Un rapport de soutenance

QUBE

Clément Castiglione

Alexis Busson

Jacques Dai

Mathys Abonnel

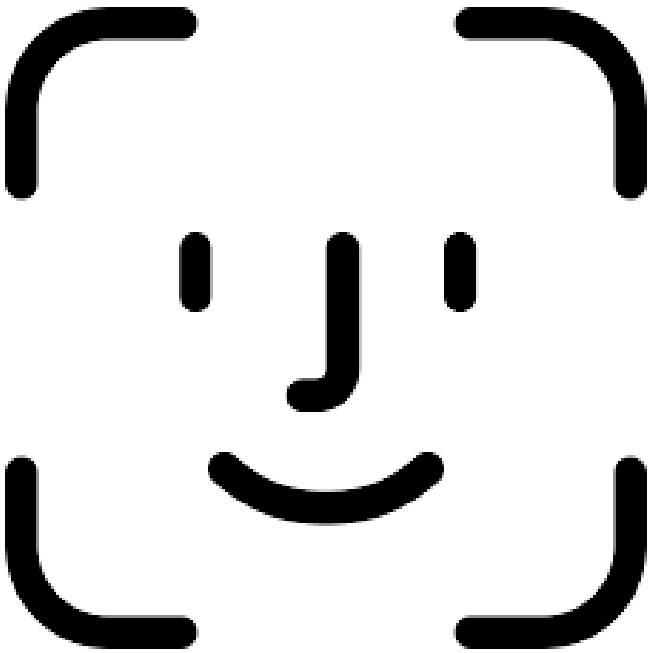


Table des matières

1 Introduction

Notre projet vise à créer une solution de reconnaissance faciale rapide et robuste dont des démonstrations pourront être effectuées à l'aide d'une application graphique. Les grandes étapes de ce projet : localiser un visage dans une image, reconnaitre un visage et la création de l'application.

2 Les technologies utilisées de nos jours

Les technologies utilisées de nos jours dans le domaine de la reconnaissance faciale ont considérablement évolué pour répondre aux demandes croissantes en matière de précision et de fiabilité. Les algorithmes d'apprentissage profond, tels que les réseaux de neurones convolutifs (CNN), sont devenus omniprésents pour la détection et la classification des visages. Ces réseaux sont capables d'apprendre des caractéristiques discriminantes à partir de grandes quantités de données d'entraînement, ce qui leur permet d'identifier efficacement des visages dans des conditions variables. De plus, l'utilisation de techniques de prétraitement d'images, telles que la normalisation et l'augmentation de données, contribue à améliorer la robustesse des modèles de reconnaissance faciale. Parallèlement, l'intégration de matériels spécialisés, comme les unités de traitement tensoriel (TPU) et les unités de traitement graphique (GPU), accélèrent les calculs nécessaires à l'inférence en temps réel. En combinant ces avancées, les technologies actuelles offrent des performances remarquables en matière de reconnaissance faciale, ouvrant la voie à une gamme étendue d'applications allant de la sécurité biométrique à la personnalisation de l'expérience utilisateur.

3 Tâches à réaliser

3.1 Détection des visages

3.1.1 Introduction

La partie concernant la détection des visages sera illustrée à l'aide de l'image suivante :

Le choix de la bibliothèque a été un processus difficile pour nous. Bien que la bibliothèque OpenCV offre une large gamme de fonctionnalités, son utilisation aurait pu être pertinente. Cependant, en tant qu'étudiants à Epita travaillant sur un projet de classe, nous avons estimé que l'achèvement de notre tâche en seulement cinq lignes en utilisant OpenCV ne serait pas très significatif. C'est pourquoi nous avons décidé d'opter pour une approche plus élémentaire en utilisant la bibliothèque "image" de Rust.

