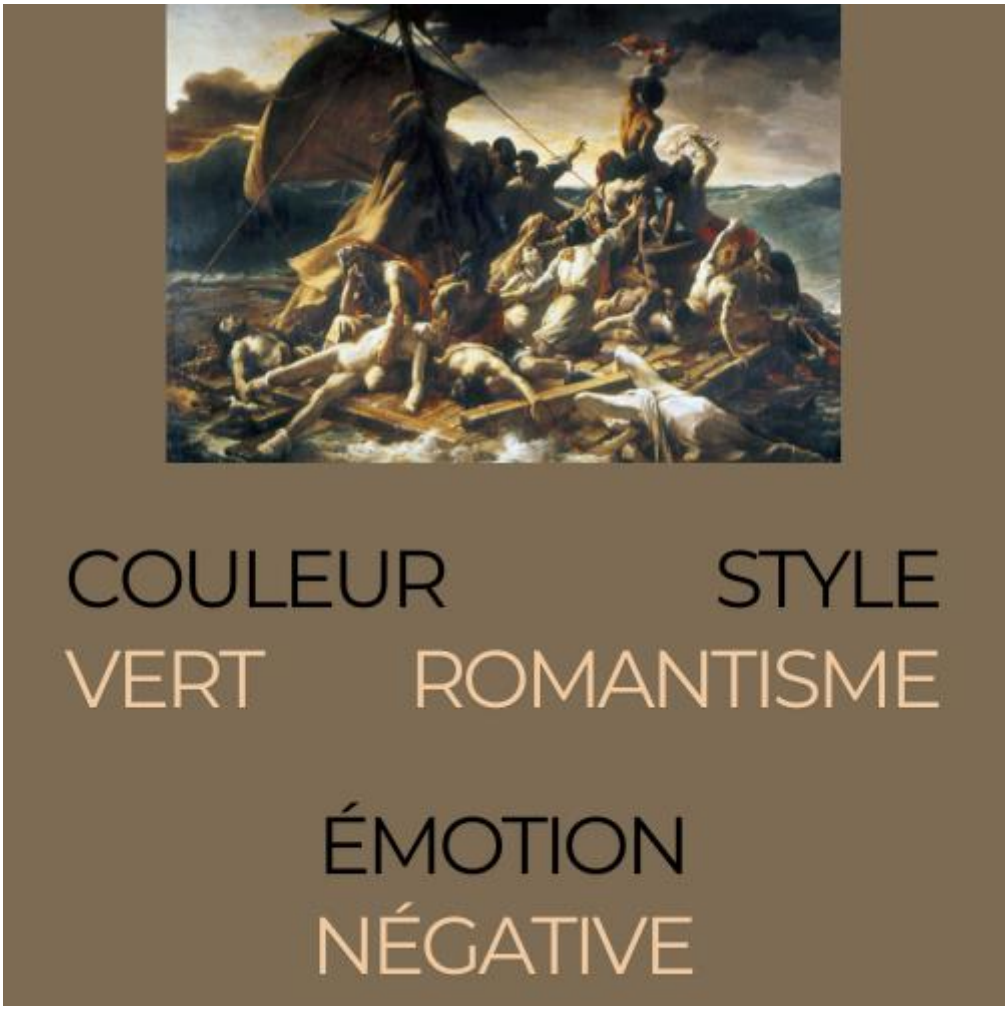
Couleur dominante : vert

Style labellisé : romantisme

Style détecté : réalisme

Probabilité par style : abstrait: 0,34 - impressionnisme: 13,33 - réalisme: 41,51 - romantisme: 30,37 - symbolisme: 12,47 - peinture en champs de couleur: 0,01 - cubisme: 0,09 - expressionnisme: 1,87

Émotion détectée : négative



Couleur dominante : vert

Style labellisé : romantisme

Style détecté : romantisme

Probabilité par style : abstrait: 0,76 - impressionnisme: 5,07 - réalisme: 21,68 - romantisme: 38,02 - symbolisme: 24,10 - peinture en champs de couleur: 0,01 - cubisme: 0,54 - expressionnisme: 9,81

Émotion détectée : négative

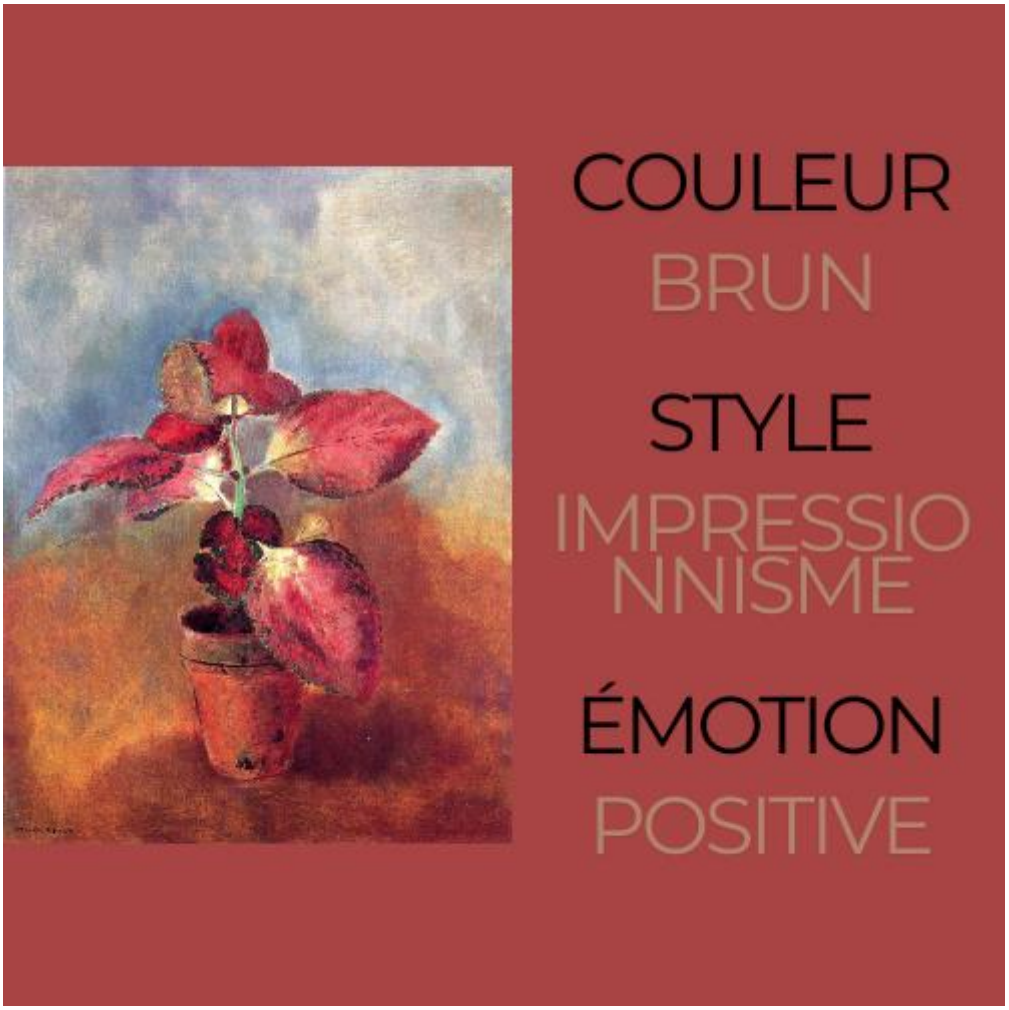
Couleur dominante : gris

Style labellisé : symbolisme

Style détecté : symbolisme

Probabilité par style : abstrait: 8,87 - impressionnisme: 3,31 - réalisme: 1,82 - romantisme: 1,83 - symbolisme: 66,83 - peinture en champs de couleur: 0,76 - cubisme: 3,80 - expressionnisme: 12,79

