

Statistique de grande dimension et apprentissage profond

Classification d'images de visages selon leur expression

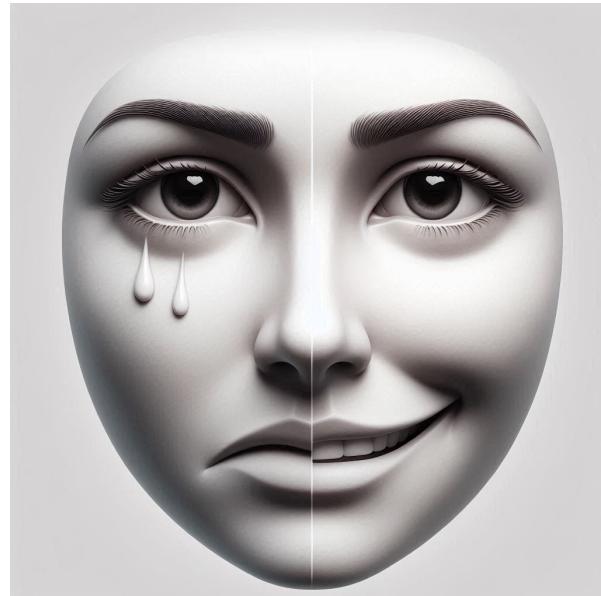


Image générée par l'IA générative de Microsoft Designer

Yanis Mac - Hicham Lagham - Maxime Moshfeghi - Anh Minh Phung

INSA Toulouse
France

18 février 2025

Table des matières

Introduction	2
1 Création du dataset	3
1.1 Utilisation de vidéo pour trouver des visages émotif	3
1.1.1 Vidéos de bêtisiers	3
1.1.2 Vidéo de Micro-trottoir	3
1.2 Récupération des données via Flickr	4
1.3 Traitement de données	4
2 Entraînement du modèle	6
2.1 Premier modèle utilisé : YOLO-classification	6
2.1.1 Entraînement sur les 4 classes originales	6
2.1.2 Influence de la taille d'image en entrée	7
2.1.3 Retrait d'une classe	8
2.1.4 Réévaluation du dataset d'entraînement	9
2.1.5 Changement du jeu de validation	10
2.2 Deuxième modèle utilisé : ResNet	11
2.2.1 Rappel sur les réseau de neurones résiduel	12
2.2.2 Préparation des données	12
2.2.3 Paramètres utilisés	12
2.2.4 Résultats	13
2.2.5 Analyse des cas symptomatiques	13
3 Prédiction en temps réel et cas d'usage	14
3.1 Avec YOLO	14
Conclusion	16

Introduction

Le projet réalisé vise ici à classifier des images de visages selon leur expression. Pour notre problème, nous avons décidé de conserver 4 expressions : joie (**happy**), colère (**angry**), surprise (**surprised**) et neutre (**neutral**).

Dans ce genre de problème, un enjeu majeur est d'abord de savoir si on va faire nos entraînements sur les images de visage uniquement, ou bien sur des images plus larges (allant jusqu'à inclure les épaules, voire même un corps en entier). Un autre problème pouvant se présenter est aussi le cas où plusieurs visages seraient présents sur l'image. Après réflexion, l'approche adoptée avec la suivante : le problème de classification sera restreint aux visages uniquement. Cependant, pour élargir l'utilisation du modèle, on va utiliser un modèle de détection d'objet afin d'extraire les visages de chaque image. D'une image large on pourra ainsi extraire un ou plusieurs visages, et obtenir une classification pour chacun de ces visages.

Chapitre 1

Création du dataset

L'objectif de cette section est de créer un ensemble de données contenant des images de visages étiquetés en fonction de leurs émotions. Nous capturerons 4 émotions :

- classe **happy**
- classe **angry**
- classe **surprised**
- classe **neutral**

Dans cette section, nous avons opté pour différentes méthodes (plus ou moins efficaces) pour obtenir ces images. Voici une explication de chaque méthode et de leurs forces et faiblesses.

1.1 Utilisation de vidéo pour trouver des visages émotif

L'idée est ici de trouver des types de vidéos où l'on voit des visages avec des émotions « fortes ».

1.1.1 Vidéos de bêtisiers

Nous avons commencé par utiliser des vidéos de bêtisiers, car nous pensions y trouver des visages très émotifs.

Pour ce faire, nous avons récupéré des vidéos sur YouTube et conçu un code qui nous permet de récupérer une photo toutes les 10 secondes de ces vidéos.

Ainsi, avons obtenus ces résultats :



FIGURE 1.1 – Exemples d'images obtenus avec des videos de bêtisier

En conséquence, les images obtenues étaient de qualité moyenne, peu exploitables et les émotions obtenues étaient souvent les mêmes (happy) ...

1.1.2 Vidéo de Micro-trottoir

Etant donné la mauvaise qualité des images obtenues précédemment, nous avons pensé que l'utilisation de vidéo de Micro-trottoir nous permettrait d'obtenir des données de meilleure qualité. Nous avons donc utilisé le même procédé et avons pu obtenir des images.

