

**Computer Vision**  
**L'estimation ou détection faciale de l'âge en  
utilisant un modèle de l'apprentissage en profond**  
**(Étude Comparative)**

**Préparé par :**

- EL AYADI Fatima Ezzahra
- BOUYARMANE Chaimae

**Sous la direction de :**

**Mr. OUSGUINE Said**

**Filière : Master M2SI**

**Année universitaire : 2023 - 2024**



<b>Introduction.....</b>	<b>3</b>
<b>Contexte générale du projet.....</b>	<b>4</b>
1. Importance de l'estimation de l'âge dans divers domaines :.....	4
2. Problématique de la précision dans l'estimation faciale de l'âge :.....	4
<b>La Méthodologie du travail.....</b>	<b>5</b>
1. Description des données utilisées :.....	5
2. Prétraitement des données :.....	5
➤ Exploration de la structure du jeu de données :.....	5
➤ Défloutage des images :.....	5
➤ Prétraitements d'image :.....	6
➤ Division du jeu de données :.....	6
➤ Code Python pour le prétraitement :.....	6
3. Choix de Modèle d'Apprentissage en Profondeur :.....	7
➤ Modèle Convolutionnel (CNN) :.....	7
➤ L'architecture du CNN :.....	7
➤ Implémentation d'une architecture CNN Personnalisé:.....	8
4. Paramètres d'entraînement et d'évaluation :.....	9
➤ Paramètres d'Entraînement :.....	9
➤ Paramètres d'Évaluation :.....	10
<b>Architecture des Modèles:.....</b>	<b>11</b>
1. Présentation détaillée de chaque modèle:.....	11
1.1 VGG16 (Visual Geometry Group 16).....	11
1.2 ResNet50 (Residual Network).....	11
1.3 InceptionV3.....	12
2. Architectures de Modèles pour l'Estimation de l'Âge.....	12
1. Architecture du modèle VGG16 utilisée:.....	13
2. Architecture du modèle ResNet50 utilisée:.....	14
3. Architecture du modèle InceptionV3 utilisée:.....	15
1. Interprétation des Résultats.....	18
1.1. Modèle CNN Personnalisé :.....	18
1.2. Modèle VGG16 :.....	18
1.3. Modèle ResNet50:.....	19
1.4. InceptionV3:.....	20
2. Visualisation des prédictions:.....	21
● Modèle CNN Personnalisé :.....	21
● Modèle VGG16 : .....	23
● Modèle ResNet50:.....	24
● InceptionV3:.....	25
<b>Benchmarking.....</b>	<b>27</b>
<b>Conclusion:.....</b>	<b>29</b>
<b>Bibliographie.....</b>	<b>30</b>

# Introduction

L'estimation ou la détection de l'âge facial représente un domaine dynamique et en constante évolution au sein de la vision par ordinateur, suscitant un intérêt significatif en raison de ses applications variées, allant de la sécurité à la personnalisation des expériences utilisateur. Avec l'émergence des modèles d'apprentissage en profondeur, il devient possible d'explorer des approches plus sophistiquées et précises pour cette tâche complexe.

Cette étude vise à examiner de manière approfondie différentes méthodes d'estimation ou de détection de l'âge facial en se concentrant sur l'application de modèles d'apprentissage en profondeur. En effet, les réseaux neuronaux profonds ont démontré une capacité remarquable à extraire des caractéristiques complexes à partir de données visuelles, les rendant particulièrement adaptés à des tâches telles que la reconnaissance faciale.

Dans ce contexte, nous entreprenons une étude comparative des performances de divers modèles d'apprentissage en profondeur appliqués à l'estimation ou à la détection de l'âge facial. Cette analyse approfondie permettra de comprendre les forces et les limitations de chaque approche, fournissant ainsi des informations cruciales pour le développement futur de systèmes plus robustes et précis.

Au cours de cette exploration, nous aborderons également les défis spécifiques liés à la tâche d'estimation ou de détection de l'âge facial, tels que la variabilité des expressions faciales, l'impact de l'éclairage et les différences ethniques. Notre objectif est de fournir des insights pratiques et des recommandations pour orienter le choix des modèles dans des applications réelles, en prenant en compte la fiabilité, la précision et la généralisation.

En résumé, cette analyse contribuera à l'avancement des techniques de traitement d'image, offrant des perspectives critiques sur l'utilisation des modèles d'apprentissage en profondeur dans le domaine de l'estimation d'âge facial.

# **Contexte générale du projet**

L'estimation de l'âge à partir de l'analyse faciale a acquis une importance croissante dans une variété de domaines, allant de la sécurité au marketing en passant par les soins de santé. En effet, la capacité à estimer l'âge d'une personne à partir de son visage peut avoir des implications significatives. Dans le domaine de la sécurité, par exemple, cela peut contribuer à l'amélioration des systèmes de surveillance en identifiant les individus susceptibles d'être impliqués dans des activités suspectes en fonction de leur groupe d'âge. Dans le secteur du marketing, la publicité ciblée en fonction de l'âge est devenue une pratique courante pour accroître l'efficacité des campagnes.

## **1. Importance de l'estimation de l'âge dans divers domaines :**

L'estimation de l'âge est cruciale dans plusieurs contextes. Dans les domaines de la sécurité et de la surveillance, elle peut contribuer à l'identification de groupes d'âge spécifiques liés à des comportements préoccupants. Dans les applications de soins de santé, la capacité à estimer l'âge peut faciliter la personnalisation des traitements en fonction des besoins spécifiques liés à la tranche d'âge d'un individu. De même, dans le secteur de la publicité, comprendre le groupe d'âge cible permet d'adapter les messages de manière plus précise pour atteindre un public spécifique.

## **2. Problématique de la précision dans l'estimation faciale de l'âge :**

Cependant, la précision de l'estimation faciale de l'âge reste une problématique centrale. Les méthodes traditionnelles, basées sur des caractéristiques morphologiques ou des approches statistiques, peuvent être limitées dans leur capacité à fournir des résultats précis. C'est dans ce contexte que les modèles d'apprentissage en profondeur offrent une nouvelle perspective, en exploitant des réseaux neuronaux complexes pour extraire des caractéristiques plus riches et améliorer la précision de l'estimation. Cette étude vise à explorer ces modèles en profondeur et à les comparer aux approches traditionnelles, dans le but de mieux comprendre leur potentiel et leurs limites dans le domaine de l'estimation faciale de l'âge.

# La Méthodologie du travail

## 1. Description des données utilisées :



On a utilisé le dataset UTKFace comme source principale de données pour cette étude. Ce dataset est composé d'un ensemble étendu d'images faciales annotées, couvrant une diversité de conditions de capture. Chaque image est associée à des métadonnées telles que l'âge, le sexe et l'origine ethnique du sujet. Avec une représentation équilibrée des différentes tranches d'âge, allant de jeunes enfants à des personnes âgées, le dataset offre une variété d'exemples pour évaluer la capacité des modèles d'estimation et de détection de l'âge facial.

## 2. Prétraitement des données :

Le prétraitement des données est une étape cruciale pour garantir la qualité et la cohérence des informations fournies aux modèles d'apprentissage en profondeur. Dans cette section, nous décrirons en détail le processus de prétraitement des données appliqué au dataset UTKFace, ainsi que le code Python correspondant.

### ➤ Exploration de la structure du jeu de données :

Le prétraitement commence par explorer la structure du jeu de données situé dans le répertoire "/content/UTKFaceN". Les images sont extraites, et les noms de fichiers, qui contiennent des informations telles que l'âge, sont utilisés pour étiqueter chaque image.

### ➤ Défloutage des images :

Une fonction de défloutage “`deblur_image`” est implémentée pour améliorer la netteté des images. Cette fonction applique un filtre de déconvolution inversant le flou gaussien, contribuant ainsi à améliorer la qualité visuelle des images.