Émotion détectée : négative



Couleur dominante : brun

Style labellisé : symbolisme

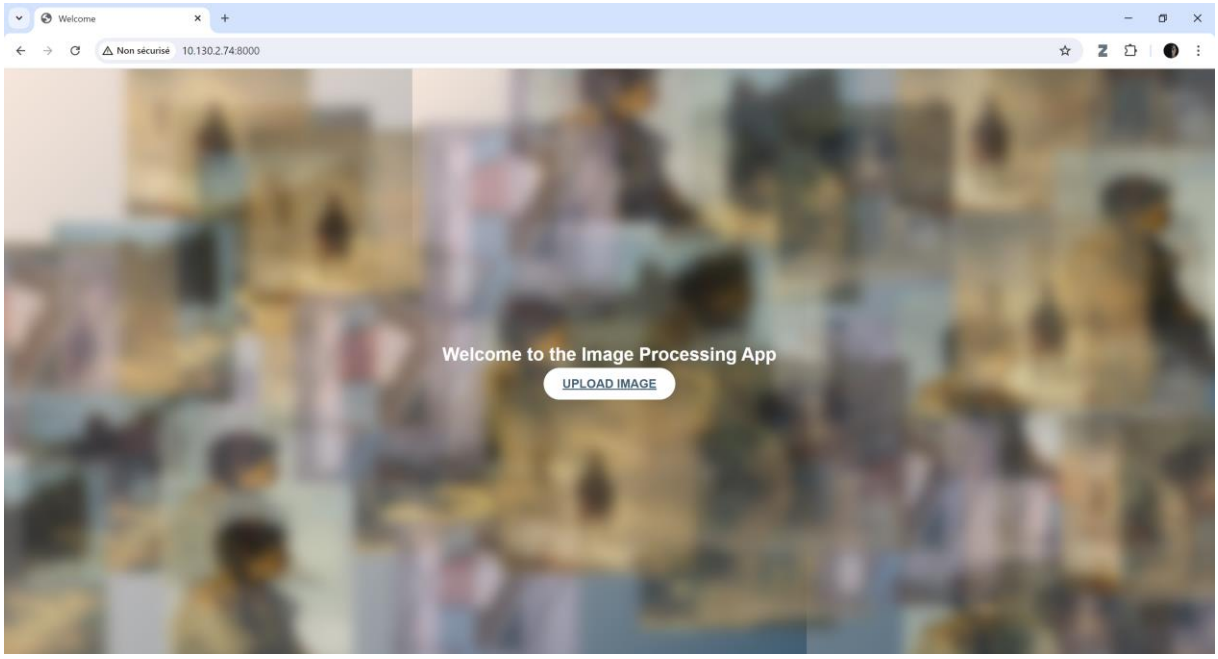
Style détecté : impressionnisme

Probabilité par style : abstrait: 0,46 - impressionnisme: 43,58 - réalisme: 16,43 - romantisme: 12,16 - symbolisme: 18,93 - peinture en champs de couleur: 0,02 - cubisme: 0,62 - expressionnisme: 7,80

