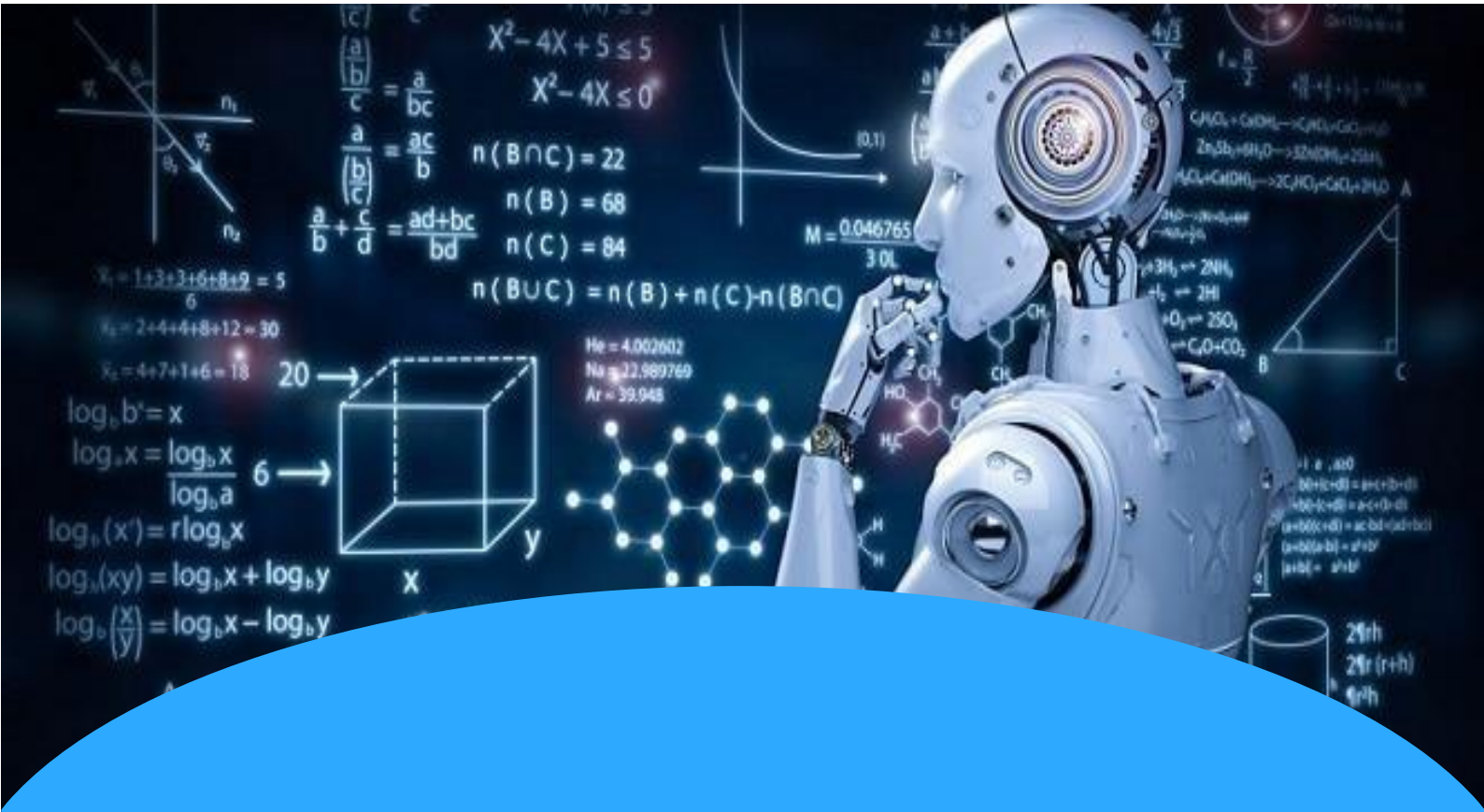
Angelo Fernandez-Vasquez 5A

Deep learning : classification de visage



Classification de visage

Projet

esiea
INGÉNIEUR·E·S D'UN NUMÉRIQUE UTILE

Deep learning : classification de visage – projet

Sommaire :

Deep learning : classification de visage – projet

Question 1

L'architecture choisie pour ce problème de classification binaire est un réseau de neurones convolutifs (CNN). Cette architecture est idéale pour traiter les images, car les CNN sont spécialement conçus pour identifier et exploiter des motifs spatiaux dans des données en grille, comme c'est le cas avec les images. Dans notre cas, il s'agit d'éléments « softbios » qui peut être la couleur des cheveux, le port de la barbe, couleurs des yeux etc..