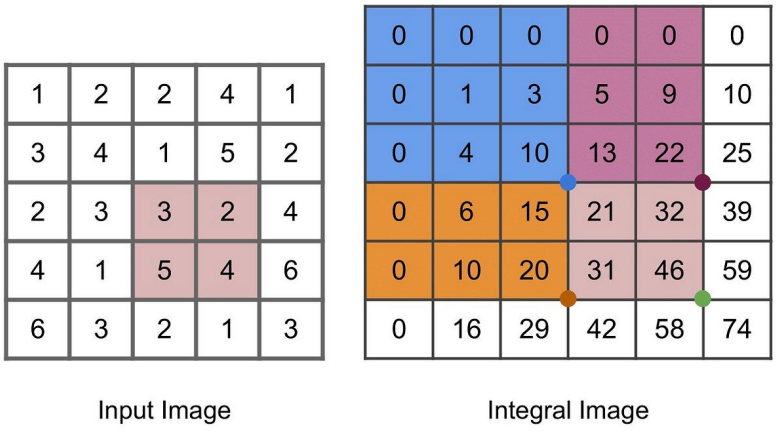
Ce choix a rendu la détection beaucoup plus délicate, mais sa réalisation nous apportera des compétences. De plus, un avantage de ce choix réside dans le partage du code. En effet, la bibliothèque "image" ne nécessite rien à télécharger, ce qui est particulièrement pratique pour exporter notre code, que ce soit à nos camarades ou à nous-mêmes lorsque nous changeons de machine.

3.1.2 Réalisation

Actuellement, nous disposons d'une fonction qui transforme une image, représentée par un vecteur de pixels, en un vecteur d'entiers. Ces entiers sont compris entre 1 et 10 en fonction du niveau de luminosité des pixels correspondants. La création de ce tableau d'entiers a une complexité linéaire : pour chaque élément, il suffit d'effectuer au maximum trois additions.

Le vecteur résultant représente l'intégrale de l'image, ce qui facilite la récupération des moyennes de pixels. Chaque zone de l'image peut être obtenue en effectuant une addition de trois composantes.

Voici un exemple de calcul de l'integrale de l'image :



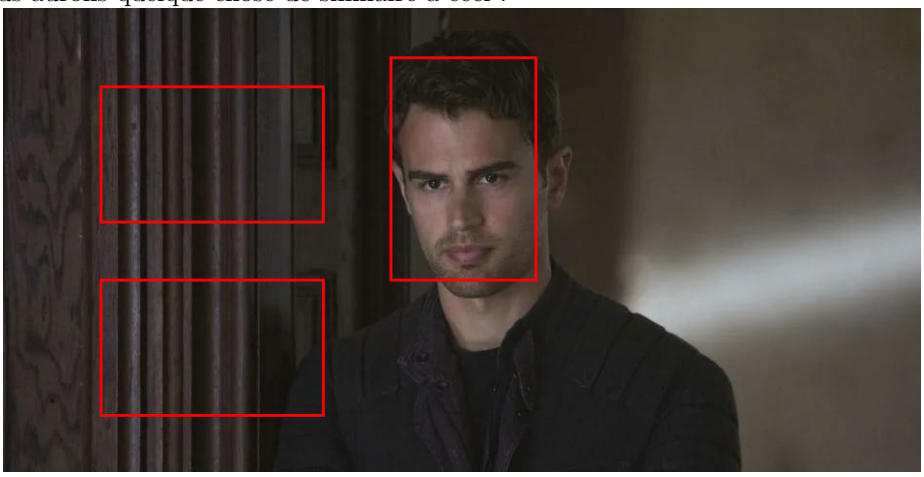
Pour atteindre cet objectif, nous comptons notamment utiliser la méthode de détection d'objets de Viola-Jones. Cette approche classique de détection de vi-

sages dans les images s'appuie sur la détection rapide de caractéristiques simples, telles que les bords, les coins et les zones de contraste, par l'intermédiaire d'une représentation intégrale de l'image. En utilisant un classificateur en cascade d'Adaboost pour combiner et sélectionner les caractéristiques les plus pertinentes, la méthode de Viola-Jones peut identifier de manière efficace et rapide les visages dans des images, même dans des conditions de faible résolution ou d'éclairage variable. Nous prévoyons également d'explorer des techniques de prétraitement d'images pour optimiser les performances de la détection de visages. Cette approche permet une détection efficace des visages en minimisant le temps de traitement et en améliorant les performances globales de l'algorithme.