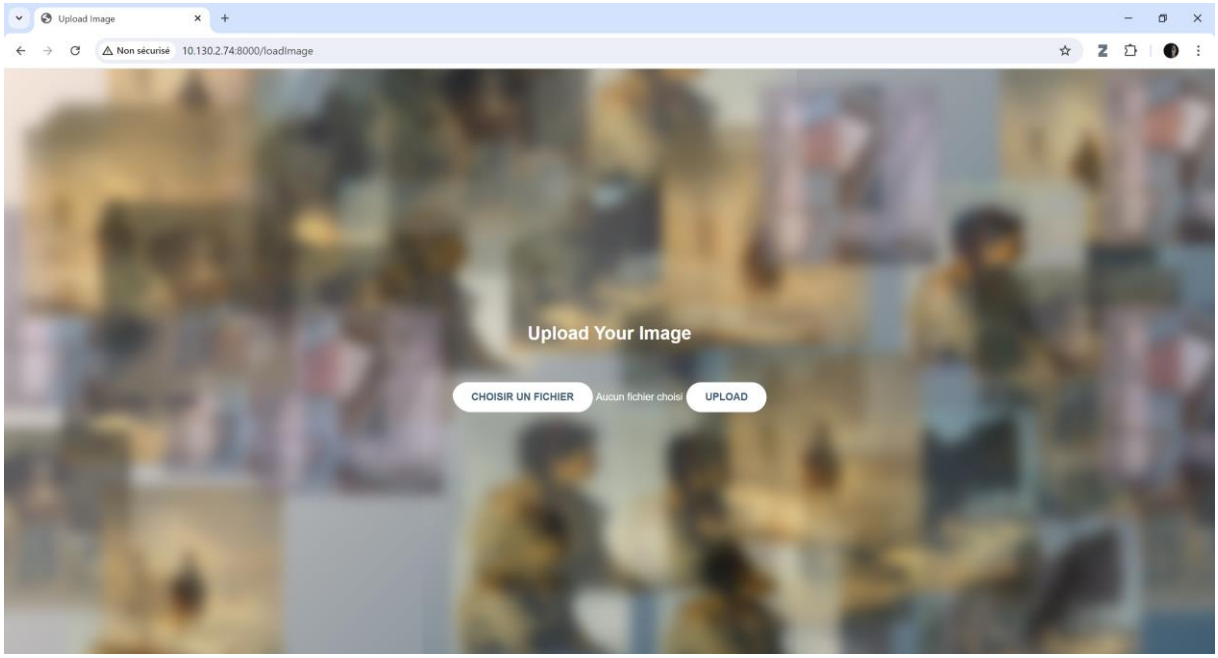
Émotion détectée : positive

10.5. Interface utilisateur de la web app

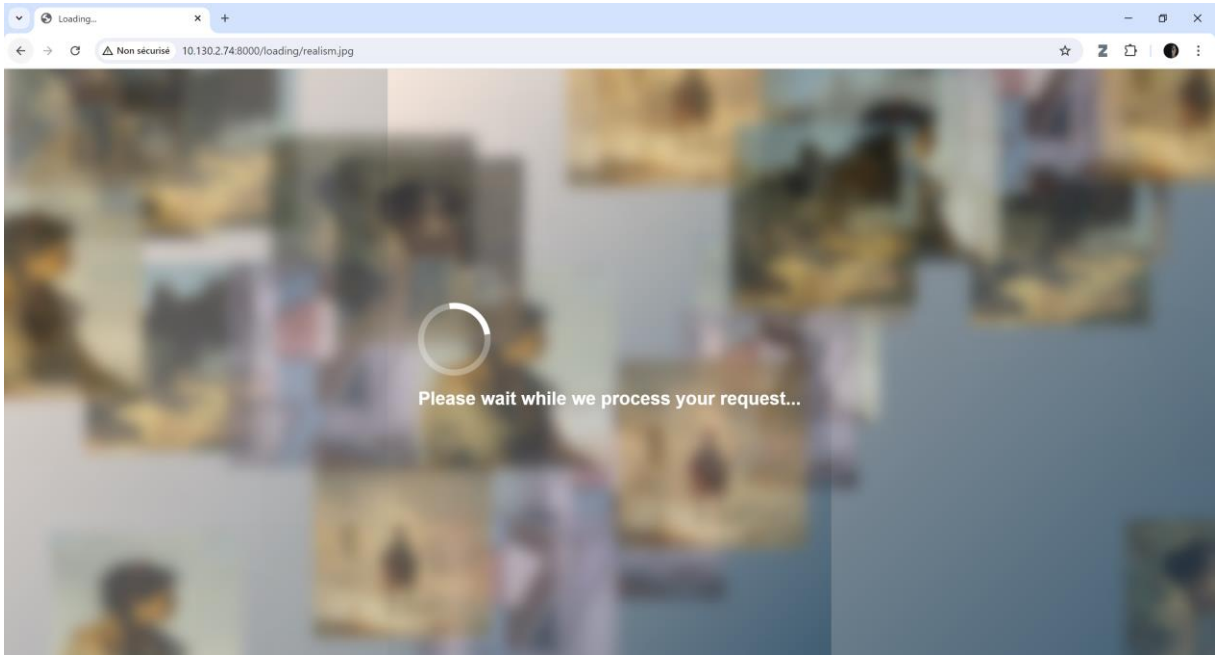
Page d'accueil



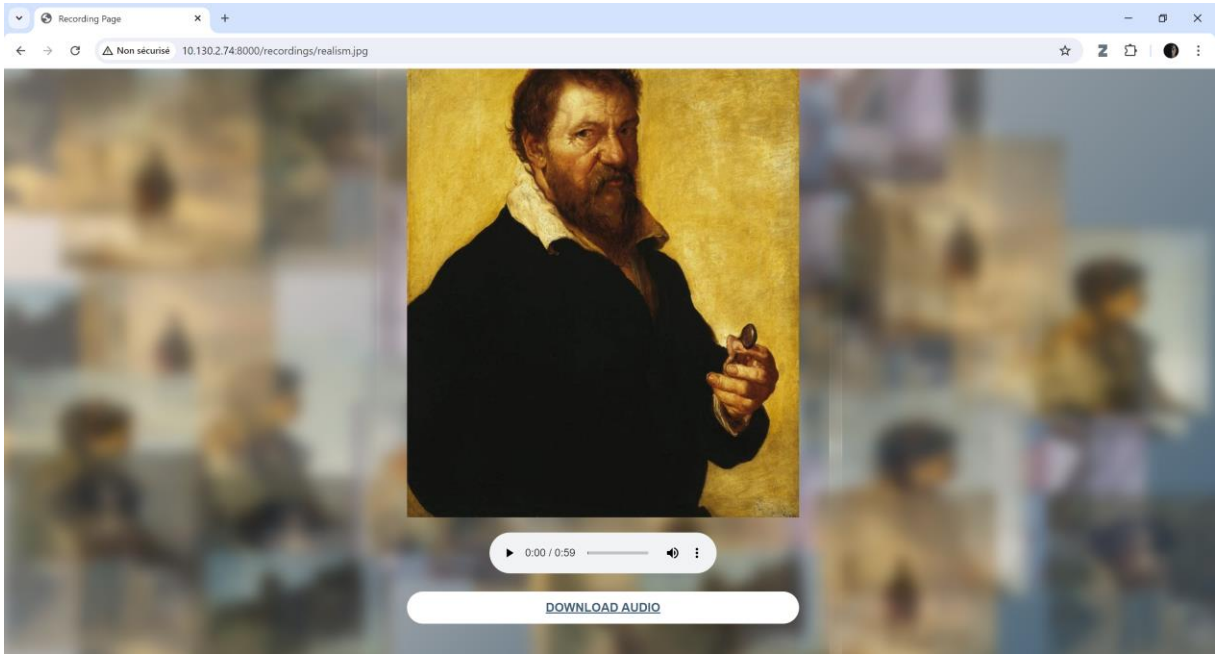
Page de dépôt de l'œuvre à analyser



Page de chargement



Page de résultat



10.6. Datasets utilisés

Les jeux de données suivants ont été manipulés pour créer ceux utilisés dans ce travail.

MART : https://staff.fnwi.uva.nl/e.bruni/mart/dlform.html

ART500K : https://deepart.hkust.edu.hk/ART500K/art500k.html

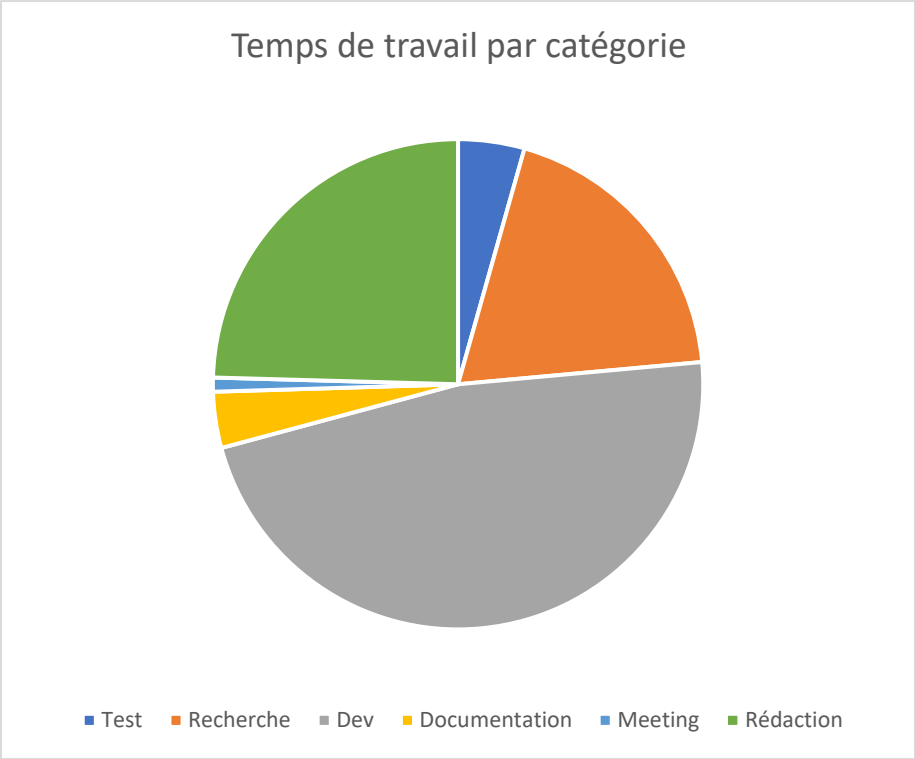
WikiArt : https://www.kaggle.com/datasets/steubk/wikiart

