

Prédiction de l’âge d’une personne

**Compte rendu Projet Annuel**

Antoine BOUDEVILLE

Alexis LIBERGE

Nicolas ROCHE

Table des matières

[Présentation du projet 3](#_Toc4172905)

[Objectifs 3](#_Toc4172906)

[Structure et technologies 3](#_Toc4172907)

[Constitution du dataset 4](#_Toc4172908)

[Recherche de dataset existant 4](#_Toc4172909)

[Enrichissement du dataset 4](#_Toc4172910)

# Présentation du projet

## Objectifs

L’objectif de ce projet est de créer une application web permettant de prédire l'âge d'une personne à partir d'une photo de son visage.

Les étapes de ce projet sont : la constitution d’un dataset, l’implémentation des différents algorithmes de machine learning, la mise en place de ces même algorithmes avec le framework Tensorflow afin de comparer les résultats et la mise en place d’une API et d’un site web permettant l’utilisation de cette application.

## Structure et technologies

Cette application est divisée en plusieurs parties avec des technologies différentes. Toutes les algorithmes de machine learning sont implémentés en C/C++. Le framework Tensorflow est utilisé avec Python. L’API est implémentée en Python aussi avec le framework Flask. Le site web est développé lui en PHP/Javascript avec le framework Materialize (front-end). Enfin une application qui permet de retoucher les photos du dataset codé en C#.

Le dataset, le site web et l’API sont hébergés sur un VPS.

# Constitution du dataset

## Recherche de dataset existant

## Enrichissement du dataset

Afin de réaliser notre propre Dataset nous avons décidé d’extraire les photographies de vidéos Youtube montrant l’évolution d’un individu de manière journalière ou annuel.

Nous avons donc réalisé des Datasets avec comme source les deux vidéos suivante :

Lotte: <https://www.youtube.com/watch?v=nPxdhnT4Ec8>

Stephanie: <https://www.youtube.com/watch?v=Ec9T__AexyU>

Pour crée des Datasets, nous avons réalisé les étapes ci-dessous :

### Récupération des vidéos YouTube

La première étape dans la réalisation à été de récupérer via un exporter en ligne chaque vidéo.

Pour cela il fallait juste entrer le lien de la vidéo YouTube pour obtenir une version mp4 téléchargeable.

### Récupération des images des vidéos

Pour récupérer les images des vidéos, nous avons ouvert la vidéo dans Photoshop et utilisé une fonctionnalité de l’application pour extraire ces images.

Avec quelques configurations nous avons réussi à exporter ces images à un rythme de 30 images par seconde de vidéo.

Cependant dans certains cas nous avons remarqué des doublons.

### Suppression des doublons

Grâce à une application que nous avons développé (YouTube Image Dataset).

Nous avons parcouru l’ensemble des images obtenue avec Photoshop et généré un tableau de 16\*16 pixels en noir et blanc pour chaque image.

Ces tableaux ont été ensuite comparés pour supprimer les tableaux similaires.

### Ajout de métadonnées pour les images

Après avoir supprimé les doublons, nous avons ajouté des métadonnées associé à chaque image.

Celle-ci sont stockés dans le nom de chaque image et on la syntaxe suivante :

*age\_genre\_race\_date\_id*

Les informations ***genre*** et ***race*** sont sélectionnées via l’application à l’aide de liste déroulante.

Pour l’âge nous avons créé 2 manières d’attribuer un Age.

Pour le dataset de Lotte

Un marqueur sont présent à chaque changement d’Age sur la photo, nous avons donc crée un dossier pour chaque Age et copier les images dans le dossier correspondant.

L’application parcoure ensuite chaque dossier et extrait l’âge du nom du dossier et renomme chaque image avec l’âge obtenu et les autres éléments sélectionnés.

Pour le dataset de Stéphanie

Aucun marqueur n’est présent dans les images, l’application parcoure dans ce cas l’ensemble des images avec un âge de départ et un intervalle (nombre de jours).

L’application renomme chaque image avec l’âge actuel.