Voici les résultats :



FIGURE 1.2 – Exemples d’images obtenus avec des vidéos de Micro-trottoir

Ces images étaient de bien meilleure qualité et plus utilisables pour notre travail. Malgré cela, nous avons opté pour une autre méthode, qui n’était pas parfaite, et nous nous sommes heurtés à deux problèmes :

- Les émotions obtenues n’étaient pas réparties uniformément dans les classes, ce qui créait des classes d’une taille disproportionnée.
- Les images devaient être classées manuellement, ce qui prenait beaucoup de temps.

Nous avons donc décidé d’abandonner l’idée d’utiliser des vidéos pour obtenir nos images.

1.2 Récupération des données via Flickr

Pour récupérer beaucoup d’images contenant des visages avec les émotions que nous souhaitons, nous pouvons faire des recherches ciblées sur des sites hébergeant des images. Dans notre cas, nous faisons le choix d’utiliser Flickr et son api pour télécharger en masse (500 images par requêtes) des images à l’aide de mots-clés. Voici certains exemples de mots clés utilisés pour chacune des classes :

- classe **happy** : wedding, party.
- classe **angry** : protest, debate.
- classe **surprised** : sudden loud noise, death sentence.
- classe **neutral** : walking in the street, chess game.

Voici figure 1.3 quelques exemples d’images pour la classe angry que nous avons obtenus avec cette méthode.



FIGURE 1.3 – Exemples d’images obtenus avec Flickr

On remarque donc sur ces images qu’il y a des données qui ne sont pas en lien avec le problème que nous souhaitons traiter, ou alors qui ne sont pas adaptées pour un entraînement de modèle. En réalité, il y en a un assez grand nombre, on a environ 1 image sur 10 (cela varie selon les requêtes) qui pourrait être utilisée pour constituer notre base de données. C’est pourquoi nous devons faire un traitement de ces données pour effectuer un tri et également pour récupérer uniquement les visages, puisque ce sont uniquement les visages qui nous intéressent.

1.3 Traitement de données

Pour effectuer ce traitement de données, nous avons plusieurs solutions. La première solution serait de le faire manuellement, mais faire un crop des visages et supprimer autant de données à la main prendrait trop de

temps, c'est pourquoi nous avons opté pour une seconde solution qui utiliserait un modèle YOLO de détection de visages. Nous avons donc utilisé ce modèle sur toutes nos images pour récupérer uniquement les visages détectés dans nos images.

Une fois ce premier tri effectué, il ne nous reste plus qu'à déterminer si les images sont adaptées au type de dataset que nous souhaitons créer et si elles correspondent bien aux classes que nous souhaitons traiter. Pour cela, nous avons du le faire manuellement.

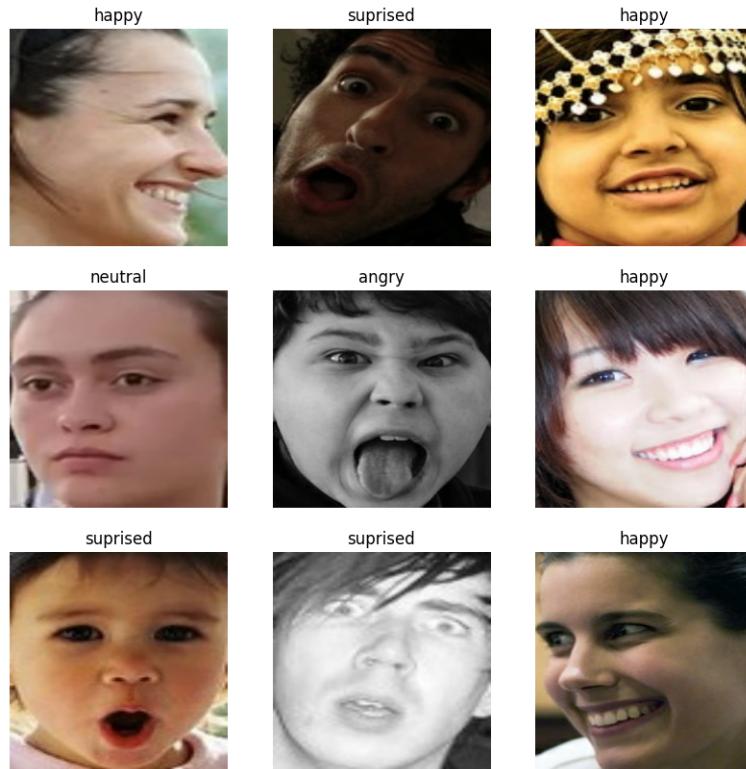


FIGURE 1.4 – Images du dataset

Nous observons figure 1.4 que nous avons enfin des images exploitables dans la base de données. Voici un lien vers l'ensemble des données obtenues :

https://drive.google.com/drive/folders/16zXFAV01hWT5x3qty5F6Q1EYYYyIlZGW?usp=drive_link

Chapitre 2

Entraînement du modèle

2.1 Premier modèle utilisé : YOLO-classification

Pour classifier les images de visage dans les quatre classes d'émotions, nous avons utilisé **YOLO-classification** (v8), un modèle de classification d'images préentraîné sur la base de données **COCO**. Il existe plusieurs versions de ce réseau de neurones, et nous en avons testé ici plusieurs. Les réglages des paramètres d'entraînement ont tous été définis par défaut conformément à la documentation d'Ultralytics (1) exceptés le nombre d'epochs, fixé à 100, et la taille d'image fixé à 128.