10.7. Product Backlog

US Nbr	Thème	Titre US	Description	Critère d'acceptation	Priorité	Status	Story Point	Sprint	Moscow
1	Documentation	Réalisation du PB	Réaliser le product backlog du projet	PB complet et validé	1000		3	1	Must
3	Recherche	État de l'art sur l'analyse d'émotion d'une image	Examiner les technologies et méthodes actuelles en analyse d'émotion	Document détaillé avec bibliographie	900		13	1	Must
5	Conception	Choix des caractéristiques de l'œuvre visuelle	Identifier les caractéristiques visuelles pertinentes pour la sonification	Liste des caractéristiques visuelles avec descriptions	880		1	1	Must
9	Choix des outils	Choix de l'outil pour l'analyse d'émotion	Comparer et choisir la méthode d'extraction de l'émotion d'une œuvre	Documentation de l'outil choisi et les raisons du choix	860		5	1	Must
10	Choix des outils	Choix de l'outil pour l'analyse de formes	Comparer et choisir la méthode d'extraction des formes d'une œuvre	Documentation de l'outil choisi et les raisons du choix	840		5	1	Must
11	Choix des outils	Choix de l'outil pour l'analyse des couleurs	Comparer et choisir la méthode d'extraction des couleurs d'une œuvre	Documentation de l'outil choisi et les raisons du choix	820		5	1	Must
11	Choix des outils	Choix de l'outil pour l'analyse des couleurs	Comparer et choisir la méthode d'extraction des couleurs d'une œuvre	Documentation de l'outil choisi et les raisons du choix	818		5	2	Must
24	Choix des outils	Choix de l'outil pour l'analyse de textures	Comparer et choisir la méthode d'extraction des textures d'une œuvre	Documentation de l'outil choisi et les raisons du choix	815		5	2	Must
23	Choix des outils	Choix de l'outil d'analyse du style	Comparer et choisir la méthode d'extraction du style de l'oeuvre	Documentation de l'outil choisi et les raisons du choix	812		5	2	Must
13	Données	Création les datasets d'images	Créer les différents datasets d'images nécessaires à l'entraînement des modèles d'analyse d'image	Datasets complets pour l'analyse d'émotion, de formes et de couleurs	810		8	2	Must
2	Recherche	État de l'art sur la sonification	Analyser les techniques existantes et théories sur la sonification d'images	Document détaillé avec bibliographie	800		5	3	Must
4	Recherche	État de l'art sur la synthèse sonore	Etudier les approches et techniques existantes	Document détaillé avec bibliographie	780		5	3	Must
6	Conception	Choix des caractéristiques de l'œuvre sonore	Identifier les caractéristiques sonores pertinentes pour la sonification	Liste des caractéristiques sonores avec descriptions	760		3	3	Must
7	Conception	Définition du lexique de sonification	Définir les correspondances entre les caractéristiques visuelles et sonores	Lexique complet comprenant le mapping des caractéristiques visuelles et sonores	740		3	3	Must
12	Choix des outils	Choix de l'outil de synthèse sonore	Identifier les outils adéquats pour la création sonore dans le cadre du projet	Documentation de l'outil choisi et les raisons du choix	720		5	3	Must
8	Conception	Définition de la structure du modèle de synthèse sonore	Identifier les besoins nécessaires à la synthèse sonore et créer la structure du modèle	Documentation de la structure du modèle	700		5	3	Must
17	Développement	Développement du modèle de synthèse sonore	Développer le modèle de synthèse sonore basé sur les caractéristiques extraites	Modèle de synthèse sonore générant du son basé sur des données entrantes	660		13	4	Must
18	Intégration	Création de la pipeline	Créer une pipeline qui intègre le processus complet de l'analyse d'image à la synthèse sonore	Processus entièrement fonctionnel sur la VM			13	4	Must
15	Développement	Développement du modèle d'analyse de textures	Développer et tester le module d'analyse de textures basé sur les outils choisis	Modèle capable d'extraire les caractéristiques des textures qui composent une image	650		5	4	Must
16	Développement	Développement du modèle d'extraction de couleurs	Développer et tester le module d'analyse de couleurs basé sur les outils choisis	Modèle capable d'extraire les couleurs principales d'une image	640		5	4	Must
14	Développement	Développement du modèle d'extraction d'émotion	Développer et tester le module d'analyse d'émotion basé sur les outils choisis	Modèle capable d'extraire l'émotion d'une image	630		13	5	Must
25	Développement	Développement du modèle d'analyse du style	Développer et tester le module d'analyse du style basé sur les outils choisis	Modèle capable de prédire le style artistique d'une oeuvre	620		8	5	Must
19	Interface	Développement du backend de la webapp	Développer le backend de la web application pour gérer les requêtes et traitements	Backend fonctionnel	550		8	5	Must
20	Interface	Développement de l'interface utilisateur	Développer l'interface utilisateur pour l'interaction avec le système	Interface utilisateur fonctionnelle et intuitive	300		5	5	Must
21	Analyse	Analyse des résultats des processus	Evaluation les résultats des modèles d'analyse et de synthèse sonore	Tous les modèles sont évalués et documentés dans le rapport	200		13	6	Must
22	Documentation	Finalisation du rapport	Finaliser la rédation du rapport détaillé du projet	Rapport complet, structuré et prêt pour soumission	100		21	6	Must
					50				

10.8. Répartition des heures

Test	13,00
Recherche	57,00
Dev	140,50
Documentation	11,00
Meeting	2,75
Rédaction	73,00
	297,25



10.9. Tâches des sprints

SPRINT 1

1	Réalisation du PB
3	État de l'art sur l'analyse d'images
5	Choix des caractéristiques de l'œuvre visuelle
9	Choix de l'outil pour l'analyse d'émotion
10	Choix de l'outil pour l'analyse de formes
11	Choix de l'outil pour l'analyse des couleurs

Start date 23/05/2024
End date 06/06/2024
Initial Sprint Estimation (SP) 32
Nb working days 6

ID US.task	Task Name	Initial Estimate	23.05 Day 1	24.05 Day 2	29.05 Day 3	30.05 Day 4	05.06 Day 5	06.06 Day 6
1.1	Estimation des étapes du projet et réalisation du product backlog	4,5	1,5	0	0	0	0	0
3.1	Recherche d'articles et ouvrages pertinents	12	12	8	7	6	4	0
3.2	Synthèses des approches trouvées dans la littérature et rédaction	6	6	5	4	3	2	0
5.1	Recherche de travaux de sonification d'images	1	1	1	0	0	0	0
5.2	Synthèse des recherches et choix des caractéristiques	0,5	0,5	0,5	0	0	0	0
9..1	Analyse et comparaison des performances et applications des outils d'analyse d'émotion	4	4	4	1	0	0	0
9.2	Synthèse des résultat et choix de l'outil d'analyse d'émotion	4	4	4	3	0	0	0
10.1	Analyse et comparaison des performances et applications des outils d'analyse de forme	4	4	4	4	4	4	4
10.2	Synthèse des résultat et choix de l'outil d'analyse de forme	4	4	4	4	4	4	4
11.1	Analyse et comparaison des performances et applications des outils d'analyse de couleur	4	4	4	2	2	2	1
11.2	Synthèse des résultat et choix de l'outil d'analyse de couleur	4	4	4	4	4	4	2
	Total	*48,0*						
	Remaining	*48,0*	*45,00*	*38,50*	*29,00*	*23,00*	*20,00*	*11,00*
	Therical	*48,0*	*48,00*	*38,40*	*28,80*	*19,20*	*9,60*	*0,00*

SPRINT 2

24	Choix de l'outil pour l'analyse de textures
11	Choix de l'outil pour l'analyse des couleurs
23	Choix de l'outil d'analyse du style
13	Création les datasets d'images

Start date	07/06/2024
End date	21/06/2024
Initial Sprint Estimation (SP)	23
Nb working days	6