Pour l'instant notre programme se concentre sur le traitement d'image, ce qui est essentiel pour la détection ultérieure des visages.

3.1.3 Objectif

Pour la prochaine soutenance, notre objectif sera de mettre en place des filtres pour récupérer les zones de l'image qui sont potentiellement des visages. Nous aurons quelque chose de similaire à ceci :



Pour la dernière soutenance, notre objectif principal sera la précision, visant à réduire, voire éliminer, le taux de faux visages détectés. De plus, nous

chercherons à détecter tous les visages présents dans l'image, même s'il y en a plusieurs.

Si possible, nous essaierons également de permettre à l'utilisateur de choisir le nombre de visages à détecter. Cela offrira une flexibilité supplémentaire dans l'utilisation de notre programme.

3.2 UI

3.2.1 Introduction

Pour concevoir l'interface graphique de notre application, nous avons opté pour la bibliothèque GTK-rs. Cette décision s'est appuyée sur notre expérience antérieure avec cette bibliothèque lors du projet de s3, en plus de sa maturité et de sa documentation approfondie, facteurs susceptibles de favoriser une rapide progression du développement de l'application.

Dans un premier temps, l'utilisateur pourra s'enregistrer soit en important des images de son visage, soit en utilisant la webcam de son ordinateur pour prendre des photos à l'intérieur même de l'application. Dans un second temps, afin de montrer les capacités de notre reconnaissance faciale, de nouvelles images pourront être prises ou importées, l'application donnera un retour en fonction de si oui ou non, il s'agit de la même personne enregistrée en premier lieu. Voici le site de la bibliothèque : `https://gtk-rs.org` Voici un site faisant part de l'état de l'art du gui en rust qui m'a aiguillé dans le choix de cette bibliothèque : `https://blog.logrocket.com/state-rust-gui-libraries`.

3.2.2 Réalisation

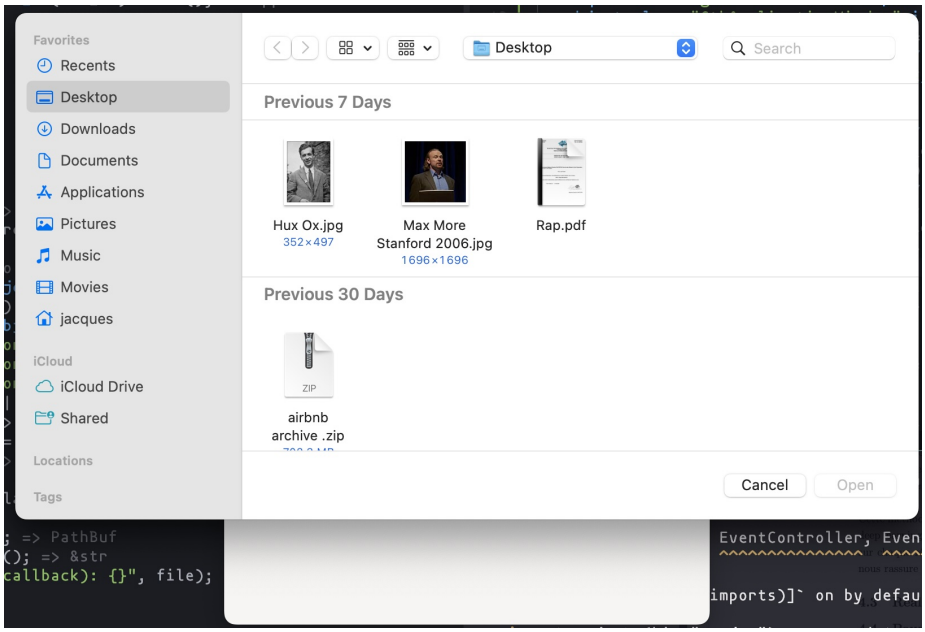
La réalisation de l'interface graphique a été une phase cruciale et enrichissante de ce projet, dans laquelle nous avons utilisé la bibliothèque gtk4-rs. Cette bibliothèque, bien qu'elle soit construite sur la base de GTK 3 en langage C, présente des spécificités qui ont nécessité une compréhension approfondie. L'une des différences notables réside dans les fonctions de rappel (callbacks) qui sont désignées autrement, avec la disparition ou l'obsolescence de certaines d'entre elles. Un défi majeur a été de constater l'absence de gpointer pour transmettre des variables à modifier dans les fonctions de rappel. Pour contourner cette limitation, nous avons adopté l'utilisation de Cell et RefCell, des mécanismes permettant de passer et de sauvegarder les variables lors des appels de fonctions de rappel.