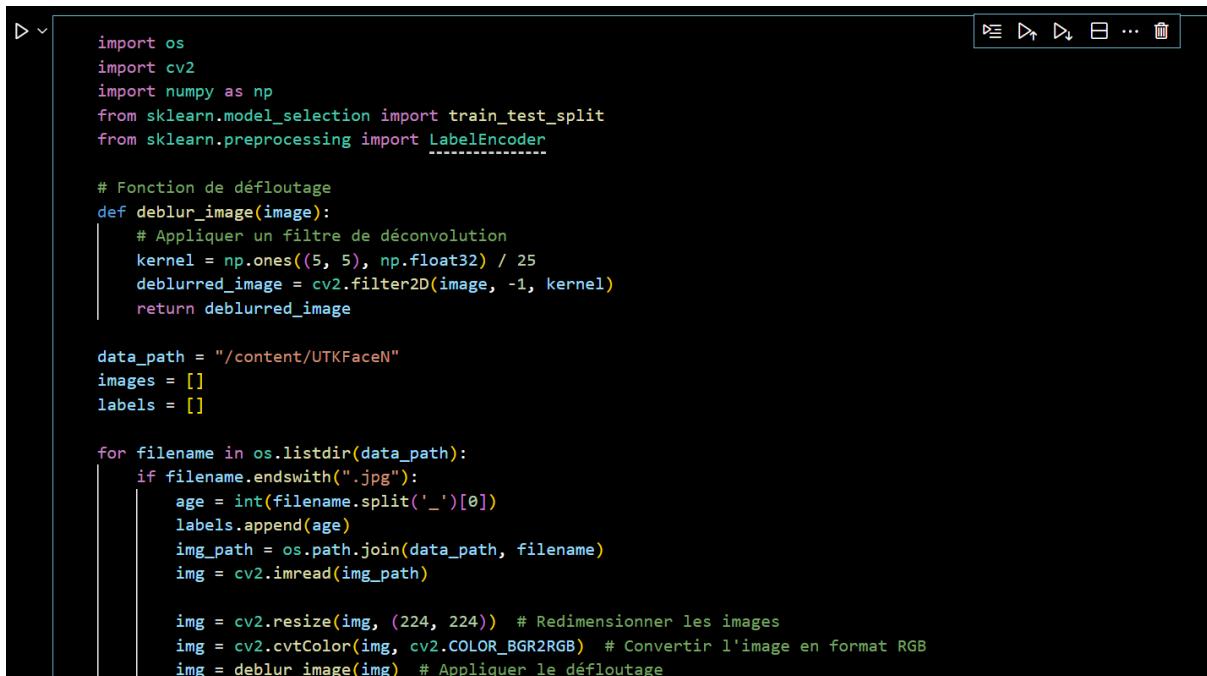
### ➤ Prétraitements d'image :

Différents prétraitements d'image sont appliqués, tels que le redimensionnement des images à 224x224 pixels, la conversion des images en format RGB, et l'application du défloutage pour réduire le bruit.

### ➤ Division du jeu de données :

Le jeu de données est divisé en ensembles d'entraînement, de validation et de test à l'aide de la fonction `train_test_split` de scikit-learn. Les images sont normalisées en mettant à l'échelle les pixels entre 0 et 1.

### ➤ Code Python pour le prétraitement :



```
import os
import cv2
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

# Fonction de défloutage
def deblur_image(image):
    # Appliquer un filtre de déconvolution
    kernel = np.ones((5, 5), np.float32) / 25
    deblurred_image = cv2.filter2D(image, -1, kernel)
    return deblurred_image

data_path = "/content/UTKFaceN"
images = []
labels = []

for filename in os.listdir(data_path):
    if filename.endswith(".jpg"):
        age = int(filename.split('_')[0])
        labels.append(age)
        img_path = os.path.join(data_path, filename)
        img = cv2.imread(img_path)

        img = cv2.resize(img, (224, 224)) # Redimensionner les images
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB) # Convertir l'image en format RGB
        img = deblur_image(img) # Appliquer le défloutage
```

```

+ Code + Markdown | ▶ Run All ⚡ Restart ⌂ Clear All Outputs | ⏷ Variables ⏷ Outline ... Python 3.11.5
▶ | | images.append(img)

# Convertir les listes en tableaux NumPy
images = np.array(images)
labels = np.array(labels)

# 2. Diviser le jeu de données en ensembles d'entraînement, de validation et de test
X_train, X_temp, y_train, y_temp = train_test_split(images, labels, test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)

# 3. Normaliser les images (mise à l'échelle des pixels entre 0 et 1)
X_train = X_train / 255.0
X_val = X_val / 255.0
X_test = X_test / 255.0

```

### 3. Choix de Modèle d'Apprentissage en Profondeur :

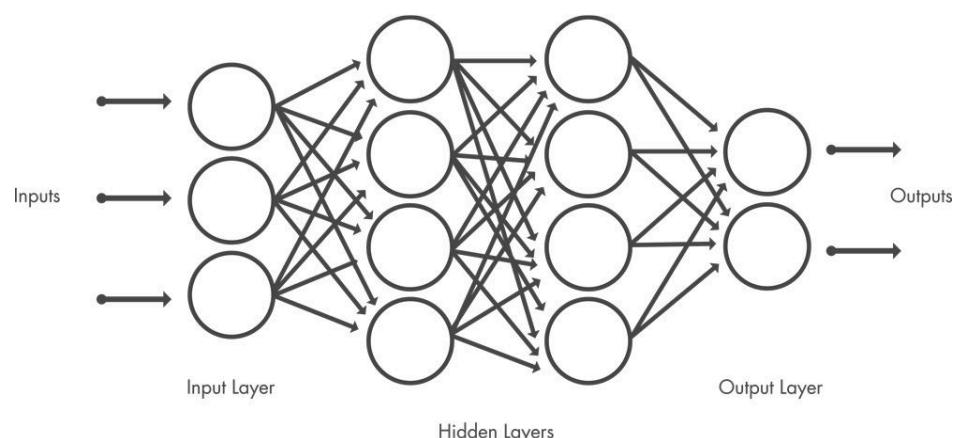
La sélection des modèles d'apprentissage en profondeur constitue une étape critique de notre démarche, influençant directement la précision et la performance de l'estimation et de la détection de l'âge facial. Nous avons opté pour une approche diversifiée en sélectionnant une architecture de réseaux neuronaux profonds reconnues pour leur efficacité dans des tâches de vision par ordinateur.

#### ➤ Modèle Convolutionnel (CNN) :

Un modèle CNN a été choisi en raison de son aptitude à extraire des caractéristiques spatiales hiérarchiques à partir des images. Cette architecture est adaptée à la reconnaissance de motifs complexes, ce qui en fait un choix pertinent pour la tâche d'estimation de l'âge facial. Le modèle CNN sera configuré avec des couches convolutionnelles suivies de couches entièrement connectées pour l'apprentissage des caractéristiques

#### ➤ L'architecture du CNN :

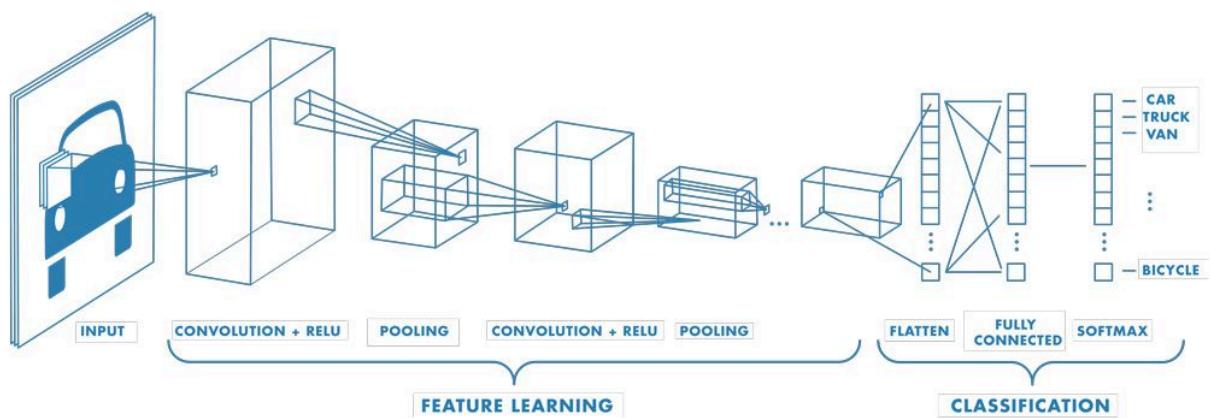
Un CNN comporte une couche d'entrée, une couche de sortie et de nombreuses couches intermédiaires cachées.