2.1.1 Entraînement sur les 4 classes originales

Dans un premier temps, nous avons tenté d'entraîner sur les 4 classes définies plus haut **YOLO-classification**.

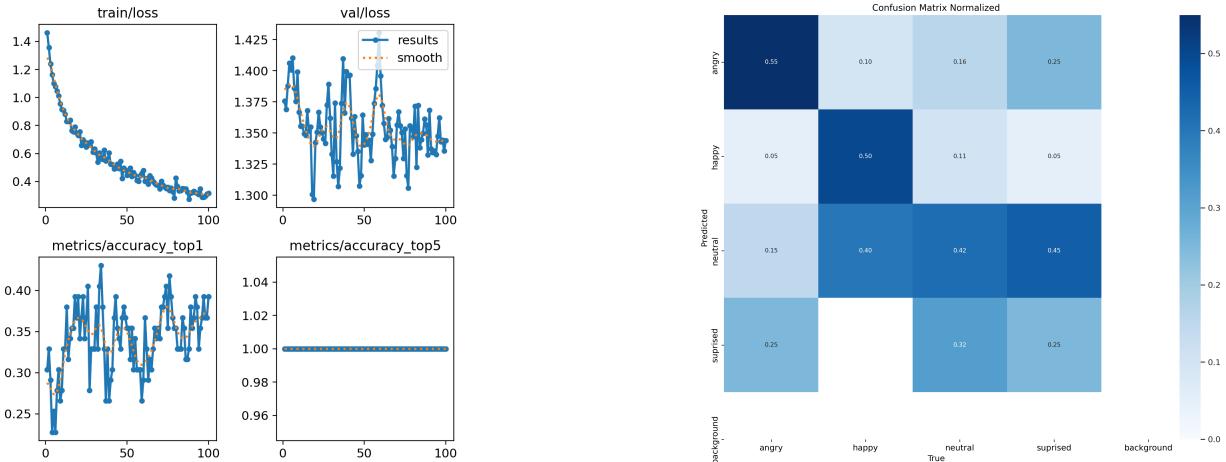
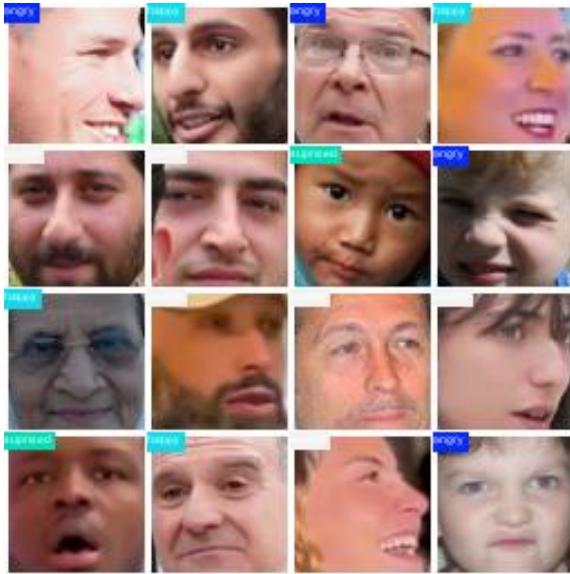


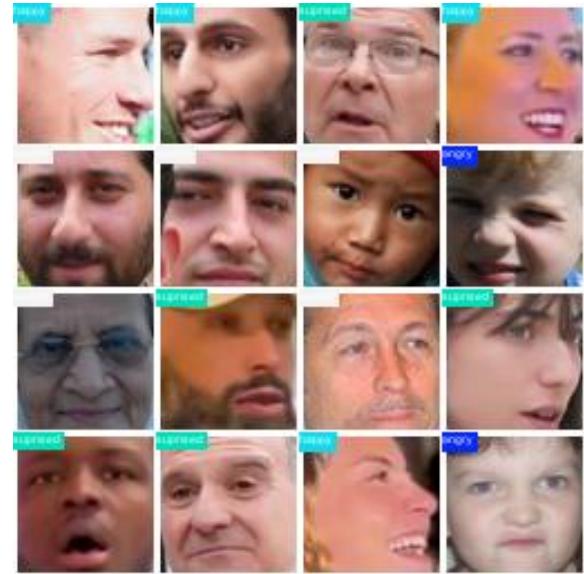
FIGURE 2.1 – Résultats de l'entraînement de **YOLO-classification** (v8-nano)

La taille du dataset de validation ayant été prise volontairement basse, on remarque qu'on ne peut pas se fier à la courbe de validation loss donnée par YOLO, qui oscille trop d'une epoch à une autre. En revanche, la matrice de confusion donne une idée assez bonne des qualités et défauts de notre classification. On tire ainsi une première conclusion de ces résultats : les classes **angry** et **happy** se distinguent très bien des autres classes, tandis que les classes **neutral** et **surprised** sont plus difficiles à correctement évaluer.