			07.06	08.06	13.06	14.06	20.06	21.06
ID US.task	Task Name	Initial Estimate	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6
24.1	Analyse et comparaison des performances et applications des outils d'analyse de textures	4	4	4	4	4	0	0
24.2	Synthèse des résultat et choix de l'outil d'analyse de textures	4	4	4	4	4	1	0
11.1	Analyse et comparaison des performances et applications des outils d'analyse de couleur	4	4	4	2	3	0	0
11.2	Synthèse des résultat et choix de l'outil d'analyse de couleur	4	4	4	1	1	1	0
23.1	Analyse et comparaison des performances et applications des outils d'analyse de style	5	5	5	5	5	2	0
23.2	Synthèse des résultat et choix de l'outil d'analyse de style	4	4	4	4	4	4	0
13.1	Création du dataset final pour l'analyse d'émoti	3	3	3	0	0	0	0
13.2	Création du dataset final pour le nommage des couleurs	2	2	2	0	0	0	0
13.3	Création du dataset final pour l'analyse du style	3	3	3	3	3	0	0
	Total	*33,0*						
	Remaining	*33,0*	*33*	*33,00*	*23,00*	*24,00*	*8,00*	*0,00*
	Therical	*33,0*	*33*	*26,40*	*19,80*	*13,20*	*6,60*	*0,00*

SPRINT 3

2	État de l'art sur la sonification
4	État de l'art sur la synthèse sonore
6	Choix des caractéristiques de l'œuvre sonore
7	Définition du lexique de sonification
12	Choix de l'outil de synthèse sonore

Start date	22/06/2024
End date	07/07/2024
Initial Sprint Estimation (SP)	26
Nb working days	6

			22.06	27.06	29.06	03.07	04.07	06.07
ID US.task	Task Name	Initial Estimate	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6
2.1	Recherche d'articles et ouvrages pertinents	4	1	0	0	0	0	0
2.2	Synthèses des approches trouvées dans la littérature et rédaction	3	3	0	0	0	0	
4.1	Recherche d'articles et ouvrages pertinents	4	0	0	0	0	0	0
4.2	Synthèses des approches trouvées dans la littérature et rédaction	3	3	0	0	0	0	0
6.1	Définition des caractéristiques sonores utilisées	2	2	2	2	2	2	0
7.1	Définition préics du mapping entre les caractéristiques visuelles et sonores	3	3	3	0	0	0	0
12.1	Analyse et comparaison des performances et applications des outils de synthèse sonore	4	4	3	0	0	0	0
12.2	Synthèse des résultat et choix de l'outil de synthèse sonore	4	4	4	4	4	2	0
	Total	*27,0*						
	Remaining	*27,0*	*20*	*12,00*	*6,00*	*6,00*	*4,00*	*0,00*
	Therical	*27,0*	*27*	*21,60*	*16,20*	*10,80*	*5,40*	*0,00*

SPRINT 4

17	Développement du modèle de synthèse sonore
18	Création de la pipeline
15	Développement du modèle d'analyse de textures
16	Développement du modèle d'extraction de couleurs

Start date	08/07/2024
End date	21/07/2024
Initial Sprint Estimation (SP)	36
Nb working days	6

			11.07	12.07	13.07	18.07	20.07	21.07
ID US.task	Task Name	Initial Estimate	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6
17.1	Développement de la structure du code de synthèse sonore	10	10	4	4	0	0	0
17.2	Test du code en 'situation réelle'	2	2	2	0	0	0	0
18.1	Installation des frameworks et software sur le serveur	4	4	4	4	0	0	0
18.2	Implémentation du code d'analyse d'image et de synthèse sonore sur le serveur	6	5	5	5	0	0	0
18.3	Automatisation des processus d'analyse d'image et de synthèse sonore	4	4	4	4	1	2	0
15.1	Développement du modèle final d'analyse de texture	4	3	0	0	0	0	0
15.2	Test du modèle d'analyse de texture	2	3	0	0	0	0	0
16.1	Développement du modèle final d'extraction de la couleur dominante	4	4	4	0	0	0	0
16.2	Test du modèle d'extraction de la couleur dominante	2	2	2	0	0	0	0
	Total	*38,0*						
	Remaining	*38,0*	*37*	*25,00*	*17,00*	*1,00*	*2,00*	*0,00*
	Therical	*38,0*	*38*	*30,40*	*22,80*	*15,20*	*7,60*	*0,00*

SPRINT 5

14	Développement du modèle d'extraction d'émotion
19	Développement du backend de la webapp
20	Développement de l'interface utilisateur
25	Développement du modèle d'analyse du style

Start date	22/07/2024
End date	04/08/2024
Initial Sprint Estimation (SP)	34
Nb working days	6

			25.07	26.07	27.07	28.07	01.08	04.08
ID US.task	Task Name	Initial Estimate	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6
14.1	Développement du modèle final d'extraction d'émotion	14	12	11	8	0	0	0
14.2	Test du modèle d'analyse d'émotion	2	2	2	2	2	0	0
19.1	Développement du backend de l'application	6	6	6	6	6	0	0
19.2	Test de la web app	2	2	2	2	2	2	0
20.1	Développement du frontend de l'application	6	6	6	6	6	6	0
20.2	Test de la web app	2	2	2	2	2	2	0
25.1	Développement du modèle final d'analyse du style	10	10	1	2	0	0	0
25.2	Test du modèle d'analyse du style	2	2	2	2	0	0	0
	Total	*44,0*						
	Remaining	*44,0*	*42*	*32,00*	*30,00*	*18,00*	*10,00*	*0,00*
	Therical	*44,0*	*44*	*35,20*	*26,40*	*17,60*	*8,80*	*0,00*