Ces couches effectuent des opérations pour modifier les données, afin d'en apprendre les caractéristiques spécifiques. Les trois couches les plus courantes sont : la convolution, l'activation (ou ReLU) et le pooling.

- **La couche de convolution** applique un ensemble de filtres convolutifs aux images en entrée, chacun d'entre eux activant certaines caractéristiques des images.
- **La couche ReLU (Rectified linear unit)** favorise un apprentissage plus rapide et plus efficace en remplaçant les valeurs négatives par des zéros et en conservant les valeurs positives. Ce procédé est parfois appelé activation, car seules les caractéristiques activées sont transmises à la couche suivante.
- **La couche de pooling** simplifie la sortie en réalisant un sous-échantillonnage non linéaire, ce qui permet de réduire le nombre de paramètres que le réseau doit apprendre.

Ces opérations sont répétées sur des dizaines ou des centaines de couches, chaque couche apprenant à identifier différentes caractéristiques.



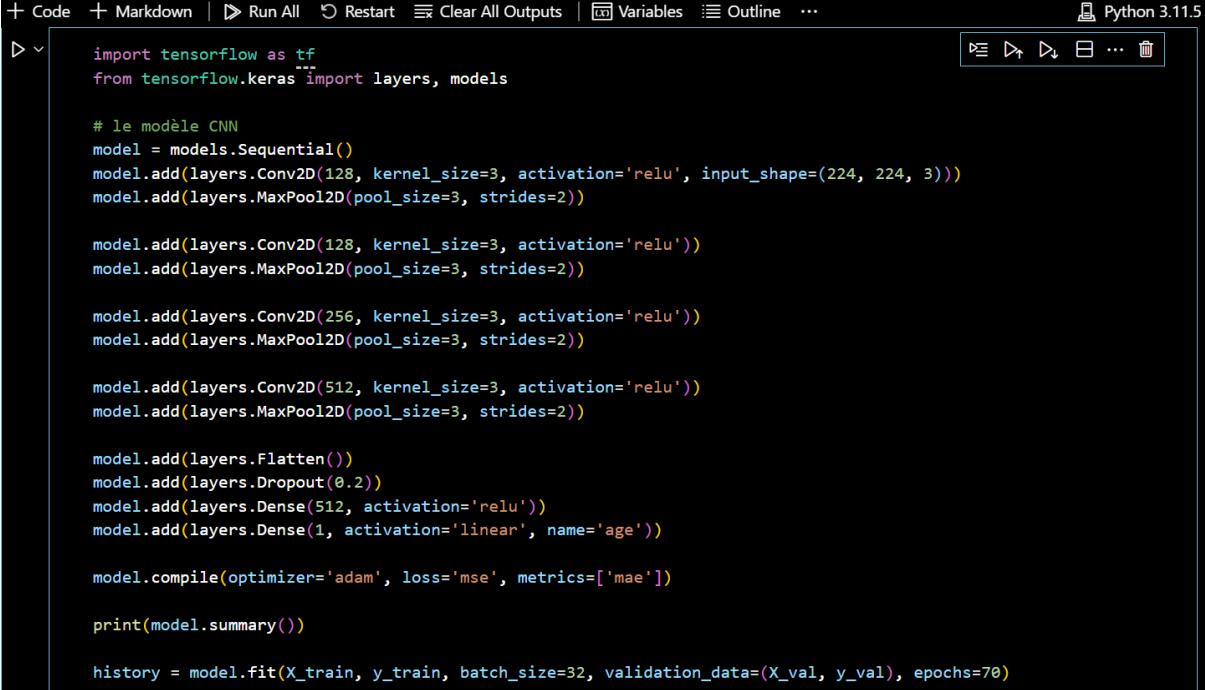
Après la phase d'apprentissage des caractéristiques au travers de ses nombreuses couches, l'architecture d'un réseau neuronal convolutif change et passe à la classification.

L'avant-dernière couche, entièrement connectée, génère un vecteur de  $K$  dimensions, où  $K$  est le nombre de classes que le réseau sera capable de prédire. Ce vecteur contient les probabilités d'appartenance d'une image à chacune des classes.

La dernière couche de l'architecture du CNN utilise une couche de classification pour générer la sortie finale de la classification.

### ➤ Implémentation d'une architecture CNN Personnalisé:

Dans le cadre de notre étude comparative sur l'estimation de l'âge facial, nous examinons diverses architectures de modèles d'apprentissage en profondeur. L'une de ces architectures inclut un modèle CNN personnalisé, dont la configuration et l'initialisation sont détaillées ci-dessous.



```
+ Code + Markdown | ▶ Run All ⏪ Restart ⏷ Clear All Outputs | ⏷ Variables ⏷ Outline ... Python 3.11.5  
▶  
import tensorflow as tf  
from tensorflow.keras import layers, models  
  
# le modèle CNN  
model = models.Sequential()  
model.add(layers.Conv2D(128, kernel_size=3, activation='relu', input_shape=(224, 224, 3)))  
model.add(layers.MaxPool2D(pool_size=3, strides=2))  
  
model.add(layers.Conv2D(128, kernel_size=3, activation='relu'))  
model.add(layers.MaxPool2D(pool_size=3, strides=2))  
  
model.add(layers.Conv2D(256, kernel_size=3, activation='relu'))  
model.add(layers.MaxPool2D(pool_size=3, strides=2))  
  
model.add(layers.Conv2D(512, kernel_size=3, activation='relu'))  
model.add(layers.MaxPool2D(pool_size=3, strides=2))  
  
model.add(layers.Flatten())  
model.add(layers.Dropout(0.2))  
model.add(layers.Dense(512, activation='relu'))  
model.add(layers.Dense(1, activation='linear', name='age'))  
  
model.compile(optimizer='adam', loss='mse', metrics=['mae'])  
  
print(model.summary())  
  
history = model.fit(X_train, y_train, batch_size=32, validation_data=(X_val, y_val), epochs=70)
```

## 4. Paramètres d'entraînement et d'évaluation :

Les paramètres d'entraînement et d'évaluation jouent un rôle crucial dans le développement et l'évaluation des modèles d'apprentissage en profondeur. Cette section détaille les paramètres spécifiques que nous avons choisis pour entraîner et évaluer nos modèles, y compris le modèle CNN personnalisé mentionné précédemment.

### ➤ Paramètres d'Entraînement :

- **Optimiseur** : Nous utilisons l'optimiseur Adam, qui adapte les taux d'apprentissage individuels pour chaque paramètre du modèle, facilitant ainsi la convergence rapide.
- **Fonction de Perte** : La fonction de perte est définie comme l'erreur quadratique moyenne (Mean Squared Error, MSE) dans le cas de l'estimation de l'âge facial. Cette fonction mesure la différence entre les valeurs prédites et les véritables valeurs d'âge, favorisant ainsi la convergence vers des prédictions précises.
- **Métrique d'Évaluation** : Pour évaluer la performance du modèle pendant l'entraînement, nous utilisons la Mean Absolute Error (MAE) ou erreur absolue moyenne. Elle mesure la moyenne des écarts absolus entre les prédictions du modèle et les véritables valeurs d'âge, fournissant une indication de la précision de la prédiction.

- **Nombre d'Époques** : Le nombre d'époques représente le nombre de fois que l'ensemble de données complet est parcouru pendant l'entraînement. Il est défini de manière à atteindre une convergence stable tout en évitant le surajustement.