On peut alors observer quelles sont les images qui on pu poser problème :



(a) Prédictions



(b) Labels réels

FIGURE 2.2 – Analyse de la validation pour **YOLO-classification (v8-nano)**

Sur cette comparaison, on constate que la résolution pourrait être à l'origine de cette mauvaise classification, et on va s'intéresser à l'influence de ce paramètre sur la qualité de notre prédiction.

2.1.2 Influence de la taille d'image en entrée

Précédemment, la taille d'image en entrée était restreinte à 64×64 pixels. Voici les résultats de notre entraînement en augmentant cette taille en entrée à 128×128 :

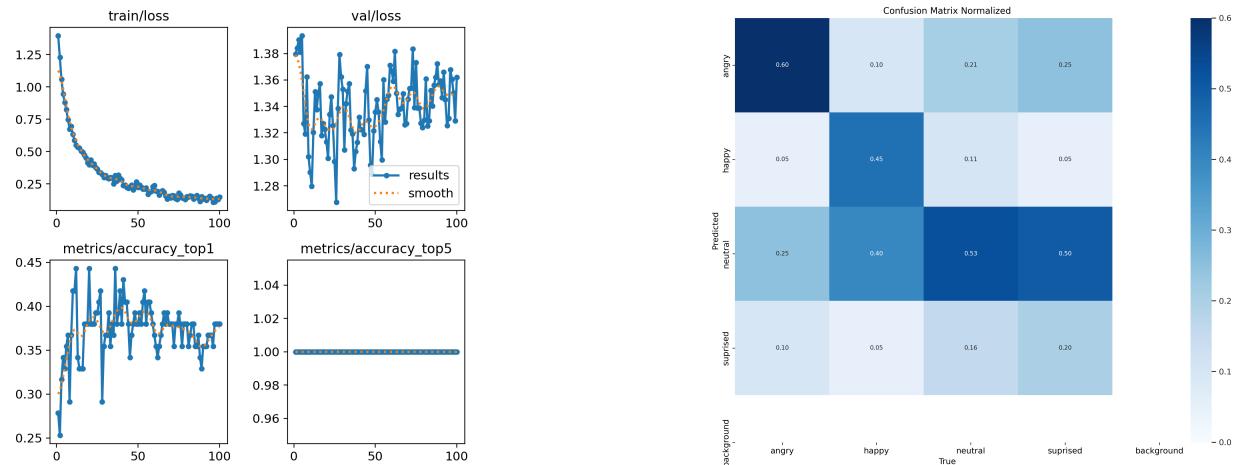


FIGURE 2.3 – Résultats de l'entraînement de **YOLO-classification (v8-nano)** en augmentant la taille d'image en entrée

La conséquence la plus flagrante de cette augmentation de la taille d'entrée semble être l'entraînement, dont la courbe de loss converge légèrement plus rapidement. Il semblerait aussi que l'on prédise un peu mieux la classe neutral.

Si on compare manuellement par rapport à l'entraînement précédents, on peut aussi constater une correction de la prédiction pour certains éléments, comme on peut le voir ci-dessous :



(a) Label réel

(b) Prédiction pour 64×64 pixels(c) Prédiction pour 128×128 pixels

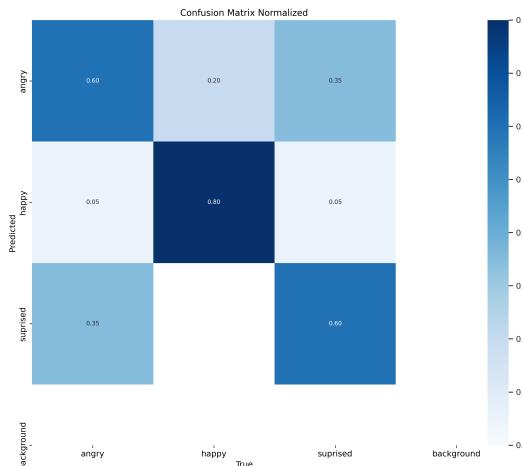
FIGURE 2.4 – Comparaison des prédictions pour différentes tailles d'image en entrée

En revanche, on a un dépeuplement de la classe surprised. On comprend donc qu'un des problèmes peut venir d'un défaut de labellisation de notre jeu d'entraînement, introduit en partie par l'appréciation personnelle de chacun des membres de notre équipe de ce qu'est une certaine expression. Un travail sera ensuite fait pour voter en groupe si chaque image de chaque classe du jeu d'entraînement a effectivement sa place dans sa classe, ou si elle devrait en être retirée.

2.1.3 Retrait d'une classe

Pour tester si les défauts de classification ne pourrait pas venir d'un problème de complexité dans la diversité de classes à prédire, on s'intéresse aux résultats obtenus après suppression d'une de 4 classes pour la phase d'entraînement et de validation.