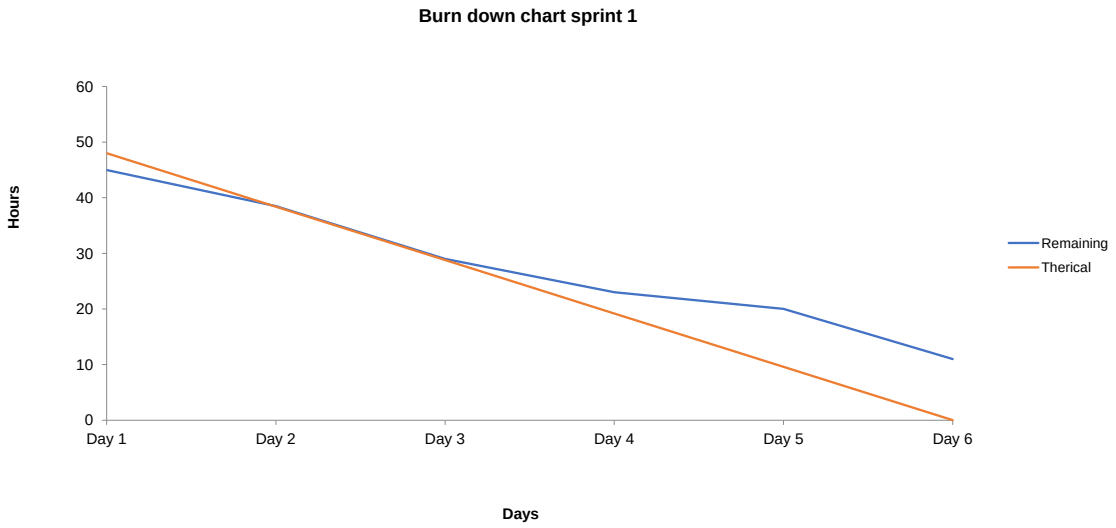
SPRINT 6

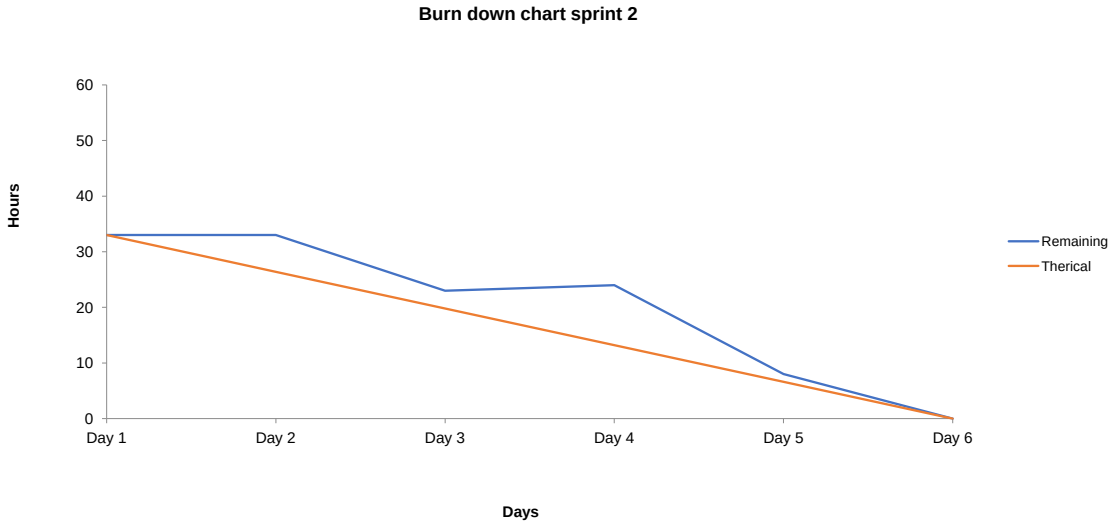
21	Analyse des résultats des processus
22	Finalisation du rapport

Start date	05/08/2024
End date	15/08/2024
Initial Sprint Estimation (SP)	21
Nb working days	6

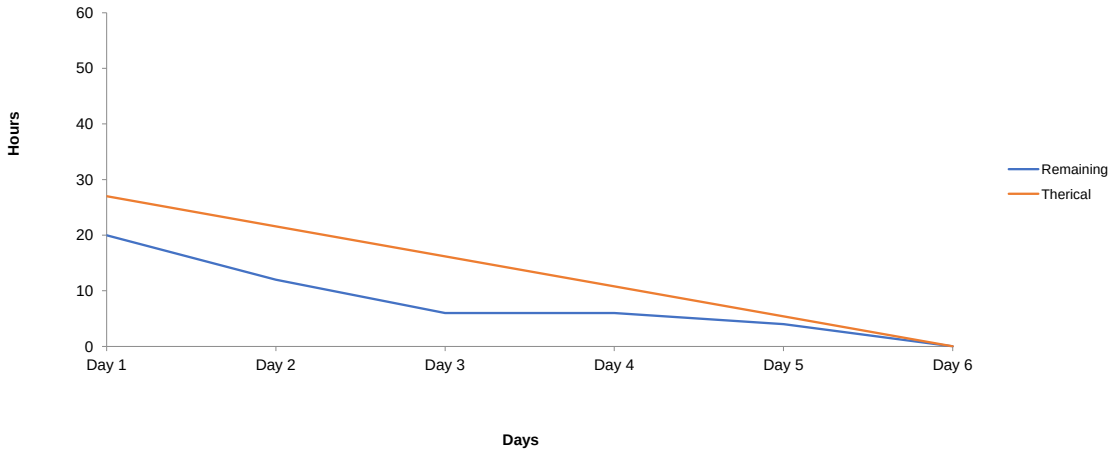
			08.08	09.08	10.08	11.08	14.08	15.08
ID US.task	Task Name	Initial Estimate	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6
21.1	Analyse des résultats de l'analyse de texture	3	3	3	3	0	0	0
21.2	Analyse des résultats de l'analyse de couleur	3	3	3	0	0	0	0
21.3	Analyse des résultats de l'analyse d'émotion	3	3	0	0	0	0	0
21.4	Analyse des résultats de l'analyse de style	3	3	2	3	0	0	0
21.5	Analyse des résultats de la synthèse sonore	3	3	3	3	3	3	0
22.1	Finalisation de la rédaction du rapport	25	25	20	15	9	4	0
	Total	*40,0*						
	Remaining	*40,0*	*40*	*31,00*	*24,00*	*12,00*	*7,00*	*0,00*
	Therical	*40,0*	*40*	*32,00*	*24,00*	*16,00*	*8,00*	*0,00*