Voici l'architecture :

```
# Architecture du modèle CNN
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(80, 80, 3)),
    MaxPooling2D(2, 2),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
    Flatten(),
    Dense(512, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')  # Sortie binaire
])
```

Tout d'abord le réseau commence avec des couches convolutives utilisant un nombre croissant de filtres (32, 64, 128), permettant au réseau d'apprendre des caractéristiques allant du simple au complexe de manière hiérarchique. Les filtres de taille 3x3 sont choisis pour leur efficacité computationnelle et leur capacité à capturer des détails pertinents. La fonction d'activation ReLU est utilisée pour sa simplicité et pour minimiser le problème de disparition des gradients.

Par la suite, les couches de pooling 2x2 jouent un rôle crucial en réduisant les dimensions des données, ce qui contribue à limiter le nombre de paramètres et à minimiser le risque de surapprentissage.

Ensuite, l'emploi d'une couche dense de 512 neurones permet de traiter adéquatement les caractéristiques complexes tout en évitant le surapprentissage.

Enfin, une couche de sortie composée d'un seul neurone avec une activation sigmoid est optimale pour la classification binaire, offrant une probabilité distincte pour chaque prédiction.

Ensemble, ces composants créent un équilibre entre la capacité d'apprentissage, la généralisation et l'efficacité computationnelle du modèle.

Deep learning : classification de visage – projet

Question 2

L'architecture de notre réseau de neurones convolutifs contient trois couches convolutives, un choix simple mais qui équilibre efficacement plusieurs facteurs clés.

Premièrement, ces trois couches permettent une extraction hiérarchique des caractéristiques des visages, allant des éléments basiques aux détails plus complexes.

Deuxièmement, cette profondeur modérée du réseau aide à capturer des représentations riches des données tout en évitant les problèmes liés aux réseaux plus profonds, comme le surapprentissage et les défis d'entraînement. Ces défis incluent la gestion des gradients, où un nombre modéré de couches aide à prévenir les problèmes de disparition ou d'explosion des gradients, qui sont plus fréquents dans les architectures plus profondes.

Troisièmement, trois couches convolutives maintiennent l'efficacité computationnelle du modèle, important pour la gestion des ressources et le déploiement pratique.

Enfin, en limitant le modèle à trois couches convolutives, on minimise également le risque de surapprentissage, où le modèle devient trop spécifique aux données d'entraînement, et on assure une complexité computationnelle gérable, facilitant ainsi l'entraînement et le déploiement du modèle dans des environnements avec des contraintes de ressources.

Deep learning : classification de visage – projet

Question 3

Pour prévenir l'over-fiting du modèle, plusieurs stratégies ont étés adoptées.

Premièrement, le dataset a été divisé en ensembles distincts d'entraînement, de validation et de test (respectivement 80%, 20%, 20%), une étape essentielle pour assurer la fiabilité de notre modèle. Cette répartition permet d'évaluer la capacité de généralisation du modèle et de prévenir le surapprentissage. L'ensemble de validation est particulièrement important pour ajuster les hyperparamètres et détecter le surapprentissage.

```
]: from sklearn.model_selection import train_test_split

# Séparer les données en 2 ensembles : entraînement+validation et test
temp_df, test_df = train_test_split(df_balanced, test_size=0.2, stratify=df_balanced["gender"])

# Séparer entraînement+validation en 2 ensembles : entraînement et validation
train_df, val_df = train_test_split(temp_df, test_size=0.25, stratify=temp_df["gender"])  # 0.25 x 0.8 = 0.2
```