## ➤ Paramètres d'Évaluation :

- **Ensembles de Données de Validation et de Test** : Le dataset UTKFace est divisé en ensembles d'entraînement, de validation et de test. L'ensemble de validation est utilisé pour ajuster les hyperparamètres et éviter le surajustement, tandis que l'ensemble de test est réservé pour évaluer la performance finale du modèle.
- **Métriques d'Évaluation** : Les performances du modèle sont évaluées à l'aide de différentes métriques telles que l'erreur absolue moyenne (MAE), l'erreur quadratique moyenne (MSE), et d'autres métriques pertinentes en fonction de la nature spécifique de la tâche d'estimation de l'âge.
- **Visualisation des Résultats** : Les résultats sont également visualisés à l'aide de courbes d'apprentissage, montrant l'évolution des métriques d'évaluation au fil des époques, et d'autres outils de visualisation pour mieux comprendre la performance du modèle.

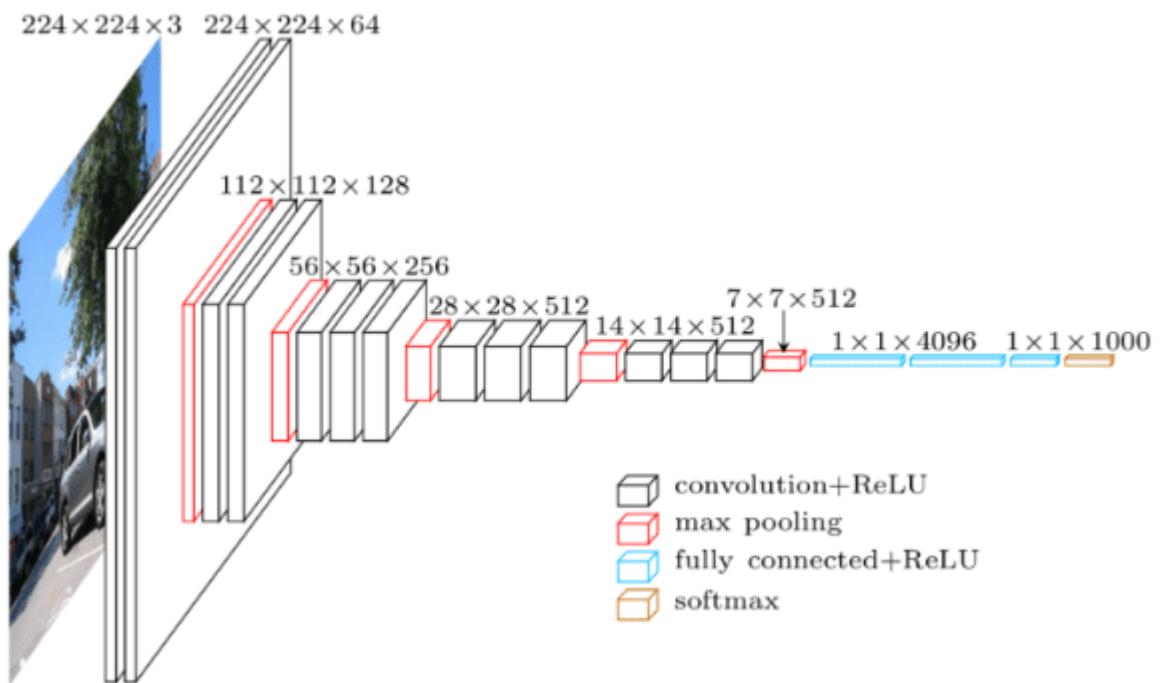
Ces paramètres ont été soigneusement choisis pour garantir une formation efficace et une évaluation robuste de nos modèles dans le contexte de l'estimation de l'âge facial en utilisant le dataset UTKFace. Les ajustements seront effectués en fonction des résultats préliminaires pour optimiser la performance des modèles.

# Architecture des Modèles:

## 1. Présentation détaillée de chaque modèle:

### 1.1 VGG16 (Visual Geometry Group 16)

VGG16, abréviation de Visual Geometry Group 16, est une architecture de réseau neuronal convolutif (CNN) utilisée dans le domaine de la vision par ordinateur. Développée par le groupe Visual Geometry Group de l'Université d'Oxford, cette architecture se distingue par sa profondeur et sa simplicité conceptuelle. La structure de VGG16 se compose de 16 couches, comprenant 13 couches de convolution suivies de trois couches de regroupement et 3 couches entièrement connectées.

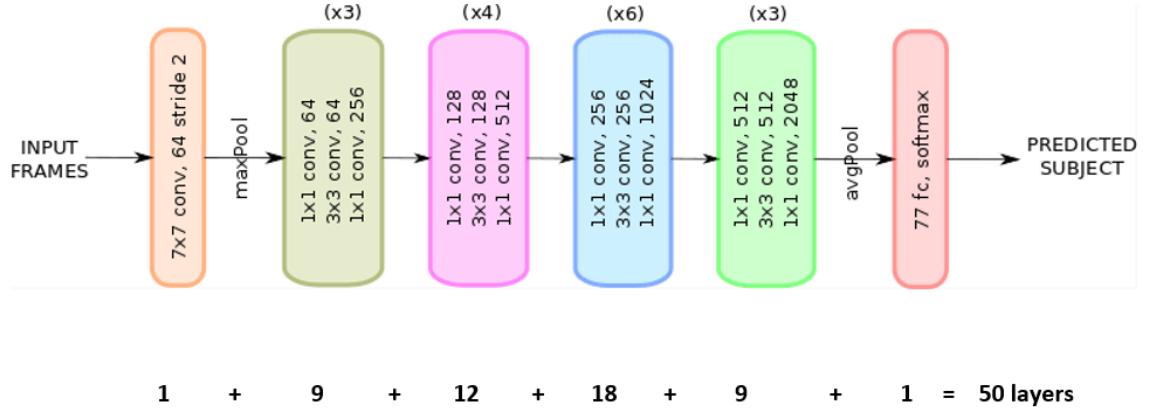


Une caractéristique distinctive de VGG16 est l'utilisation exclusive de filtres de petite taille ( $3 \times 3$ ) avec une fonction d'activation ReLU sur l'ensemble des couches de convolution. Cette approche permet au modèle d'apprendre des caractéristiques complexes en utilisant des filtres plus petits, favorisant ainsi la capture de motifs détaillés dans les images.

### 1.2 ResNet50 (Residual Network)

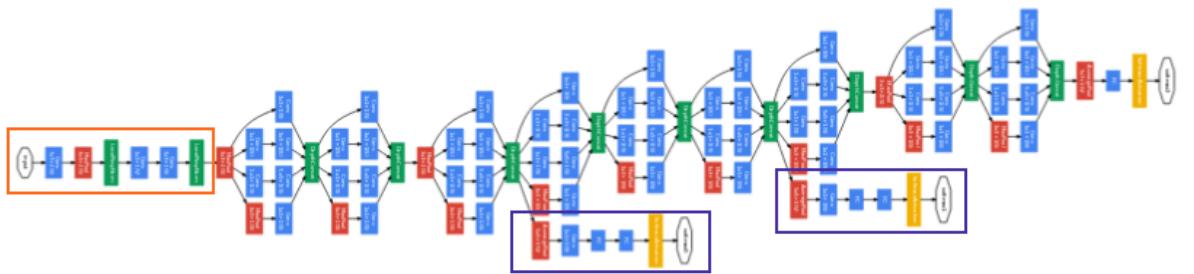
ResNet50, ou Residual Network avec 50 couches, représente une architecture de réseau neuronal convolutif (CNN) qui a révolutionné l'entraînement de réseaux profonds. Introduit en 2015 par Kaiming He et ses collègues, ResNet50 est célèbre pour son utilisation novatrice de blocs résiduels. Ces blocs permettent de résoudre le problème de la disparition du gradient en introduisant des connexions de dérivation,

facilitant ainsi la propagation efficace des informations à travers le réseau, même avec une profondeur considérable. Avec ses 50 couches, chaque bloc résiduel intègre des couches de convolution suivies d'une connexion de dérivation ajoutant l'entrée d'origine à la sortie du bloc.