10.10. Burn down charts

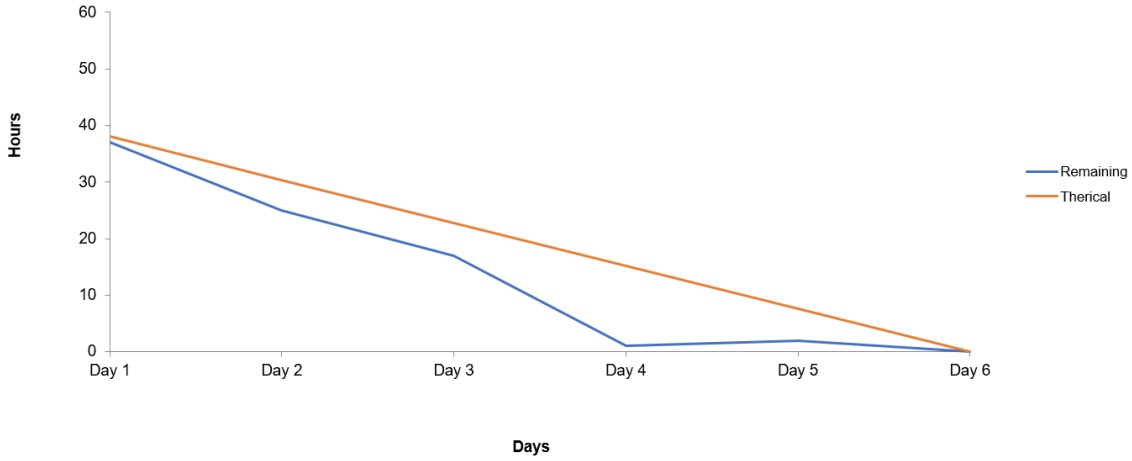


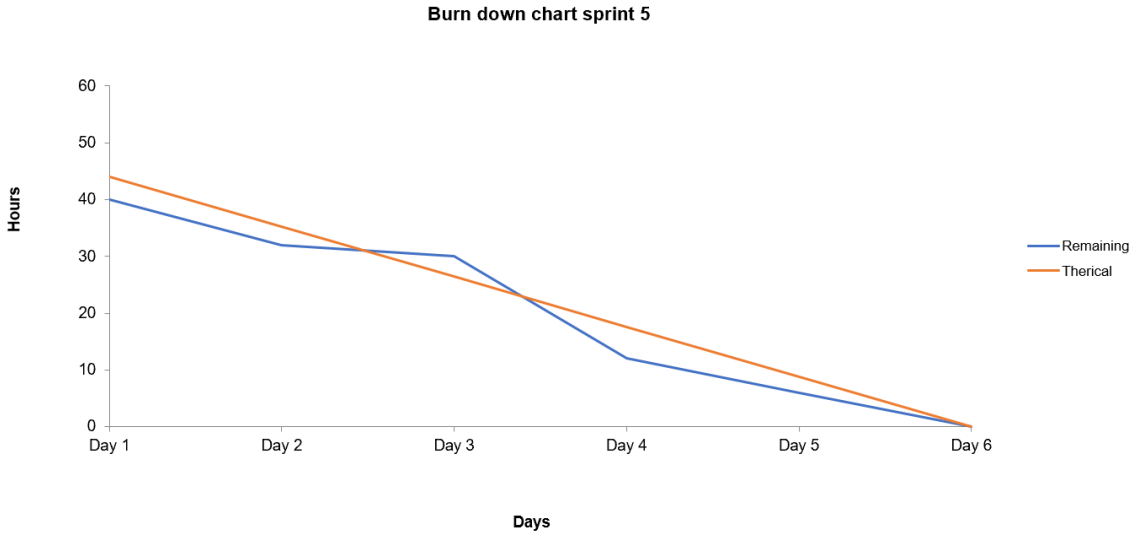


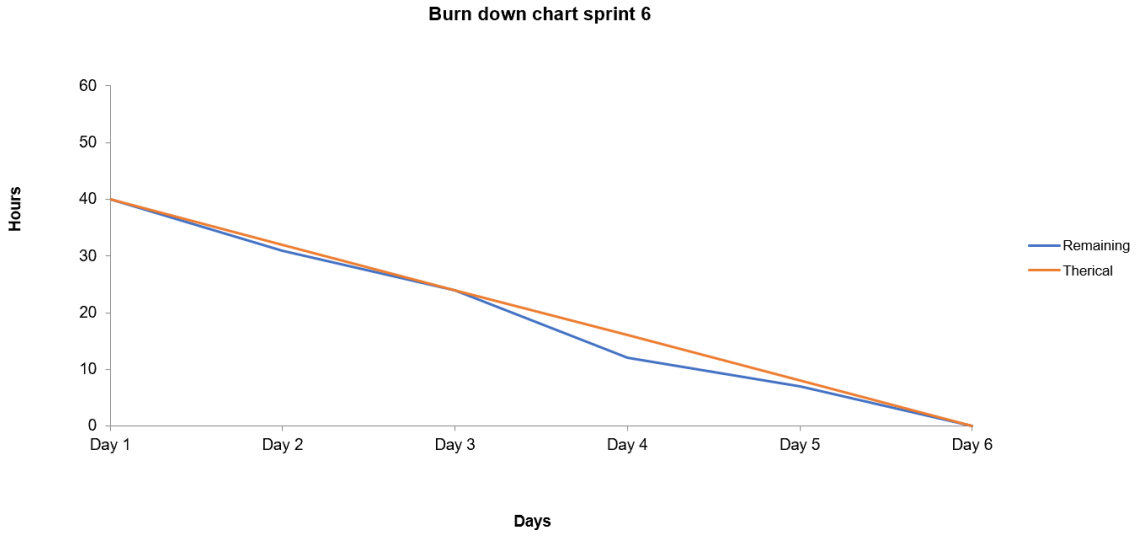
Burn down chart sprint 3



Burn down chart sprint 4




Burn down chart sprint 5
Hours
60
50
40
30
20
10
0
Day 1
Day 2
Day 3
Day 4
Day 5
Day 6
Days
Remaining
Therical


Burn down chart sprint 6
Hours
60
50
40
30
20
10
0
Day 1
Day 2
Day 3
Day 4
Day 5
Day 6
Days
Remaining
Therical

10.11. Données du travail de Bachelor

HES-SO Valais

FEE	FIG	FTO
	X	

Données du travail de bachelor
Daten der Bachelorarbeit

FO.2.2.02.28.EC
Version 2022

Filière / *Studiengang*: INFORMATIQUE DE GESTION / WIRTSCHAFTSINFORMATIK

Confidentiel / *Vertraulich* ☐

Etudiant-e / *Student/in*		**Année / *Jahr***
NOM / NAME: Prénom / Vorname: **Portable / Mobiltelefon:**	REY Guillaume 079 944 67 94	2024
Proposé par / *vorgeschlagen von:* Mme Valérie Félix : valerie.felix@hevs.ch Responsable de la Filière HEA Responsable pédagogique du Bachelor en Arts visuels **Professeur / *Dozent/in*:**	**Henning MÜLLER**	**Langue d'exécution / *Ausführungssprache*** ☒ Français ☐ Deutsch ☐ English

Titre / *Titel*: "L'Art en Résonance : Convergence du Visuel et de l'Audible par l'Intelligence Artificielle"

Description / *Beschreibung*:

Contexte (entreprise, projet de recherche, etc.) / *Hintergrund (Unternehmen, Forschungsprojekt usw.)*

L'idée de ce projet est née d'un intérêt profond pour les domaines croisés de l'intelligence artificielle (IA) et de la sonification. L'essor récent de l'IA et son potentiel illimité dans divers secteurs sont fascinants.

Pour donner vie à ce projet, des contacts ont été établis avec l'École de Design et Haute École d'Art du Valais (EDHEA), qui a exprimé un vif intérêt à participer à ce projet en tant que partenaire métier. Leur contribution se matérialisera à travers leur expérience dans le monde de l'art, ainsi que par la potentielle mise à disposition de données artistiques, telles que des œuvres d'art visuelles. Des contacts sont également établis avec le musée d'art du Valais pour la mise à disposition d'un catalogue d'œuvres visuelles numérisées. Ces données seront essentielles pour développer et affiner les algorithmes de sonification basés sur l'IA. Si ni l'EDHEA, ni le musée ne peuvent mettre un catalogue d'œuvres à disposition, un plan de secours sera de récupérer des œuvres en ligne.

Au-delà de ce travail de bachelor, l'EDHEA envisage de poursuivre le développement de ce projet, y voyant un potentiel considérable dans des cadres tels que des projets de recherche et des travaux d'étudiants futurs. Ce projet pourrait ouvrir des voies nouvelles pour des collaborations interdisciplinaires et servir de plateforme pour explorer davantage les interactions entre technologie, son, et art visuel.

A quelle problématique concrète le projet répond-il ? / *Mit welcher konkreten Problematik befasst sich das Projekt?*

La problématique centrale de ce projet interroge les possibilités d'utilisation des techniques de data mining et de deep learning pour transformer des œuvres d'art visuelles en expériences sonores. L'objectif est de créer une nouvelle forme d'appréciation artistique, accessible notamment aux personnes malvoyantes. Cette question guide l'exploration des moyens par lesquels la technologie peut transcender les frontières entre les sens et rendre l'art plus inclusif et accessible.

L'ambition est de développer un système employant des techniques de deep learning pour convertir des œuvres d'art visuelles - telles que des photographies ou des peintures - en compositions sonores. Ce processus ambitionne d'offrir une nouvelle dimension d'expérience artistique, particulièrement pour les personnes malvoyantes, pour lesquelles l'accès à l'art visuel est limité.

Pour réaliser cet objectif, le projet envisage de :

- Analyser les œuvres visuelles : Utiliser des algorithmes de vision par ordinateur pour analyser les caractéristiques détaillées des œuvres d'art, telles que les couleurs, les textures, la saturation, la disposition spatiale et les formes.
- Modéliser par Deep Learning : Développer et entraîner des modèles de deep learning capables de comprendre et d'interpréter ces caractéristiques visuelles. Ces modèles apprendront à établir des correspondances entre les aspects visuels des œuvres d'art et des paramètres sonores spécifiques.
- Sonifier : Concevoir un système de sonification qui interprète les données visuelles extraites en sons, en impliquant l'utilisation de logiciels de génération sonore et d'autres outils de traitement audio.