A chaque image on ajoute l’intervalle à un index quand celui-ci atteint 365, on augmente l’âge de 1.

### Suppression des images floues et potentiels biais.

Nous avons ensuite supprimer manuellement les photos qui etaient floues ( transition entre les photos) et celle où le visage n’apparaissaient pas clairement.

# Librairies Utilisés

Notre Librairie de machine Learning utilise 2 librairies afin de fonctionner :

* Eigen : cette librairie est utilisée pour faire des opérations et manipulations vectorielles et matricielles.
* OpenCV : Cette librairie est utilisée pour extraire les pixels d’une image du dataset sous la forme d’une matrice.

# Régression Linéaire (Moore-Penrose)

La régression linéaire est un modèle de régression qui cherche à établir une relation linéaire entre les entrées et les sorties.

## Exemple d’utilisation

## Entrainement

Pour l’entrainement de la régression linéaire, notre librairie utilise la formule avec la Pseudo-inverse de Moore-Penrose.

## Prédiction

Pour la prédiction, il suffit de multiplier la matrice de poids ayant une forme (x, 1) avec la matrice des inputs (1, x).

Le résultat de cette multiplication matricielle donne une matrice (1,1) correspondant à la prédiction.

Note : x correspond au nombre d’entrée par image.

## Problèmes rencontrés

Nous avons rencontré des problèmes avec la formule d’entrainement car si la matrice contient plus de colonnes que de ligne la formule à utiliser est :

L’autre formule ne fonctionnait pas lorsque nous testions certains cas.

Pour résoudre le problème nous avons utilisé l’implémentation du pseudo inverse d’Eigen qui utilise automatiquement la bonne formule en fonction de la matrice.

## Sauvegarde des poids et Chargement

Les poids peuvent être sauvegardés sous un format CSV, celui –ci contiendra l’ensemble des poids sous une ligne.

La librairie permet aussi le chargement du modèle via CSV.

## Résultats

Cas de tests

Datasets

Tout d’abord nous avons commencé avec un dataset qui possède une résolution 20 \* 10.

Les résultats étaient mitigés. Puis nous avons augmenté la taille et la définition des images et nous avons obtenu de meilleurs résultats

# Classification Linéaire (Perceptron de Rosenblatt)

Le perceptron de Rosenblatt est un réseau de neurone simple avec une seule couche. On parle de classifieur linéaire.

## Exemple d’utilisation

## Entrainement

Notre librairie implémente la règle d’apprentissage du perceptron.

Cet entrainement itératif se base sur la répétition de la formule :

Dans cette formule :

* Alpha correspond au pas d’apprentissage
* correspond à la sortie attendue
* correspond à la prédiction

## Prédiction

La prédiction de la classification linéaire utilise la prédiction de la régression linéaire en appliquant la fonction signe sur le résultat afin d’obtenir -1 ou 1.