En retirant la classe neutral, on obtient la matrice de confusion suivante :

FIGURE 2.5 – Résultats de l'entraînement de **YOLO-classification** (v8-nano) après retrait de la classe neutral

Sans grande surprise, la classification se fait mieux. On remarque tout de même qu'il reste une faiblesse sur la classification des surprised et angry qui demeurent régulièrement confondues.

On peut aussi essayer de retirer la classe surprised :

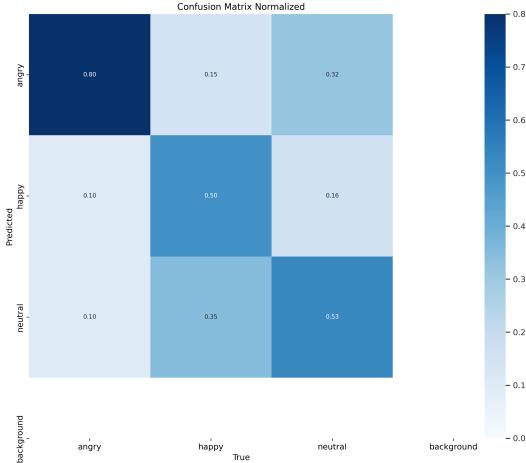


FIGURE 2.6 – Résultats de l'entraînement de **YOLO-classification** (v8-nano) après retrait de la classe surprised

Là aussi, la matrice de confusion est bien meilleure que pour 4 classes. On remarque que la classe neutral semble "diluer" les autres classes, en soulignant que beaucoup de happy sont classifiés neutral, et beaucoup de neutral sont classifiés angry.

2.1.4 Réévaluation du dataset d'entraînement

Après les premiers entraînements, nous avons estimé qu'il était essentiel de relabelliser nos classes, pour que nous soyons d'accord sur l'appartenance de chaque image à sa classe. Nous avons donc procéder à un passage en revue de chacune des images, avec un vote à 4 personnes à main levé. Chaque image doit obtenir une majorité absolue de "oui" pour rester dans la classe qui lui a initialement été affectée. On peut donner quelques exemples dessous d'images qui ont été retirées des datasets :



(a) Classe angry

(b) Classe neutral

(c) Classe surprised

FIGURE 2.7 – Images retirées de chaque classe

La classe happy étant déjà plutôt bien définies, presque aucune image n'en a été retirée. On remarque tout de même certains visages "artificiels" que l'on retire (et on les retire d'ailleurs aussi des autres classes). Voici un exemple des visages "artificiels" retirés pour la classe happy :



FIGURE 2.8 – Retrait des visages artificiels pour la classe happy

Après cette réévaluation du dataset et en entraînant sur le modèle medium, on obtient le résultat suivant :

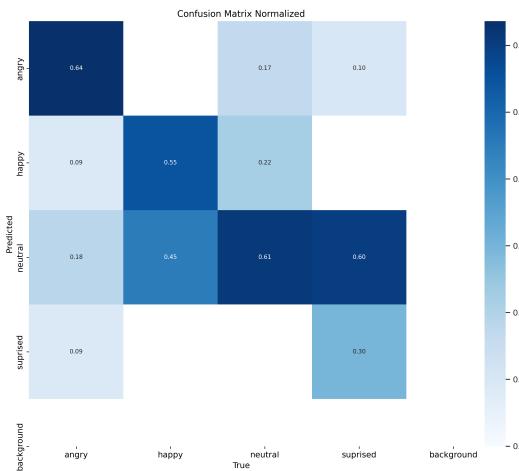


FIGURE 2.9 – Résultats de l’entraînement de **YOLO-classification** (v8-medium) sur le dataset réévalué

2.1.5 Changement du jeu de validation

N’ayant jusque là pas fait de cross-validation avec les modèles YOLO-classification et n’ayant pas tant de visibilité sur les loss de validation et l’accuracy de validation, une idée a été de changé le jeu de validation, à savoir de re-splitter notre jeu global, d’une part pour avoir un ensemble de validation plus grand mais aussi que l’échantillon soit représentatif c’est-à-dire de piocher une image tout les 5 images de chaque classe (et non pas les 20 premières images de chaque classe, ce qui avait malheureusement été fait jusqu’à présent). On obtient alors, en utilisant le plus grand modèle (v8-extra), avec des images en entrée de 128px et sur 100 epochs :