Au fil du développement, il est devenu évident que cette bibliothèque est relativement nouvelle, ce qui se reflète dans la qualité de la documentation disponible. Souvent, les informations sont soit obsolètes, soit insuffisamment détaillées, avec de nombreuses fonctions manquant d'explications claires. Il est à noter également qu'il existe très peu de tutoriels sur l'utilisation de GTK4 avec Rust, bien que quelques ressources aient été trouvées, elles étaient souvent dépassées. Cependant, certaines de ces ressources se sont révélées précieuses pour comprendre les fonctionnalités non documentées.

Pour définir les éléments présents dans l'interface utilisateur (UI), j'ai utilisé la commande gtk4 tool simplify pour convertir le fichier .glade en fichier .ui, éliminant ainsi les balises XML incompatibles avec GTK4. Glade et les fichiers .ui sont tous deux écrits en XML et permettent de spécifier l'emplacement des éléments de l'interface utilisateur.

Un aspect particulièrement chronophage de ce projet a été de comprendre le fonctionnement de la bibliothèque GTK4 et de traduire les pratiques établies avec GTK3. Par exemple, dans GTK3 en langage C, l'élément d'interface utilisateur "filechooser" est devenu obsolète en GTK4, remplacé par "file dialog" avec une fonction "open" pour ouvrir la fenêtre de sélection de fichier. Cette migration a demandé du temps et de la patience pour s'adapter aux changements de fonctionnalités et de nomenclature, ainsi que pour maintenir la cohérence et la compatibilité avec les versions précédentes du logiciel.

Pour instant, ce qui à été fait sont , le ui avec 4 buttons dont un button qui permet de choisir l'image à afficher et la zone à afficher. Pour les prochaines soutenances, j'implémenterai les couleurs de l'interface et faire en sorte que la taille de l'image s'adapte dynamiquement à la taille de la fenêtre.


Favorites
Recents
Desktop
Downloads
Documents
Applications
Pictures
Music
Movies
jacques
iCloud
iCloud Drive
Shared
Locations
Tags
Desktop
Search
Previous 7 Days
Hux Ox.jpg
352×497
Max More Stanford 2006.jpg
1696×1696
Rap.pdf
Previous 30 Days
ZIP
airbnb archive .zip
Cancel
Open

3.3 Reconnaissance faciale

3.3.1 Présentation

Cette étape a pour but de donner une distance entre deux visages qui pourra ensuite être utilisé avec un simple mécanisme de seuil pour considérer que oui ou non les deux visages présentés sont les mêmes. Le process intervient après la localisation du visage dans l'image et comparera l'image extraite du visage avec les différentes images de l'utilisateur préalablement enregistré.

Au vu de la difficulté de la tâche, uniquement des méthodes de deep learning sont envisageables pour extraire les caractéristiques d'un visage. Pour cela, nous avons prototypé un réseau de neurone convolutif (CNN) de taille réduite afin de se familiariser avec la création de réseau de neurones en Rust. nous utilisons la bibliothèque tch-rs. Elle consiste en des liaisons Rust pour l'api C++ de PyTorch. Cela permet pour nous d'être en terrain connu et de s'assurer l'accès à une documentation riche. En effet, la seule alternative, le seul autre framework de Deep Learning en Rust, Burn, est encore à un stade primaire de développement. Nous pensons que ce n'est pas une bonne initiative que de baser notre projet sur des bibliothèques en "beta".

3.3.2 Protocole

Entrainé un réseau de neurones directement pour de la reconnaissance faciale et inenvisageable car la tâche et ardu et qu'il n'existe pas de dataset contenant un nombre suffisant de personnes avec un nombre suffisant d'images de chaque personne dans des contextes différents pour envisager entrainer directement un réseau de la sorte. L'idée est qu'au lieu d'espérer un réseau magique sortant une probabilité que deux visages soient similaires, on ait un réseau extrayant les caractéristiques d'un visage qui pourront ensuite être comparées et traitées de manière séparée.