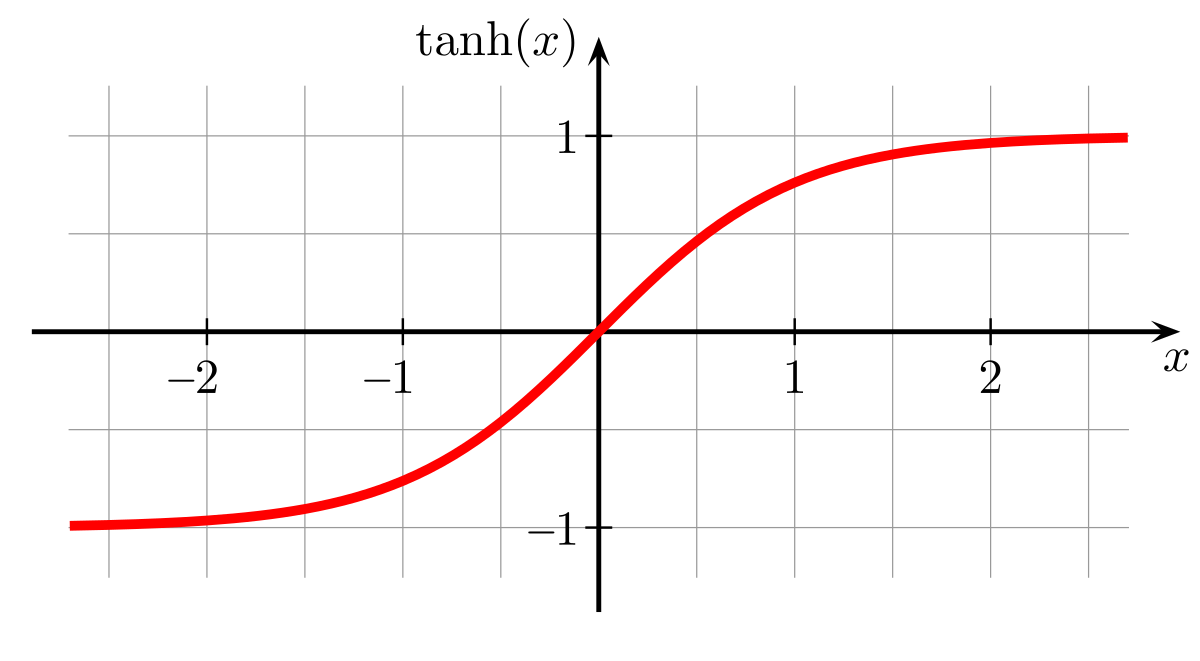
# Perceptron Multicouches

Le perceptron multicouches est un type de réseau de neurones. Le PMC peut contenir des couches cachés et la dernière couche correspond aux sorties (résultats).

Notre Librairie implémente deux types de PMC, un pour la régression et l’autre pour la classification.

## Fonction d’activation

La fonction d’activation utilisée est la tangente hyperbolique qui permet d’obtenir un résultat entre -1 et 1.



Représentation de la fonction tangente hyperbolique.

## Structure du PMC

La structure du réseau de neurones est définie par un tableau de nombres.

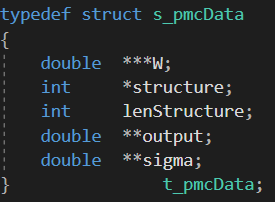
Le nombre de neurones pour la couche d’index i correspond à sa valeur dans le tableau.

La structure [500, 2, 1] correspond :

* 500 neurones en couche 0 ( inputs ).
* 2 neurones cachés en couche 1
* 1 neurone de sortie.

### Structure de données du PMC

Afin de stocker les différents éléments nécessaires pour le PMC, nous avons créé une structure permettant de stocker tous ces éléments.



Dans cette structure :

* W correspond aux poids.
* structure correspond à la structure du réseau de neurones.
* lenStructure est le nombre de couche du neurone.
* Output contient les sorties obtenues lors de la propagation.
* Sigmas contient les sigmas obtenues lors de la rétropopagation.

## Entrainement

Pour L’entrainement le PMC que nous avons implémenté se base sur propagation et rétropopagation du gradient.

L’entrainement se déroule en 3 étapes :

* Propagation du gradient.
* Rétropopagation du gradient.
* Mise à jour des poids.

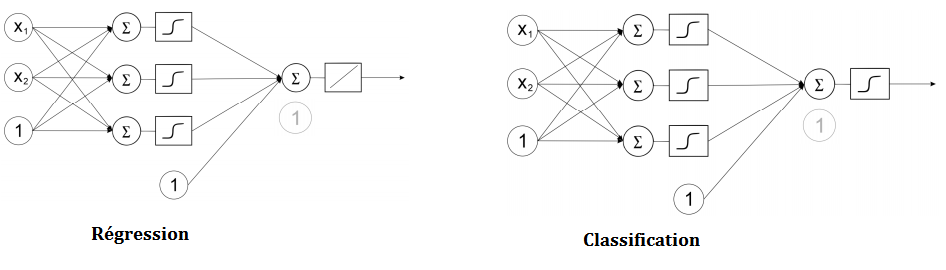
### Propagation du gradient

Cette étape consiste à calculer les sorties de chaque neurone et d’appliquer la fonction d’activation.