HES-SO Valais

FEE	FIG	FTO
	X	

Données du travail de bachelor
Daten der Bachelorarbeit

FO.2.2.02.28.EC
Version 2022

Etapes de réalisation envisagées / Vorgehen im Hinblick auf die Durchführung

Analyse ou état de l'art / Analyse oder "State of the Art"

L'une des premières étapes du projet consistera en la recherche d'outils open source existants dans les domaines de l'analyse d'image et de la génération sonore. L'objectif de cette recherche est d'identifier des technologies disponibles qui pourraient servir de base au projet. Des outils d'analyse d'image avancés seront utilisés pour décomposer les œuvres d'art en caractéristiques exploitables, tandis que des technologies de génération sonore offriront les moyens de transformer ces caractéristiques en expériences auditives. Il sera également crucial d'étudier les travaux d'artistes et de chercheurs qui ont déjà exploré le domaine de la conversion de l'image en son. Des figures telles que Neil Harbisson, connu pour son « œil électronique » qui transforme les couleurs en sons, et Vassily Kandinsky, dont les théories sur la synesthésie et l'art abstrait sont pertinentes, seront examinées.

Choix avec prise de position de l'étudiant / Entscheid und Begründung des/der Studierenden

Divers choix devront être effectués parmis les éléments suivants (*les personnes responsables de la validation de ces choix sont mentionnés entre parenthèse*) :

Data mining :

- Collecte de données : Il faudra déterminer quelles données (mouvement artistique, couleurs, formes, textures, etc.) seront collectées, en quelle quantité et sous quelle forme. (*EDHEA*)
- Analyse exploratoire : Identifier les caractéristiques les plus pertinentes d'un panel d'œuvres d'art et définir comment elles peuvent être utilisées efficacement dans le processus de sonification. (*EDHEA*)
- Prétraitement des données : Normalisation, transformation, et réduction de dimensionnalité des données pour une utilisation optimale dans les modèles de deep learning. (*Professeur responsable du suivi*)

Deep learning :

- Association de caractéristiques sonores et visuelles : Définir comment mapper les caractéristiques visuelles sur des paramètres sonores. (*Professeur responsable du suivi*)
- Modèle de détection : Choisir l'outil le plus approprié pour l'analyse d'images. (*Professeur responsable du suivi*)
- Génération sonore : Sélectionner l'outil idéal pour la génération de séquences sonores cohérentes. (*L'étudiant par ses connaissances liées à sa formation précédente*)

Autres technologies (*Professeur responsable du suivi*) :

- Stockage des données : Déterminer le format et le système de stockage des données
- Pipeline : Choisir le langage de programmation et les modes de communication entre les différents outils utilisés

Implémentation & Testing / Umsetzung & Tests

Une étape essentielle de l'implémentation consistera en la création d'un product backlog. Ce document comprendra toutes les fonctionnalités, tâches, et exigences nécessaires pour mener le projet à bien. Le backlog servira de guide pour le développement et garantira que toutes les composantes clés du projet soient prises en compte. La création de ce document se fera en préparation du travail de bachelor et sera soumise à l'approbation du professeur en début de processus. Chaque phase du processus sera soumise à des tests rigoureux afin de vérifier la performance du système et son adéquation avec les objectifs fixés. Ce projet est un « proof of concept » qui pourra être poursuivi après le travail de bachelor, et prendra la forme d'une application web sur laquelle des œuvres d'art destinées à la sonification pourront être déposées.

HES-SO Valais

FEE	FIG	FTO
	X	

Données du travail de bachelor
Daten der Bachelorarbeit

FO.2.2.02.28.EC
Version 2022

Ressources à disposition / *Verfügbare Ressourcen*

Datas / *Daten*

(Préciser quelles sont les données, quelles sont les métriques utilisées, sous quelle forme sont-elles disponibles / *Geben Sie den Datentyp und das Datenformat sowie die verwendeten Metriken an*)

Données d'entrée : Images

Format des Images : Les images servant de données d'entrée comprendront principalement des photographies numériques ou des scans de haute qualité d'œuvres d'art (peintures, illustrations, etc.), au format JPEG, PNG, ou TIFF, afin d'assurer une restitution de bonne qualité.

Conversion en Données Analytiques : Une analyse de chaque image sera effectuée pour en extraire des informations quantitatives et qualitatives, telles que :

Couleur : Détails sur les gammes de couleurs, leur intensité et leur distribution.

Forme : Identification des formes principales et de leur disposition.

Saturation : Mesure de l'intensité des couleurs.

Position : Localisation spatiale des différents éléments dans l'image.

Texture : Analyse des motifs de surface, tels que le lissé, le rugueux ou le pointillé.

Mapping des Données Visuelles et Correspondances Sonores

Processus de Mapping : Mise en place d'un système de correspondance par lequel chaque caractéristique visuelle (couleur, forme, texture, etc.) est reliée à un paramètre sonore spécifique, basé sur des règles prédéfinies et affiné par des tests et retours.

Métriques Utilisées : Les métriques pour le mapping incluront, par exemple, la fréquence pour les couleurs, le rythme pour les formes, la dynamique pour la saturation.

Synthèse Sonore

Outils Utilisés : Utilisation de logiciels de génération sonore pour la sonification de l'image.

Données de sortie : Le résultat sera une piste audio au format WAV ou MP3.

Use case

(Décrire ici le use case utilisé / *Beschreiben Sie den verwendeten Use Case*)

Un utilisateur utilise un web service pour sélectionner une œuvre d'art visuelle. Le système analyse l'image, extrayant des caractéristiques telles que la couleur et la forme. Ces données sont ensuite converties en sons correspondants, créant une expérience sonore qui reflète l'œuvre d'art visuelle.

Hardware & ressources / *Hardware & Ressourcen*

(Préciser quelles sont les ressources qui seront mises à disposition ainsi que leur environnement / *Geben Sie die zur Verfügung gestellten Ressourcen sowie deren Umgebung an*)

Il n'y aura pas de ressources spécifiques mises à disposition.

Ce projet étant un proof of concept, l'échelle de données et les exigences de calcul pourraient être adaptées aux capacités de calcul d'un ordinateur portable et nécessiterait donc pas de hardware spécifique.

Les logiciels et le hardware nécessaires au traitement de son sont fournis par l'étudiant.

Signatures ou visa / *Unterschriften oder Visum*

Responsable de la filière Informatique de gestion / Leiter des Studiengangs Wirtschaftsinformatik:

.....

Professeur / Dozent/in: .....

Etudiant / Student/in: .....

Délais / *Termine*

Début du TB / Start der BA:
20.05.2024

Dépôt du TB / Abgabe der BA:
16.08.2024 à 12h00

Exposition publique – Silicon Valais / Öffentliche Ausstellung – Silicon Valais:
Date à définir

10.12. Projet GitHub

https://github.com/guillaumrey2/convergence-du-visuel-et-de-l-audible-par-l-intelligence-artificielle

DÉCLARATION DE L'AUTEUR

Je déclare, par ce document, que j'ai effectué le travail de Bachelor ci-annexé seul, sans autre aide que celles dûment signalées dans les références, et que je n'ai utilisé que les sources expressément mentionnées.

Icogne, le 15 août 2024 **Guillaume Rey**