### 1.3 InceptionV3

InceptionV3 est une architecture de réseau neuronal convolutif (CNN) développée par Google, appartenant à la famille des modèles Inception. Introduite en 2015, elle se distingue par l'utilisation de modules Inception qui exploitent des filtres de convolution de différentes tailles en parallèle, ainsi que des opérations de pooling, pour capturer des caractéristiques à des échelles variées. Conçu pour la classification d'images et la reconnaissance d'objets, InceptionV3 maintient une profondeur significative tout en optimisant le nombre de paramètres, offrant ainsi une efficacité computationnelle.



## 2. Architectures de Modèles pour l'Estimation de l'Âge

Dans cette section, nous détaillerons les architectures finales des modèles que nous avons adaptés pour notre étude sur l'estimation de l'âge à partir d'images faciales. Nous nous concentrerons spécifiquement sur les versions modifiées de VGG16, ResNet50 et InceptionV3, en mettant en lumière les ajustements que nous avons apportés à chaque architecture pour optimiser leur performance dans le contexte de notre tâche spécifique.

## 1. Architecture du modèle VGG16 utilisée:

a - Chargement du Modèle VGG16 Pré-entraîné :

```
# Charger le modèle VGG16 pré-entraîné (poids ImageNet)
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
```

Création du modèle VGG16 pré-entraîné avec des poids provenant d'ImageNet. Les couches supérieures (fully connected) sont exclues, et la forme d'entrée est spécifiée comme (224, 224, 3).

b - Réglage de la Trainabilité des Couches:

```
for layer in base_model.layers:
    layer.trainable = False
```

Les couches du modèle VGG16 sont gelées (non entraînables) pour conserver les poids pré-entraînés.

c - Ajout des Couches Personnalisées :

```
x = base_model.output
x = layers.Flatten()(x)
x = layers.Dense(128, activation='relu')(x)
output = layers.Dense(1, activation='linear')(x)
```

Des couches supplémentaires sont ajoutées au modèle pour personnaliser la prédiction de l'âge. Une couche 'aplatissement' est introduite pour convertir les données en un vecteur unidimensionnel, suivie d'une couche dense avec activation ReLU. Enfin, une couche dense finale avec activation linéaire est ajoutée pour la prédiction de l'âge.

d - Création du Modèle Complet :

```
model = models.Model(inputs=base_model.input, outputs=output)
```

Un modèle complet est créé en spécifiant les entrées (celles du modèle VGG16) et les sorties (la couche de prédiction de l'âge).

e - Compilation du Modèle :

```
model.compile(optimizer='adam', loss='mean_squared_error', metrics=['mae'])
```

Le modèle est compilé avec l'optimiseur Adam, une fonction de perte MSE (Mean Squared Error) adaptée à la régression, et la métrique MAE (Mean Absolute Error) pour évaluer la performance.

f - Entraînement du Modèle:

```
history = model.fit(X_train, y_train, batch_size=32, validation_data=(X_val, y_val), epochs=70)
```

Ce code entraîne le modèle sur les données d'entraînement (X\_train, y\_train) pendant 70 epochs. Un batch de 32 échantillons est utilisé à chaque mise à jour des poids. Les performances sont également évaluées sur les données de validation (X\_val, y\_val).

## 2. Architecture du modèle ResNet50 utilisée:

a - Chargement du Modèle ResNet50 Pré-entraîné :

```
base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
```

Cette ligne charge le modèle ResNet50 pré-entraîné avec des poids provenant d'ImageNet. La couche supérieure (la couche de classification ImageNet) est exclue (`include_top=False`) car nous allons ajouter nos propres couches personnalisées pour la prédiction de l'âge.

b - Réglage de la Trainabilité des Couches:

```
for layer in base_model.layers[:-10]:  
    layer.trainable = True
```

Cette boucle rend les dix dernières couches du modèle ResNet50 pré-entraîné entraînables. Cela permet au modèle de s'adapter aux caractéristiques spécifiques de notre tâche d'estimation de l'âge.

c - Ajout des Couches Personnalisées :

```
x = base_model.output
x = layers.GlobalAveragePooling2D()(x)
x = layers.Dense(128, activation='relu')(x)
x = layers.Dropout(0.5)(x)
x = layers.Dense(64, activation='relu')(x)
x = layers.Dropout(0.5)(x)
output = layers.Dense(1, activation='linear')(x)
```

Ces lignes ajoutent des couches personnalisées au-dessus de la sortie de ResNet50. On utilise une moyenne globale pour aplatiser les activations, suivi de deux couches denses avec des fonctions d'activation ReLU, des couches de dropout pour réduire le surajustement, et enfin une couche dense avec une activation linéaire pour effectuer une régression d'âge.

d - Création du Modèle Complet :

```
model = models.Model(inputs=base_model.input, outputs=output)
```

Cette ligne crée le modèle complet en spécifiant les entrées du modèle ResNet50 pré-entraîné comme entrées et les sorties de nos couches personnalisées comme sorties.

e - Compilation du Modèle :

```
model.compile(optimizer='adam', loss='mean_squared_error', metrics=['mae'])
```

Le modèle est compilé avec l'optimiseur Adam, une fonction de perte MSE (Mean Squared Error) adaptée à la régression, et la métrique MAE (Mean Absolute Error) pour évaluer la performance.

f - Entraînement du Modèle:

```
history = model.fit(X_train, y_train, batch_size=32, validation_data=(X_val, y_val), epochs=70)
```

Ce code entraîne le modèle sur les données d'entraînement (X\_train, y\_train) pendant 70 epochs. Un batch de 32 échantillons est utilisé à chaque mise à jour des poids. Les performances sont également évaluées sur les données de validation (X\_val, y\_val).

### 3. Architecture du modèle InceptionV3 utilisée:

a - Chargement du Modèle InceptionV3 pré-entraîné :

```
base_model = InceptionV3(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
```

Cela charge le modèle InceptionV3 pré-entraîné avec des poids provenant d'ImageNet, excluant la couche fully connected supérieure (include\_top=False).

b - Fixation des Paramètres du Modèle de Base :

```
for layer in base_model.layers:  
    layer.trainable = True
```

Cette boucle rend toutes les couches du modèle de base InceptionV3 entraînables. Cela signifie que lors de l'entraînement, les poids de ces couches peuvent être ajustés pour s'adapter à la tâche spécifique.

c - Création du Modèle Séquentiel :

```
model = models.Sequential()
```

On initialise un modèle séquentiel pour empiler les différentes couches.

d - Ajout de Couches Supplémentaires pour la Détection de l'Âge :

```
# Ajouter InceptionV3 comme couche de base  
model.add(base_model)  
# Ajouter des couches supplémentaires pour la détection de l'âge  
model.add(layers.GlobalAveragePooling2D())  
model.add(layers.Dense(512, activation='relu'))  
model.add(layers.Dropout(0.5))  
model.add(layers.Dense(1, activation='linear', name='age'))
```

Des couches supplémentaires sont ajoutées, notamment une couche de pooling global, une couche dense avec activation ReLU, une couche de dropout pour réduire le surajustement, et une couche dense avec activation linéaire pour la régression d'âge.

e - Compilation du Modèle :

```
model.compile(optimizer='adam', loss='mse', metrics=['mae'])
```

Le modèle est compilé avec l'optimiseur Adam, une fonction de perte MSE (Mean Squared Error) adaptée à la régression, et la métrique MAE (Mean Absolute Error) pour évaluer la performance.

f - Entraînement du Modèle:

```
history = model.fit(X_train, y_train, batch_size=32, validation_data=(X_val, y_val), epochs=70, shuffle = True)
```