Deuxièmement, la technique d'early stopping a été utilisée avec une patience de 1 pour arrêter l'entraînement lorsque la performance sur l'ensemble de validation cesse de s'améliorer sur l'époch suivante, prévenant ainsi l'over-fitting.

```
# Entraînement du modèle
history = model.fit(
    train_generator,
    steps_per_epoch=len(train_generator),  # Nombre d'étapes par époque
    epochs=10,  # Nombre d'époques
    validation_data=val_generator,
    validation_steps=len(val_generator),
    callbacks=[early_stopping]
)
```

```
Epoch 1/10
2979/2979 [==============================] - 428s 143ms/step - loss: 0.0046 - accuracy: 0.8976 - val_loss: 0.0085 - val_accuracy: 0.8984
Epoch 2/10
2979/2979 [==============================] - 399s 134ms/step - loss: 0.0044 - accuracy: 0.9044 - val_loss: 0.0085 - val_accuracy: 0.9044
Epoch 3/10
2979/2979 [==============================] - 441s 148ms/step - loss: 0.0043 - accuracy: 0.9097 - val_loss: 0.0073 - val_accuracy: 0.9169
Epoch 4/10
2979/2979 [==============================] - 442s 148ms/step - loss: 0.0042 - accuracy: 0.9137 - val_loss: 0.0072 - val_accuracy: 0.9183
Epoch 5/10
2979/2979 [==============================] - 445s 149ms/step - loss: 0.0041 - accuracy: 0.9151 - val_loss: 0.0073 - val_accuracy: 0.9158
Epoch 5: early stopping
```

Deep learning : classification de visage – projet

Question 4

L'asymétrie de la base d'apprentissage est en effet un problème car elle peut conduire à des performances de classification déséquilibrées entre les genres, voici comment j'ai procédé :

Tout d'abord les données ont étés analysées, notamment l'équilibre sur les classes et les genres :

```
]: # vérifier l'équilibre
print("Equilibre des genres :", np.unique(df["gender"], return_counts=True)) # label de genre -1: femme 1: homme
print("Equilibre des classes :", np.unique(df["class"], return_counts=True)) # class à prédire

Equilibre des genres : (array([-1,  1], dtype=int64), array([113154,  79422], dtype=int64))
Equilibre des classes : (array(['0', '1'], dtype=object), array([155435,  37141], dtype=int64))
```

Les Femmes sont sur-représenté : 113154 contre 79422 pour les hommes La classe majoritaire est "0" et la classe minoritaire est "1" Cherchons d'abord à équilibrer la proportion homme/femme

Après avoir observé un déséquilibre dans la répartition des genres et dans le but d'optimiser la métrique d'équité entre les genres, j'ai opté pour un sous-échantillonnage des données du genre majoritaire (les femmes). Cette décision visait à équilibrer la représentation des genres dans notre dataset. Bien que cette méthode réduise le risque que le modèle se spécialise excessivement sur un genre particulier, elle entraîne aussi une perte d'informations. Néanmoins, j'ai privilégié cette approche pour sa simplicité. Voici l'équilibre des genres dans le dataset après cette opération :

```
]: # vérifier l'équilibre
print("Equilibre des genres :", np.unique(df_balanced["gender"], return_counts=True)) # label de genre -1: femme 1: homme
print("Equilibre des classes :", np.unique(df_balanced["class"], return_counts=True)) # class à prédire

Equilibre des genres : (array([-1,  1], dtype=int64), array([79422, 79422], dtype=int64))
Equilibre des classes : (array(['0', '1'], dtype=object), array([130088,  28756], dtype=int64))
```

On constate qu'il y a toujours un déséquilibre entre les classes à prédire : les labels « 0 » sont plus fréquents que les labels « 1 ». Cependant, il est important de rappeler que ces labels représentent des caractéristiques soft-bio, telles que la couleur des cheveux, dont certaines peuvent être naturellement plus rares que d'autres, à l'exemple de la couleur blonde. Ainsi, cette répartition pourrait en réalité refléter fidèlement la population du dataset original. Par conséquent, j'ai décidé de ne pas intervenir sur ce déséquilibre.