La sortie d’un neurone correspond à la formule suivante :

## C:\Users\WT57\AppData\Local\Microsoft\Windows\INetCache\Content.Word\chrome_2019-06-14_14-41-33.png

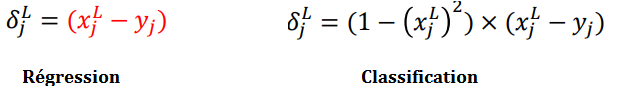
Pour cette étape il existe un cas particulier, à la dernière couche la fonction d’activation n’est pas appliquée pour la régression.



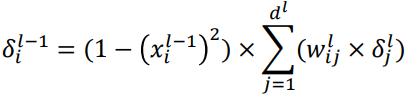
### Rétropopagation du gradient

La seconde étape est la rétropopagation, cette étape consiste à calculer les sigmas (gradient de l’erreur) pour chaque neurone en partant de la dernière couche à la première.

La formule pour la dernière couche est :



La formule de l’avant dernière couche à la première couche est :



### Mise à jour des poids

La dernière est la mise à jour des poids avec les sigmas calculés précédemment.

La formule est la suivante :



## Prédiction

La prédiction du PMC correspond à la réalisation d’une propagation du gradient.

La sortie de la dernière couche correspond à la prédiction.

Note : pour la classification il y aura plusieurs sorties correspondant à chaque classe.

## Sauvegarde du PMC et chargement

Le modèle entrainé du PMC peut être sauvegardé sous un format CSV grâce à la librairie.

Le fichier CSV contient :

* Ligne 1 = structure du PMC
* Ligne 2 à N = poids des neurones de la couche

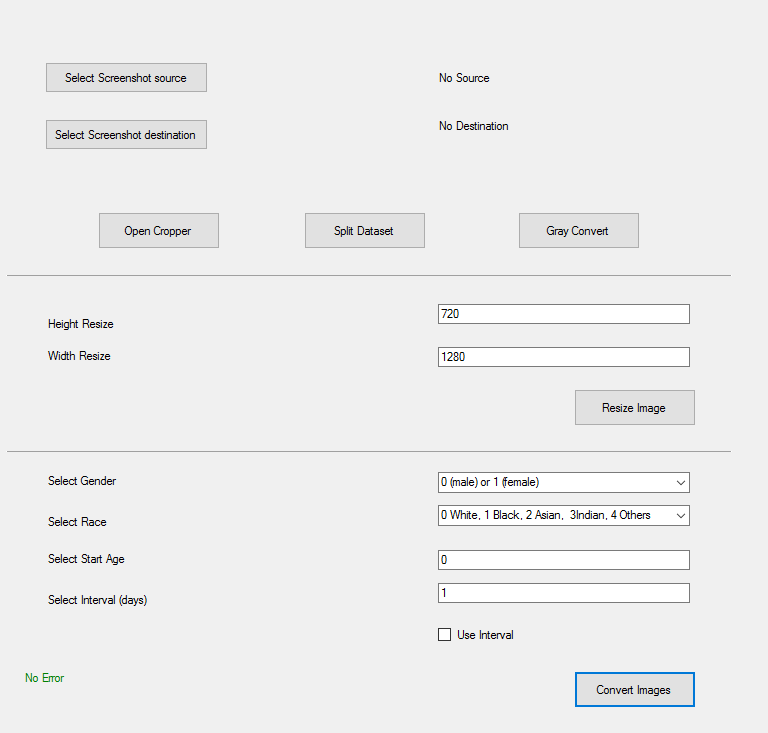
Le modèle peut aussi être chargé afin d’être utilisé grâce à une méthode de la librairie C++.