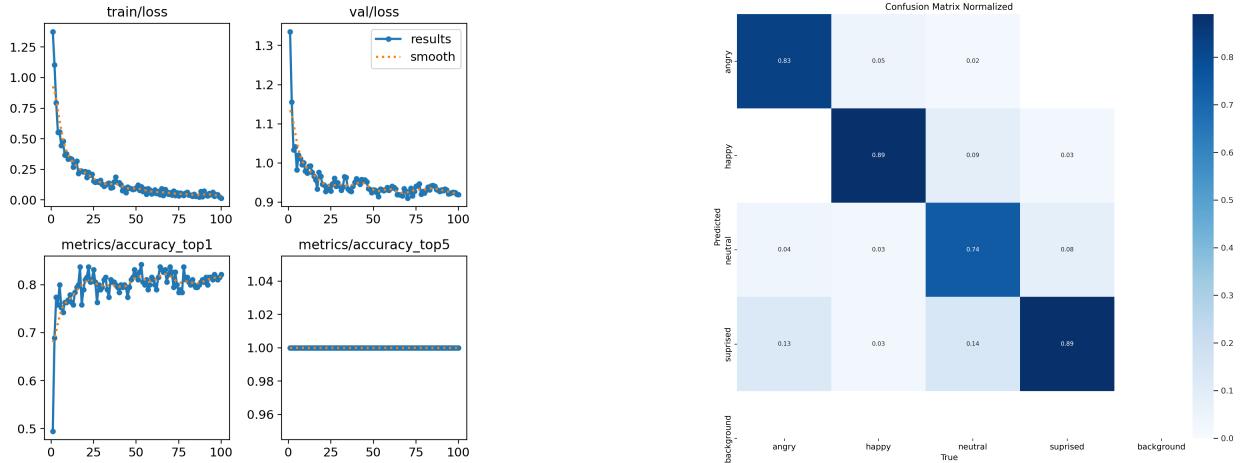


FIGURE 2.10 – Résultats de l’entraînement de **YOLO-classification** (v8-extra) après changement du jeu de validation

On peut ici correctement constater que la loss de validation décroît correctement, et l’accuracy est aussi plus interprétable. Dans les meilleures itérations, on monte au dessus de 0.8 d’accuracy en validation.

De même, on obtient une matrice de confusion à l’allure bien meilleure. La classe la plus dure à correctement prédire demeure la classe neutral.

Voici un lien vers les poids obtenus après l’entraînement :
<https://drive.google.com/drive/folders/13vyXKq-iCjqs8D4hKrBn0xrQrljjRXRS?usp=sharing>

2.2 Deuxième modèle utilisé : ResNet

Il peut être intéressant de voir les résultats obtenus avec la même base de données sur d’autres architectures de réseaux de neurones. Ainsi, dans cette section, nous utiliserons un réseau ResNet-50 (environ 25 millions de paramètres) pré-entraîné sur la base de données ImageNet pour comparer les résultats obtenus avec YOLO-classification. Le modèle que nous avons entraîné est représenté sur le schéma figure 2.11.

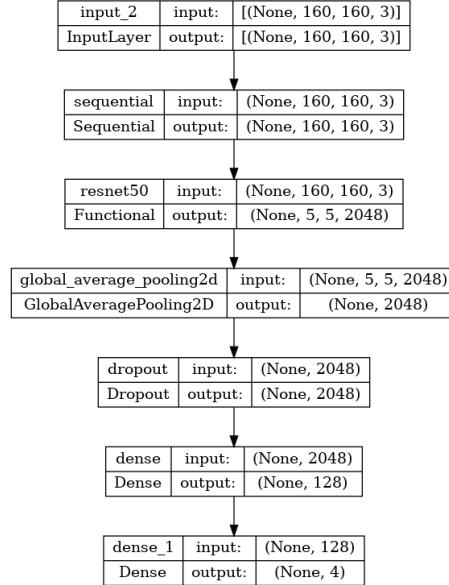


FIGURE 2.11 – Schéma du modèle utilisé

2.2.1 Rappel sur les réseaux de neurones résiduels

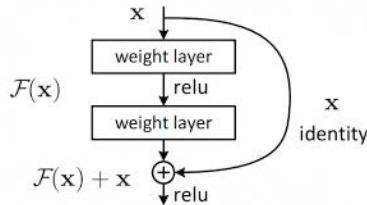


FIGURE 2.12 – Schéma d'un bloc résiduel

L'idée est d'ajouter une connexion comme présentée ci-dessus (*skip connection*) pour former un bloc résiduel.

2.2.2 Préparation des données

Pour préparer nos données, nous avons divisé notre base de données en trois sections :

- Entrainement : 20/25 des données.
- Validation : 4/25 des données.
- Test : 1/25 des données.

Nous avons transformés nos images en les redimensionnant de taille 160x160 et nous avons également augmenté artificiellement notre dataset en appliquant une rotation aléatoire à nos images de façon à augmenter la taille du dataset.

2.2.3 Paramètres utilisés

Nous avons séparé l'entraînement en deux parties avec des paramètres distincts :

Tout d'abord, nous avons effectué une phase de "Transfer Learning", au cours de laquelle nous avons fixé le réseau de neurones convolutifs et n'avons autorisé l'apprentissage que sur les dernières couches du réseau.

Les paramètres utilisés ici, étaient les suivants :

- Loss : Cross-entropy
- Learning rate : 0.0001
- Optimiseur : Adam
- Nombre d'epochs : 10

Nous avons ensuite procédé à un "fine-tuning" du modèle.

Les paramètres utilisés ici, étaient les suivants :

- Loss : Cross-entropy
- Learning rate : 0.00001
- Optimiseur : Adam
- Nombre d'epochs : 40

Nous avions également un callback pour diviser le learning rate par 5 lorsque le loss de validation ne diminue plus depuis 5 epochs.