Deep learning : classification de visage – projet

Question 5

Oui le fait que les labels soient bruités peut poser un problème. Les labels bruités peuvent entraîner des difficultés dans l'apprentissage du modèle, car ils peuvent induire en erreur le modèle, conduisant à une mauvaise généralisation et à de mauvaise performances.

Pour remédier à cela, j'ai opté pour la Focal Loss, une version modifiée de la fonction de coût Binary Cross Entropy, spécialement conçue pour atténuer les effets des labels bruités et des déséquilibres de classes. Cette fonction de coût avancée modifie la contribution de chaque exemple à la perte totale, en mettant l'accent sur les exemples difficiles à classer.

La Focal Loss comporte deux paramètres clés : alpha et gamma. Alpha sert à équilibrer les contributions des différentes classes à la perte totale, aidant à éviter un biais en faveur de la classe majoritaire. En parallèle, gamma augmente la concentration sur les exemples où le modèle est incertain. Cette combinaison rend la Focal Loss particulièrement adaptée pour traiter les problèmes liés aux labels bruités et au déséquilibre des classes.

Dans mon implémentation, le paramètre gamma a été fixé sur une valeur élevé de 5.0, favorisant une focalisation accrue sur les exemples complexes, crucial en présence de labels bruités, et un alpha de 0.25 pour maintenir un équilibre entre l'impact des classes minoritaires et majoritaires.

```
# Créer une instance de BinaryFocalCrossentropy
focal_loss = tf.keras.losses.BinaryFocalCrossentropy(gamma=5.0, alpha=0.25)
```

```
# Compilation du modèle
model.compile(optimizer='adam', loss=focal_loss, metrics=['accuracy'])
```

Deep learning : classification de visage – projet

Question 6

Pour la base de test, les labels sont déterminés à partir des prédictions du modèle.

Les valeurs à la sortie du réseau sont des probabilités comprises entre 0 et 1, elles sont binarisées en utilisant un seuil de décision fixé à 0.5. Si une prédiction est supérieure à 0.5, le label attribué est 1; si elle est inférieure ou égale à 0.5, le label attribué est -1.

```
predictions = model.predict(pred_generator)
labels = np.where(predictions > 0.5, 1, -1)
pred["label"] = labels.flatten()
```

```
313/313 [==============================] - 11s 35ms/step
```

```
pred.head()
```

	filename	label
0	000000.jpg	-1
1	000001.jpg	-1
2	000002.jpg	1
3	000003.jpg	-1
4	000004.jpg	-1

Deep learning : classification de visage – projet

Question 7

Pour aborder les enjeux d'équité de l'algorithme, j'ai mis en place une pondération spécifique aux genres durant l'entraînement du modèle. Suite à des tests effectués sans pondération, il a été observé que le modèle présentait de meilleures performances sur les hommes que sur les femmes. Cette tendance est illustrée dans la capture d'écran ci-dessous :

```
# Évaluation sur les femmes
test_loss_women, test_accuracy_women = model.evaluate(test_generator_women)

# Évaluation sur les hommes
test_loss_men, test_accuracy_men = model.evaluate(test_generator_men)

# Imprimez ou enregistrez les résultats
print(f"Performance sur les Femmes - Loss: {test_loss_women}, Accuracy: {test_accuracy_women}")
print(f"Performance sur les Hommes - Loss: {test_loss_men}, Accuracy: {test_accuracy_men}")
```

```
497/497 [==============================] - 14s 27ms/step - loss: 0.2064 - accuracy: 0.9194
497/497 [==============================] - 14s 27ms/step - loss: 0.1055 - accuracy: 0.9682
Performance sur les Femmes - Loss: 0.20642319321632385, Accuracy: 0.9193578958511353
Performance sur les Hommes - Loss: 0.1055423766374588, Accuracy: 0.9682070016860962
```

:

```
Fair_Metric = np.mean([test_accuracy_women, test_accuracy_men]) - (2*np.abs(test_accuracy_women - test_accuracy_men))
Fair_Metric
```

```
0.8460842370986938
```