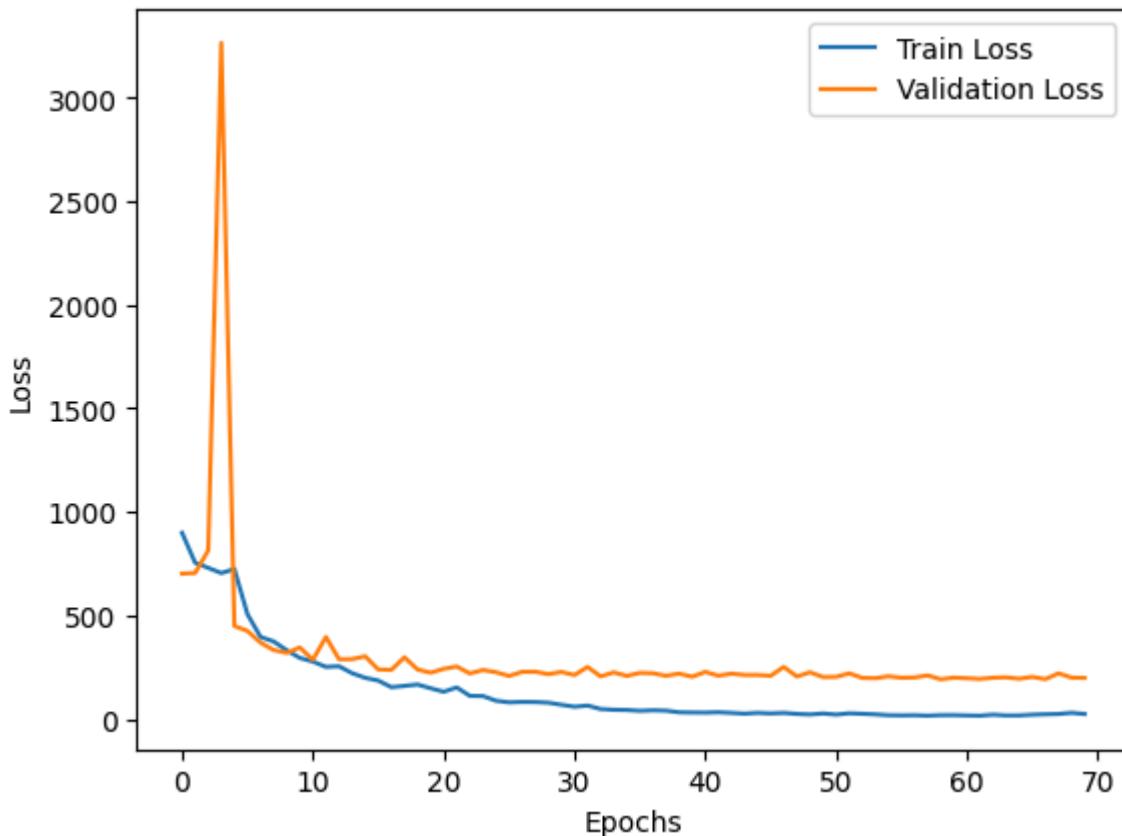
Cette ligne de code exécute la phase d'entraînement du modèle ,elle utilise les données d'entraînement (X\_train et y\_train) et de validation (X\_val et y\_val) avec un

batch de 32 échantillons. Le modèle est entraîné pendant 70 epochs, et les données d'entraînement sont mélangées à chaque epoch pour améliorer la généralisation.

# Visualisation et Prédictions

## 1. Interprétation des Résultats

### 1.1. Modèle CNN Personnalisé :



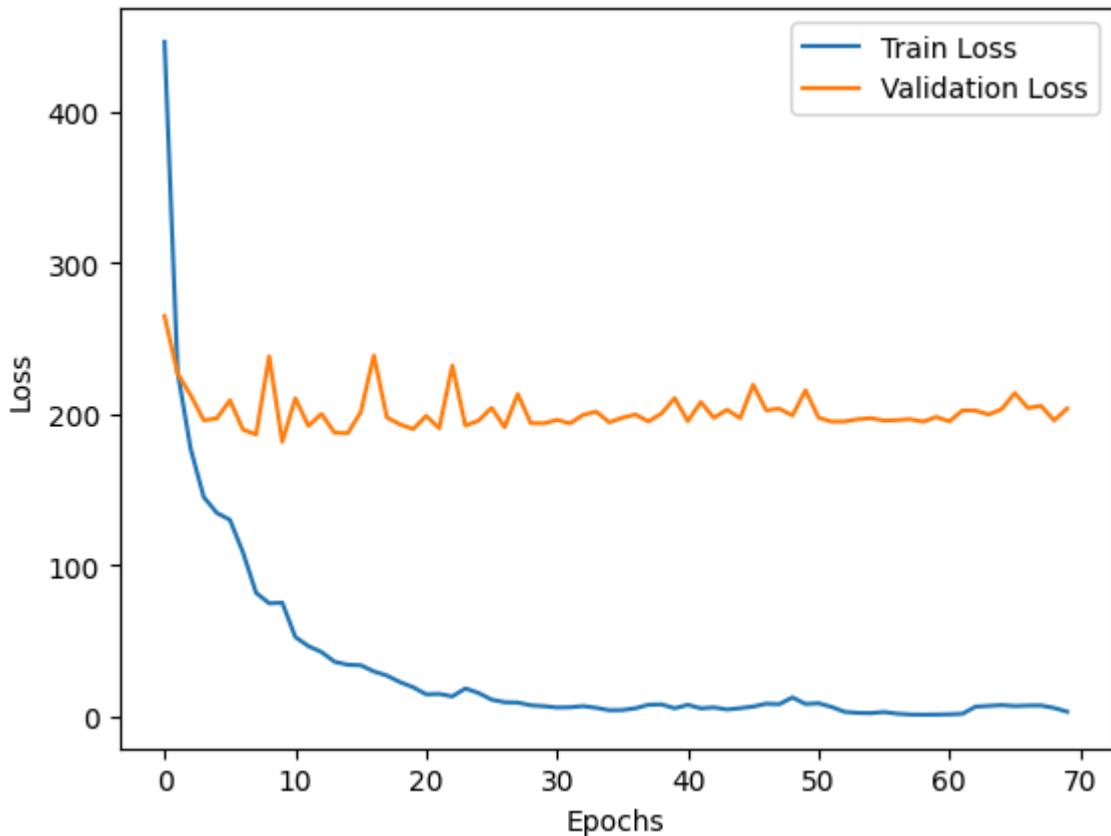
#### ➤ Perte d'entraînement :

La perte d'entraînement diminue rapidement au début de l'entraînement, puis plus lentement au fur et à mesure que le modèle s'approche de la convergence.

#### ➤ Perte de validation :

La perte de validation diminue également au début de l'entraînement, mais elle commence à augmenter après environ 50 époques.