Pour traiter une image, nous considérons évidemment d'utiliser un réseau de neurones convolutif. Par considération des difficultés présentées précédemment, le réseau doit être entrainé dans un premier temps sur des choses génériques, différencier des objets, des animaux. Cela permet au réseau de comprendre les mécanismes de base de la reconnaissance de caractéristiques.

Ensuite, le réseau peut être spécialisé dans la reconnaissance d'attributs faciaux. Le dataset CelebA est une option envisagée pour cela, l'objectif pour le réseau est de dire si quarante attributs son présent ou non sur le visage présenté, par exemple : la couleur des yeux, l'expression faciale, le sexe. Grâce à cela, le réseau se spécialise spécifiquement dans le traitement de visages.

Ainsi, si on retire les layers totalement connectés du réseau utilisé pour la prédiction finale et qu'on se suffit des layers convolutionnels le réseau donne une soupe d'information résumant le visage, donnant les informations les plus importantes. Ce sont ces informations qui peuvent ensuite être utilisées pour déduire si oui ou non deux visages sont similaires.

En effet, en utilisant une simple distance euclidienne sur cette "soupe" et en définissant de manière empirique un seuil, nous pouvons prédire si deux visages sont similaires. Si cette option n'est pas fructueuse, une alternative existe, entrainer un petit réseau de type MLP afin de donner une mesure de distance, de ressemblance.

Cette méthode est similaire à ce qui se fait depuis des années dans le domaine du deep learning et ne présente pas de risque d'échec en soit, la difficulté se présente sur chaque réalisation individuelle plus que sur l'ensemble du protocole ce qui nous rassure afin de mener à bien ce projet.

3.3.3 Avancement

Nous nous sommes familiarisés avec tch-rs et avons créé un réseau de neurone convolutif (CNN) entrainé pour reconnaitre les dix catégories que contient CIFAR-10. Ce dataset de 60K images est composé de dix catégories (chien, avion, camion, etc.) utilisé pour benchmarker une architecture ou justement dans notre cas préentrainer un modèle. Le modèle, dont l'architecture suit, a été entrainé avec une logloss en fonction coût et Adam, optimizer réputé pour ses bons résultats avec une convergence rapide et des batchs de 32 images.

Type	Output Shape
Input	$[3, 32, 32]$
Conv2D $(3, 32, 3 \times 3)$	$[32, 32, 32]$
MaxPool2d	$[32, 16, 16]$
ReLU	$[32, 16, 16]$
Conv2D $(32, 64, 3 \times 3)$	$[64, 16, 16]$
MaxPool2d	$[64, 8, 8]$
ReLU	$[64, 8, 8]$
Conv2D $(64, 128, 3 \times 3)$	$[128, 4, 4]$
MaxPool2d	$[128, 2, 2]$
ReLU	$[128, 2, 2]$
Flatten	$[512]$
Linear $(512, 1024)$	$[1024]$
ReLU	$[1024]$
Linear $(512, 1024)$	$[10]$
Softmax	$[10]$

TABLE 1 – Architecture du CNN

Ce modèle atteint une précision de 95% sur le jeu de test après 50 époques, ce qui prend une quinzaine de minutes. Nous sommes satisfaits des résultats et sommes confiants pour faire évoluer l'architecture du réseau en ajoutant de nouveaux blocs de convolution afin de le préentrainer sur CIFAR-10, voire CIFAR-100 puis transférer l'apprentissage sur CelebA afin de décortiquer les caractéristiques d'un visage.

4 Tableau de répartitions des tâches

Taches	Responsable
UI	Jacques
Détection visages	Mathys & Alexis
Reconnaissance faciale	Clément

5 Planning de réalisation

En avance sur la réalisation du projet, nous maintenons notre planning.

Taches	Soutenance 1	Soutenance 2	Soutenance 3
Détection des visages	20%	60%	100%
UI	20%	60%	100%
reconnaissance faciales	20%	60%	100%