J'ai donc décidé d'attribuer un poids plus élevé aux femmes (1) et un poids plus faible pour les hommes (0.1), dans le but de promouvoir une meilleure équité dans l'apprentissage du modèle. Cette stratégie vise à équilibrer les performances entre les genres, même si cela peut impliquer une légère baisse de la performance du modèle sur les hommes. Cet ajustement des poids est une tentative de corriger les biais de genre observés antérieurement, en veillant à ce que le modèle traite de manière plus équilibrée et juste les images de chaque genre.

```
train_df['gender_mapped'] = train_df['gender'].map({-1: 1, 1: 0.1})
```

```
train_generator = datagen.flow_from_dataframe(
    dataframe=train_df,
    directory=None,  # Aucun répertoire racine, les chemins sont déjà complets dans le DataFrame
    x_col='filename',  # Colonne contenant les chemins des fichiers
    y_col='class',     # Colonne contenant les labels des classes
    target_size=(80, 80),
    batch_size=32,
    class_mode='binary',
   weight_col='gender_mapped'
)
```

Ci-dessous les résultats sur la base de test en séparant les hommes et les femmes.

```
497/497 [==============================] - 13s 26ms/step - loss: 0.0078 - accuracy: 0.9074
497/497 [==============================] - 13s 26ms/step - loss: 0.0069 - accuracy: 0.9203
Performance sur les Femmes - Loss: 0.007827784866094 59, Accuracy: 0.9073910713195801
Performance sur les Hommes - Loss: 0.006852862890809774, Accuracy: 0.920302152633667
```

```
Fair_Metric = np.mean([test_accuracy_women, test_accuracy_men]) - (2*np.abs(test_accuracy_women - test_accuracy_men))
Fair_Metric
```

```
0.8880244493484497
```

On constate une amélioration de la Fair Metric, j'ai donc gardé cette configuration.

Deep learning : classification de visage – projet

Question 8

Concentrer la classification sur des caractéristiques "softbio" plutôt que sur l'image entière augmente la complexité du modèle. Il doit identifier et interpréter des détails subtils et variables tels que la couleur des yeux ou la forme du menton. Cela nécessite une architecture de réseau de neurones plus sophistiquée, capable de capter ces nuances fines.

De plus, la nature subjective de ces caractéristiques implique un besoin accru de traitement avancé des données pour assurer une labellisation précise et cohérente. La gestion des labels bruités devient également un enjeu majeur, étant donné la plus grande probabilité d'erreurs ou d'ambiguïtés dans les labels.

Enfin, atteindre une précision de classification élevée représente un défi supplémentaire, en raison de la subtilité des caractéristiques "softbio" par rapport aux caractéristiques visuelles directes.

Deep learning : classification de visage – projet

Question 9

Oui, j'ai tenté d'utiliser le modèle pré-appris ResNet50

```
import tensorflow as tf
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.layers import Dense, Flatten, Dropout
from tensorflow.keras.models import Sequential

# Charger le modèle ResNet50 pré-entraîné sans la couche supérieure
base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(80, 80, 3))
```

```
# Geler les couches du modèle pré-entraîné
for layer in base_model.layers:
    layer.trainable = False
```

```
# Construire le modèle avec les nouvelles couches de classification
model = Sequential([
    base_model,
    Flatten(),
    Dense(512, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])
```

cependant les résultats n'ont pas été satisfaisant comme le montre la capture ci-dessous :

```
# Évaluation sur l'ensemble de test (population totale : homme + femme)
test_loss, test_accuracy = model.evaluate(test_generator)
print(f"Test Loss: {test_loss}, Test Accuracy: {test_accuracy}")
```

```
993/993 [==============================] - 184s 185ms/step - loss: 0.3443 - accuracy: 0.8171
Test Loss: 0.3443133533000946, Test Accuracy: 0.817085862159729
```

La sous-performance du modèle peut être attribuée à plusieurs facteurs clés : l'adaptation insuffisante de l'architecture ResNet50 aux caractéristiques "softbio" spécifiques, le gel potentiellement excessif des couches du modèle pré-entraîné empêchant l'apprentissage de caractéristiques importantes dans les données, des ajustements d'hyperparamètre inadéquats, une divergence entre les données du pré-entrainement et celles du projet, l'équilibrage des classes des données d'entraînement.