2.2.4 Résultats

Ainsi, après avoir entraîné notre modèle, nous obtenons les résultats suivants du point de vue de la matrice de confusion :

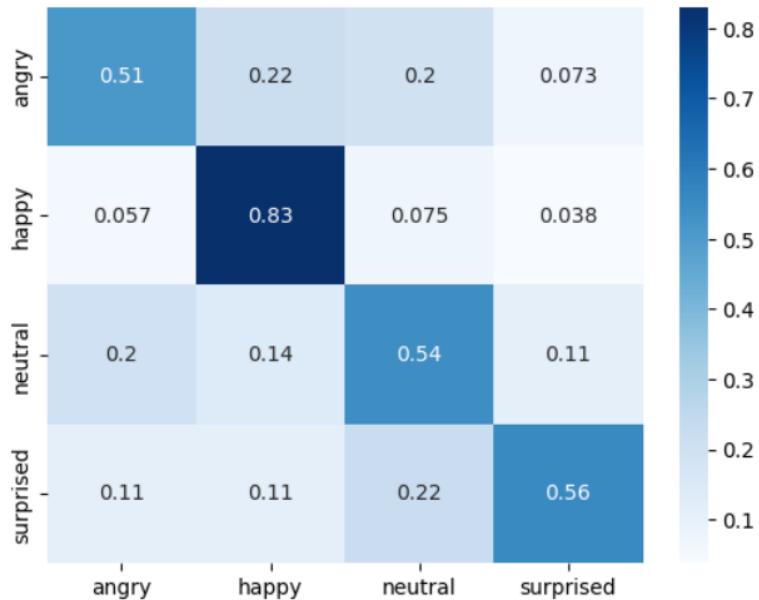


FIGURE 2.13 – Confusion Matrix on the validation set

En calculant la précision et le score F1 de notre modèle, nous constatons que les résultats ne sont pas aussi bons que ceux obtenus par le modèle YOLO-classification. Nous conserverons donc le modèle YOLO-classification, plus performant.

Par ailleurs on retrouve le même comportement pour le modèle YOLO-classification et le modèle ResNet50 sur la figure 2.14. En effet on se rend compte que l'on obtient une meilleure précision sur l'expression "happy" que sur les autres expressions. Cela peut se traduire par le fait que l'expression "happy" est bien plus facile à reconnaître et à identifier que les autres, là où pour les autres classes il pourrait y avoir beaucoup plus de nuances.

2.2.5 Analyse des cas symptomatiques

Il est néanmoins intéressant d'analyser les erreurs commises par notre modèle sur l'ensemble de test :



FIGURE 2.14 – Example of prediction error on the resnet model.

Lorsque nous examinons les erreurs de prédiction de notre modèle, nous constatons qu'un large échantillon d'erreurs peut s'expliquer par le fait que les personnes portent des lunettes, ce qui rend plus difficile la lecture de leurs émotions par le réseau neuronal.

Chapitre 3

Prédiction en temps réel et cas d'usage

3.1 Avec YOLO

Pour tester notre modèle, nous avons trouvé intéressant de tester sur certaines personnes de la classe. Là aussi, pour prédire l'expression du visage, on utilise d'abord un modèle de détection d'objets pour extraire les visages et on classe ensuite avec notre modèle. Voici quelques résultats :

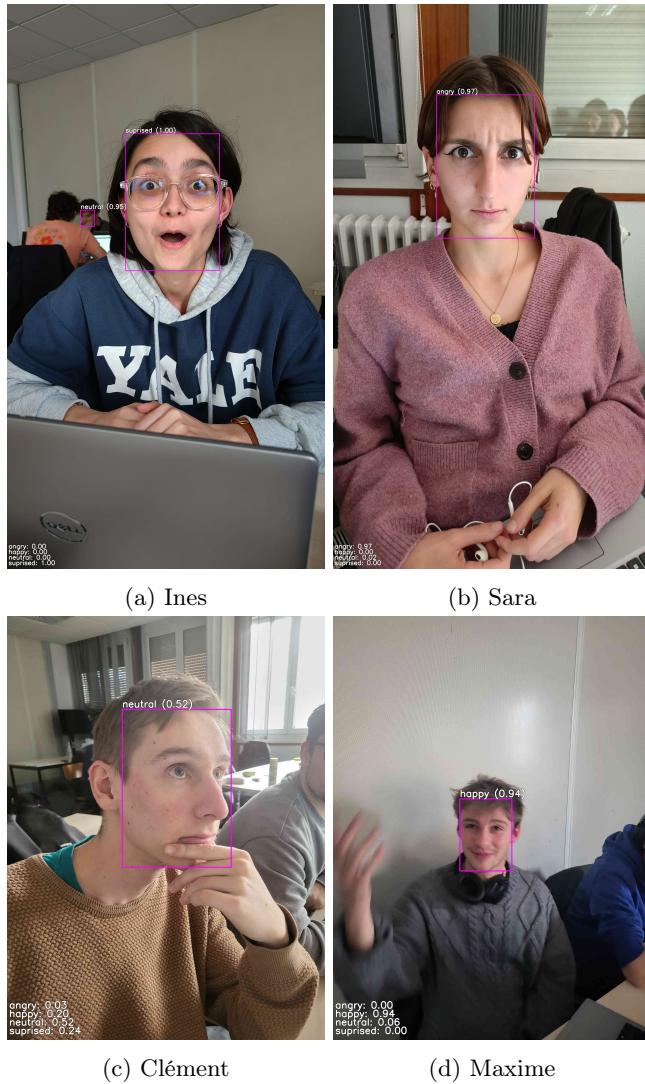


FIGURE 3.1 – Quelques prédictions faites avec YOLO-classification

Certains cas sont aussi intéressants à étudier. Par exemple, une expression `angry` est mieux détectée avec lunettes que sans lunettes :