### 1.2. Modèle VGG16 :



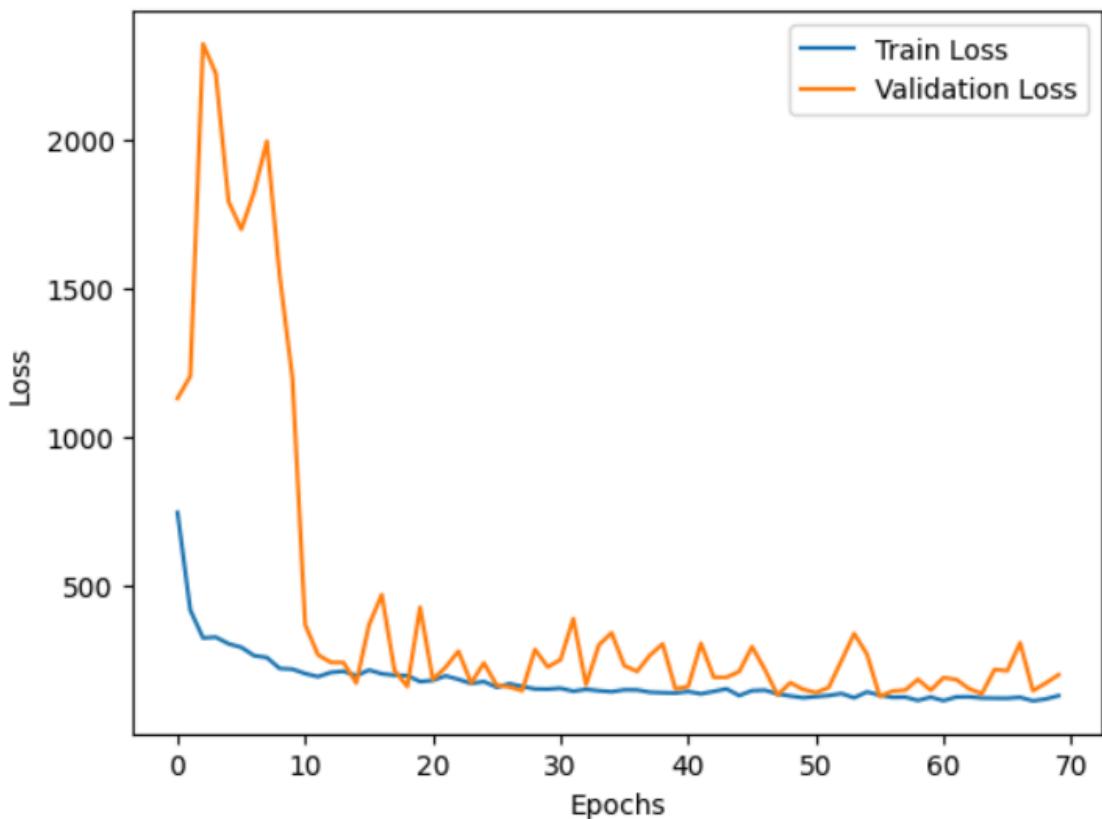
➤ **Perte d'entraînement :**

La perte d'entraînement diminue rapidement au début de l'entraînement, car le modèle apprend rapidement à identifier les caractéristiques qui distinguent les images de personnes de différents âges. La perte d'entraînement continue de diminuer à un rythme plus lent au fur et à mesure que le modèle apprend de plus en plus de données.

➤ **Perte de validation :**

La perte de validation suit un schéma similaire à la perte d'entraînement, mais elle reste légèrement supérieure. Cela indique que le modèle est capable de prédire correctement les âges des images d'entraînement, mais qu'il a encore du mal à généraliser à de nouvelles images.

### 1.3. Modèle ResNet50:



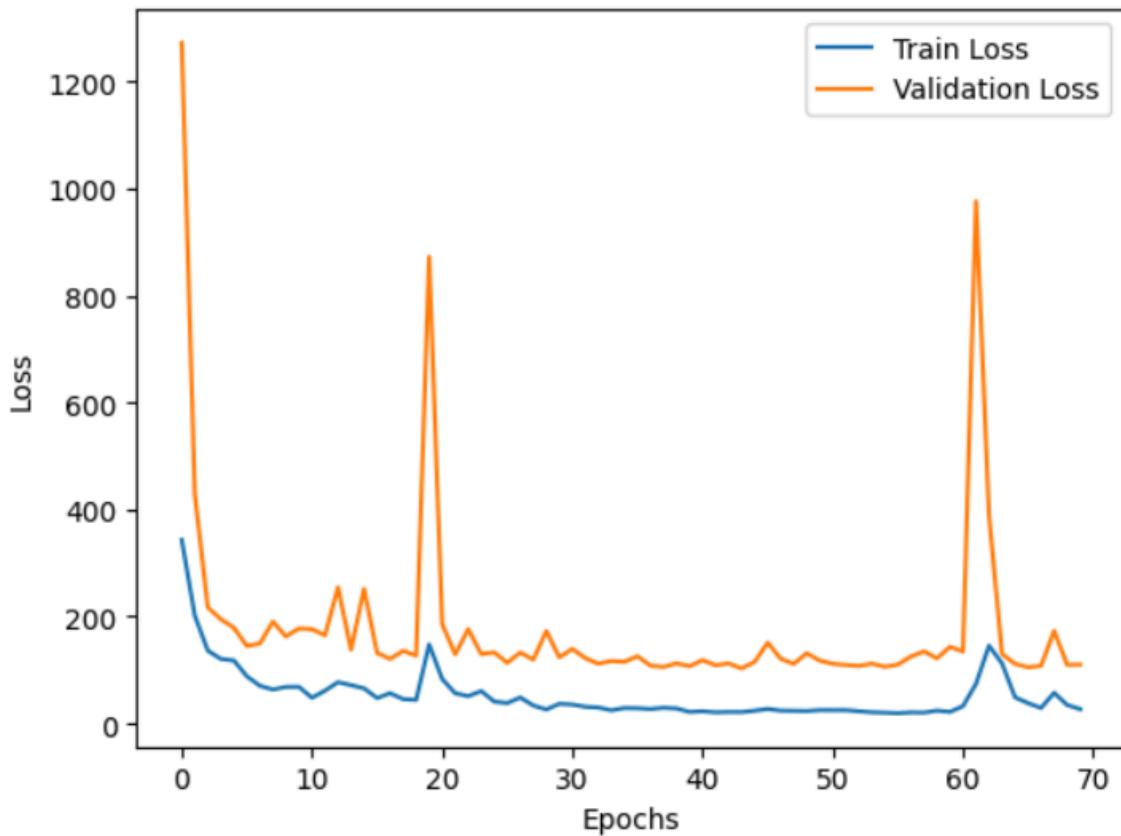
#### ➤ Perte d'entraînement :

La perte d'entraînement diminue progressivement au fil des époques, ce qui est un bon signe. Cela signifie que le modèle s'améliore progressivement à la tâche d'estimation de l'âge à partir des données d'entraînement.

#### ➤ Perte de validation :

La perte de validation décroît au fil des époques, mais elle le fait à un rythme plus modéré par rapport à la diminution de la perte d'entraînement. Cependant, elle présente des fluctuations, augmentant légèrement avant de retrouver sa trajectoire descendante, et ce schéma se répète plusieurs fois.

## 1.4. InceptionV3:



➤ **Perte d'entraînement :**

La perte d'entraînement diminue progressivement au fil des époques. Cela indique que le modèle apprend à prédire l'âge des visages avec plus de précision.

➤ **Perte de validation :**

La perte de validation suit une tendance similaire à la perte d'entraînement. Cela suggère que le modèle ne surapprend pas.

## 2. Visualisation des prédictions:

- **Modèle CNN Personnalisé :**

16/16 [=====] - 1s 45ms/step

Vrais Âges vs Prédictions d'Âge :

```
[[ 34.          25.35480118]
 [ 52.          65.89985657]
 [ 58.          48.39609528]
 [ 27.          29.84371185]
 [ 40.          33.86074066]
 [ 13.          19.89254189]
 [ 64.          53.46496582]
 [ 72.          67.5578537 ]
 [ 31.          19.34771538]
 [ 84.          85.73146057]
 [ 60.          59.90656281]
 [ 52.          33.02290344]
 [  6.          5.60978413]
 [ 54.          51.28156281]
 [ 33.          29.442173  ]
 [ 93.          49.65916061]
 [ 48.          34.57241058]
 [ 15.          28.34681702]
 [  7.          7.37051582]
 [ 25.          16.63422203]
 [  4.          7.28836346]
 [ 32.          23.2047081 ]
```

True Age: 76, Predicted Age: 67.93



- Modèle VGG16 :

```
16/16 [=====] - 2s 131ms/step
Vrais Âges vs Prédictions d'Âge :
[[ 3.4000000e+01  3.31643906e+01]
 [ 5.2000000e+01  6.67519531e+01]
 [ 5.8000000e+01  6.21862259e+01]
 [ 2.7000000e+01  4.25602188e+01]
 [ 4.0000000e+01  3.86187630e+01]
 [ 1.3000000e+01  7.43743181e+00]
 [ 6.4000000e+01  5.20920753e+01]
 [ 7.2000000e+01  7.48065567e+01]
 [ 3.1000000e+01  2.45490723e+01]
 [ 8.4000000e+01  5.68309860e+01]
 [ 6.0000000e+01  7.91246338e+01]
 [ 5.2000000e+01  5.10110283e+01]
 [ 6.0000000e+00  9.80651474e+00]
 [ 5.4000000e+01  3.42391357e+01]
 [ 3.3000000e+01  4.15195007e+01]
 [ 9.3000000e+01  5.46420403e+01]
 [ 4.8000000e+01  3.26060104e+01]
 [ 1.5000000e+01  4.14370689e+01]
 [ 7.0000000e+00  5.35588264e+00]
 [ 2.5000000e+01  2.24488697e+01]]
```