Pour utiliser efficacement un modèle pré-entraîné il faut d'abord choisir un modèle approprié en considérant la complexité de la tâche, la taille et la qualité des données, les ressources informatiques disponibles, et les performances attendues.

Des modèles comme ResNet ou VGG sont des choix populaires en raison de leur polyvalences. Ensuite, il est nécessaire d'adapté le modèle choisi aux données d'entrainement en gelant certaines couches pour conserver les connaissances pré-apprises et ajoutez des couches personnalisées pour la classification spécifique à la tâche que l'on souhaite accomplir.

Enfin, il faut entrainer, évaluer et validé le modèle sur les ensembles d'entrainement, validation et de test et faire des ajustement itératifs pour améliorer les résultats.

Deep learning : classification de visage – projet

Question 10

Pour améliorer les performances du modèle, on pourrait considérer plusieurs stratégies :

1. L'augmentation des données : Au lieu de réduire la classe minoritaire (femmes) pour atteindre l'équilibre, on pourrait générer de nouvelles images en appliquant des transformations légères aux images existantes. Cela augmenterait la variabilité des données d'entraînement.

2. Réglage des hyperparamètres : Il est possible d'ajuster les hyperparamètres du modèle tels que le taux d'apprentissage, la taille du batch, le nombre d'époques, en effectuant une recherche par validation croisée.

3. Utilisation d'architecture plus complexe : L'expérimentation avec des architectures de modèles plus complexes pourrait consister à ajouter davantage de couches convolutives, de couches entièrement connectées, ou à augmenter le nombre de filtres dans chaque couche.

4. Utilisation d'un modèle pré-entrainé : : On pourrait adapter un modèle pré-entraîné à notre tâche de classification en ajoutant des couches personnalisées.

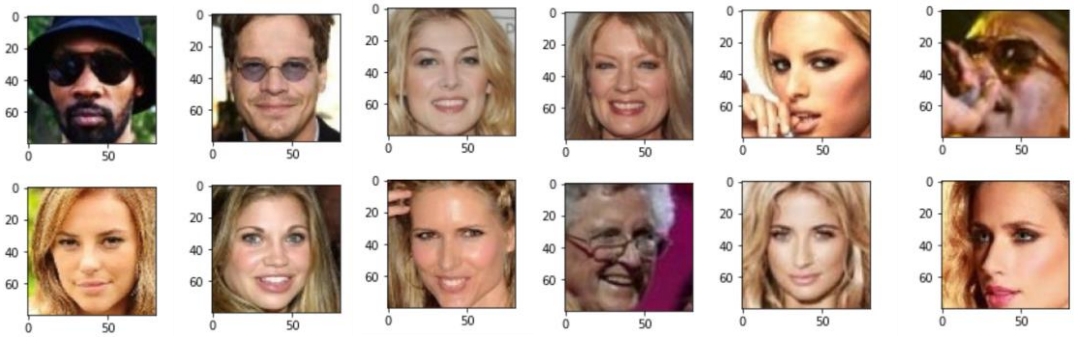
5. Utilisation d'une fonction de perte de fairness plus avancée : La création d'une fonction de perte personnalisée visant à minimiser les biais et à améliorer l'équité dans les prédictions du modèle pourrait être envisagée.

Deep learning : classification de visage – projet

Question 11

Après une analyse des labels et des images, On peut en déduire que le label recherché correspond à une combinaison de deux caractéristiques : les personnes ayant des cheveux blonds ou blondes et les personnes portant des lunettes sont catégorisées comme appartenant à la classe "1", tandis que les autres sont catégorisées comme appartenant à la classe "-1". Cela est illustré par les exemples suivants :

Echantillon de personnes appartenant à la classe « 1 »



Et ceux classés comme « -1 »