## Résultats

# Application ToolBox : Youtube Image Dataset

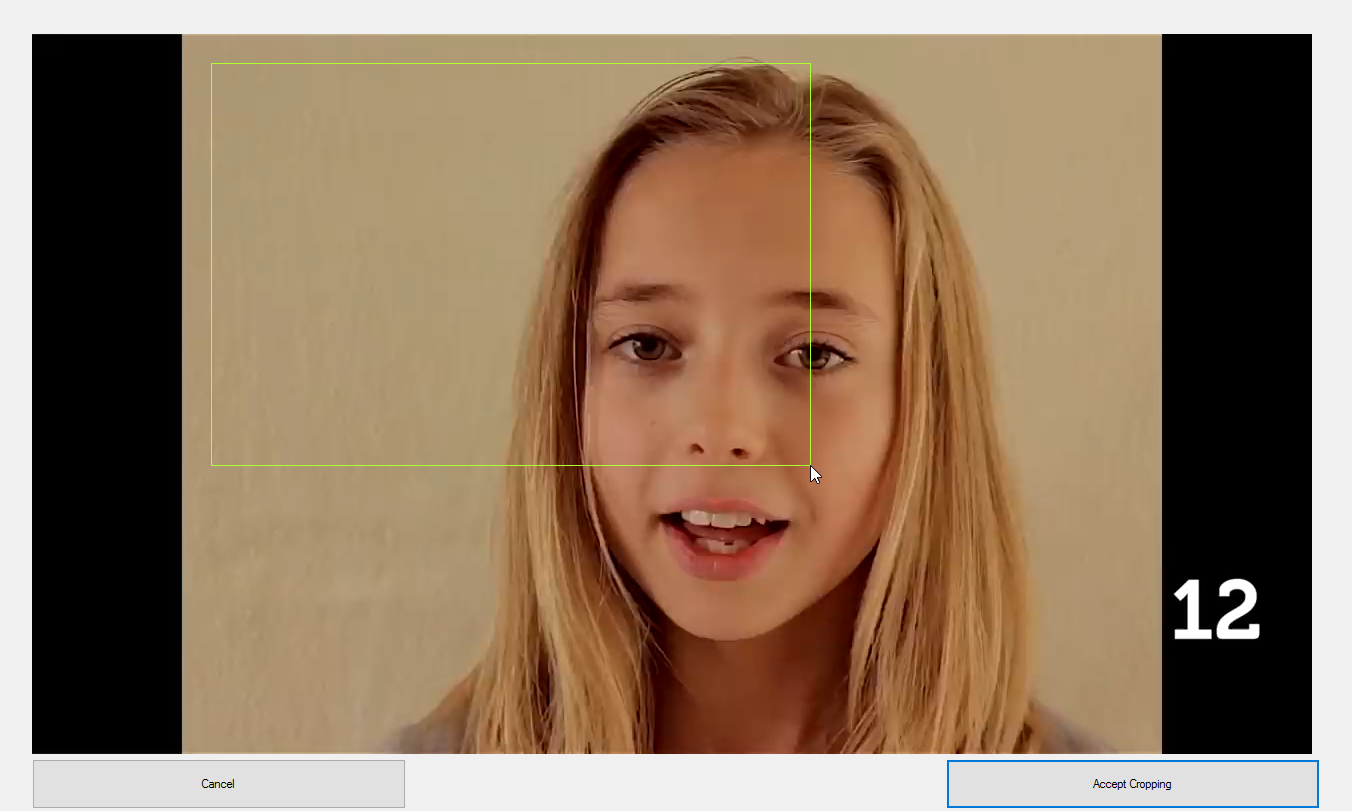
Nous avons réalisé une application en C# que nous avons nommé « YoutubeImageDataset ».

Cet outil est une boite à outil nous permet d’effectuer un multiple d’actions sur notre dataset.



Cet outil nous permet

* Rognage de chaque photos Dataset afin d’éliminer les bandes noires ou de sélectionner une zone précise.
* Convertir les images en nuances de gris
* Redimensionner l’image afin à une taille spécifique
* Séparation aléatoire du Dataset en 2 dossiers (Train, Test)



Exemple de « Rognage » avec L’application.