True Age: 76, Predicted Age: 44.52



- Modèle ResNet50:

```
16/16 [=====] - 3s 95ms/step
Vrais Âges vs Prédictions d'Âge :
[[ 34.          22.24396133]
 [ 52.          49.92613602]
 [ 58.          45.58759308]
 [ 27.          28.8026619 ]
 [ 40.          26.98794937]
 [ 13.          9.06828022]
 [ 64.          51.51537704]
 [ 72.          60.95429993]
 [ 31.          21.08222961]
 [ 84.          65.07866669]
 [ 60.          52.78307724]
 [ 52.          32.98615646]
 [  6.          6.65705109]
 [ 54.          42.54462814]
 [ 33.          36.33716202]
 [ 93.          61.14270782]
 [ 48.          59.12644196]
 [ 15.          19.37716293]
 [  7.          8.77554893]
 [ 25.          16.40270805]
 [  4.          9.06050873]
 [ 32.          25.00630188]
 [ 12.          16.19720078]
 [ 52.          36.08439636]
 [ 59.          37.94009781]
 [ 53.          49.99523544]]
```

True Age: 76, Predicted Age: 60.85



## ● InceptionV3:

16/16 [=====] - 2s 69ms/step  
Vrais Âges vs Prédictions d'Âge :

[ [ 34.	25.822155 ]
[ 52.	50.64526367]
[ 58.	57.02012634]
[ 27.	24.83224869]
[ 40.	45.56434631]
[ 13.	6.34078646]
[ 64.	59.99066544]
[ 72.	71.61694336]
[ 31.	27.19112968]
[ 84.	92.46396637]
[ 60.	62.69880295]
[ 52.	33.65551758]
[ 6.	5.21680927]
[ 54.	48.78073883]
[ 33.	38.49973679]
[ 93.	81.90665436]
[ 48.	61.72910309]
[ 15.	20.25441933]
[ 7.	5.16622591]
[ 25.	25.19902039]
[ 4.	8.16920471]
[ 32.	19.96526718]
[ 12.	32.45549011]

True Age: 76, Predicted Age: 69.49



# Benchmarking

La phase de benchmarking est très importante pour évaluer et comparer les performances des modèles. Cela peut inclure l'utilisation de métriques telles que l'erreur absolue moyenne (MAE) et l'erreur quadratique moyenne (MSE), ainsi que des considérations sur la vitesse d'inférence. Une analyse comparative des performances entre les modèles doit être présentée de manière structurée.

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import numpy as np

# Calcul des métriques
mae = mean_absolute_error(y_test, predictions.flatten())
mse = mean_squared_error(y_test, predictions.flatten())
rmse = np.sqrt(mse)
r2 = r2_score(y_test, predictions.flatten())

# Affichage des résultats
print(f'MAE: {mae:.2f}')
print(f'MSE: {mse:.2f}')
print(f'RMSE: {rmse:.2f}')
print(f'R^2: {r2:.2f}'
```

Métriques	Modèle CNN Personnalisé	Modèle VGG16	Modèle ResNet50	Modèle InceptionV3
MAE	9.58	10.05	9.81	6.63
MSE	179.37	188.64	174.54	86.42
RMSE	13.39	13.73	13.21	9.30
R^2	0.74	0.73	0.75	0.88
Accuracy	62.53%	60.25%	59.84%	77.66%

- Le modèle **InceptionV3** affiche la meilleure performance en termes de MAE, MSE, RMSE, R^2 et Accuracy, indiquant une précision supérieure dans la prédiction de l'âge par rapport aux autres modèles.
- Le modèle **ResNet50** présente également des performances solides, avec un MAE légèrement inférieur à celui du modèle VGG16 et du modèle CNN personnalisé.

- Les performances comparables entre le modèle **VGG16** et le modèle **CNN personnalisé** suggèrent que la conception personnalisée a atteint des résultats similaires sans être significativement inférieure à l'architecture pré-entraînée.
- L'accuracy élevée du modèle **InceptionV3** (77.66%) souligne sa capacité à bien classifier les groupes d'âge.

En conclusion, le choix de l'architecture du modèle a un impact significatif sur les performances de la détection d'âge, avec InceptionV3 se démarquant comme le meilleur modèle dans cette étude comparative.

## **Conclusion:**

Ce projet axé sur l'estimation faciale de l'âge à l'aide de modèles d'apprentissage en profondeur a fourni des perspectives importantes dans le domaine de la vision par ordinateur. En examinant les différents modèles, à savoir VGG16, ResNet50, InceptionV3, et un modèle CNN personnalisé, nous avons pu évaluer leur performance respective dans la tâche d'estimation d'âge.

La contextualisation du projet a souligné l'importance de l'estimation de l'âge dans divers secteurs, tout en mettant en lumière les défis liés à la précision de cette estimation. La méthodologie détaillée a décrit le processus de collecte et de prétraitement des données, ainsi que le choix des modèles d'apprentissage en profondeur.

Les résultats obtenus ont révélé que le modèle InceptionV3 a surpassé les autres, démontrant une précision remarquable dans la prédiction de l'âge. Cependant, il est intéressant de noter que le modèle CNN personnalisé a également présenté des performances compétitives, montrant la pertinence de l'approche sur mesure.

L'architecture des modèles a été examinée en détail, mettant en évidence les spécificités de VGG16, ResNet50, et InceptionV3. Les paramètres d'entraînement et d'évaluation ont été ajustés pour optimiser les performances des modèles.

En conclusion, ce projet a contribué à l'avancement de la compréhension des méthodes d'estimation faciale de l'âge en utilisant des modèles d'apprentissage en profondeur. Les résultats obtenus fournissent des indications précieuses pour guider le choix d'architectures dans des applications futures de vision par ordinateur axées sur l'estimation de l'âge facial. Ce travail ouvre également la porte à des recherches plus approfondies sur les améliorations potentielles des modèles et des techniques de prétraitement des données pour des résultats encore plus précis.

# Bibliographie

<https://susangq.github.io/UTKFace/>

<https://ch.mathworks.com/fr/discovery/convolutional-neural-network-matlab.html>

<https://www.google.com/url?sa=i&url=https%3A%2F%2Fsrsapireddy.medium.com%2Fresnet-50-introduction-b5435fdbaa66f&psig=AOvVaw1BXzuFaFGFs2XQm-Agg2yp&ust=1702861117998000&source=images&cd=vfe&opi=89978449&ved=0CBIQjRxqFwoTCODrgZaiYMDFQAAAAAdAAAAABAI>

[https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.researchgate.net%2Ffigure%2FVGG16-architecture-16\\_fig2\\_321829624&psig=AOvVaw2BIwUkl-eSn00MqPQGmQCq&ust=1702859321547000&source=images&cd=vfe&opi=89978449&ved=0CBIQjRxqFwoTCLipmr6bIYMDFQAAAAAdAAAAABAD](https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.researchgate.net%2Ffigure%2FVGG16-architecture-16_fig2_321829624&psig=AOvVaw2BIwUkl-eSn00MqPQGmQCq&ust=1702859321547000&source=images&cd=vfe&opi=89978449&ved=0CBIQjRxqFwoTCLipmr6bIYMDFQAAAAAdAAAAABAD)

[https://www.google.com/url?sa=i&url=https%3A%2F%2Ftowardsdatascience.com%2Fa-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202&psig=AOvVaw2bCAp7ih9aGeGwNpygE\\_tL&ust=1702862150821000&source=images&cd=vfe&opi=89978449&ved=0CBIQjRxqFwoTCODxkYSmlYMDFQAAAAAdAAAAABAD](https://www.google.com/url?sa=i&url=https%3A%2F%2Ftowardsdatascience.com%2Fa-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202&psig=AOvVaw2bCAp7ih9aGeGwNpygE_tL&ust=1702862150821000&source=images&cd=vfe&opi=89978449&ved=0CBIQjRxqFwoTCODxkYSmlYMDFQAAAAAdAAAAABAD)