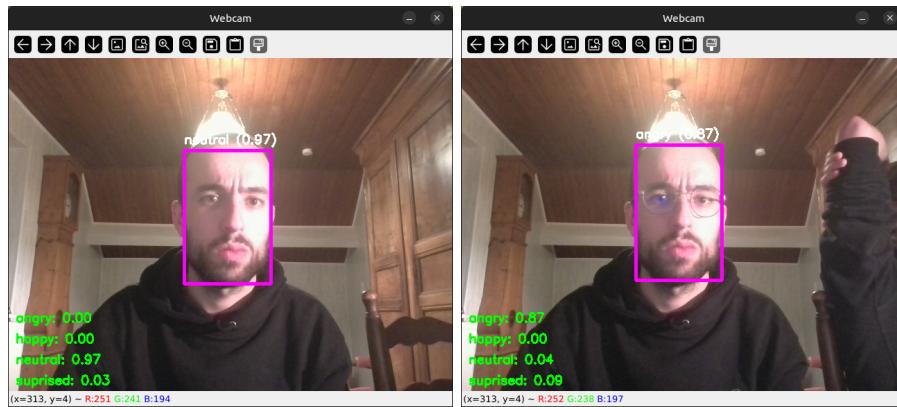


FIGURE 3.2 – Comparaison de la prédition avec et sans lunettes

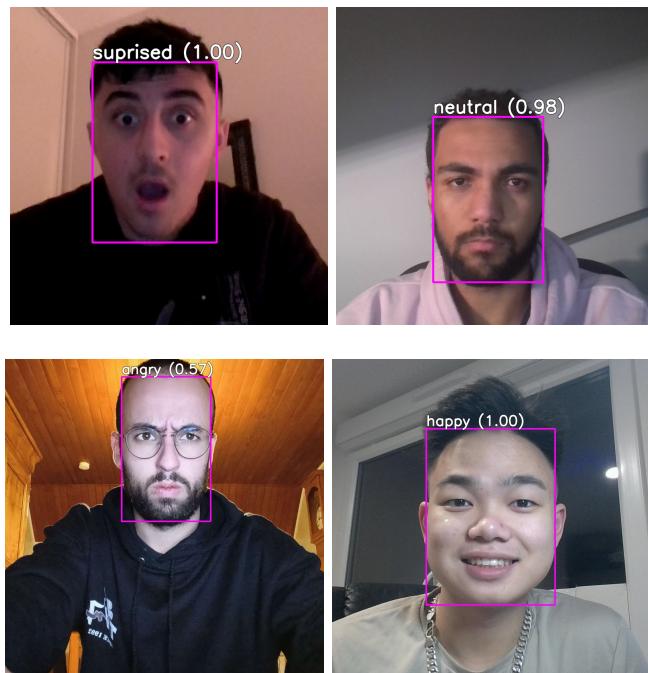
Conclusion

En conclusion, ce projet nous a permis de classifier des expressions faciales en utilisant des modèles d'apprentissage profond, dans ce projet YOLO-classification et ResNet. Nous avons ainsi pu nous approprier le cadre théorique appris en cours sur ces types de réseaux. De plus, la création du jeu de données nous a permis de comprendre la complexité et le temps nécessaire à la création d'une simple base de données d'environ 1000 images.

Une des principales difficultés de ce projet aura été la constitution de la base de données. Réussir à trouver des images de qualité correspondant aux expressions faciales que nous recherchions aura été une des premières difficultés rencontrées. Ensuite, lorsque nous avons formé notre première base de données et après notre premier entraînement, nous nous sommes rendus compte que nous n'étions pas forcément d'accord sur la classification de nos images entre nous. Cela nous a indiqué qu'un certain biais propre à chaque humain existe pour déterminer l'expression ou l'émotion sur un visage humain, ce qui pose un problème pour l'entraînement de notre modèle car ayant été plusieurs à labelliser chaque image le modèle doit apprendre tous nos biais en même temps.

Cela nous amène à des évolutions potentielles de ce projet. On pourrait changer légèrement le problème en faisant en sorte que notre modèle puisse donner plusieurs émotions à un visage. On aurait alors un problème de classification multi-étiquette.

On pourrait sinon étendre le contexte des images, peut-être en prenant en compte une séquence temporelle sur les images pour les vidéos, ce qui donnerait un contexte plus riche à notre réseau de neurones.



Bibliographie

[1] <https://docs.ultralytics.com/fr/modes/train/>