



## Rapport final Projet Deep Learning

Mathieu Campan, Priscilia Gonthier, Sarah Abou Daya,  
Amine Benchaaboun, Simon Hautesserres  
Groupe L12 - 3

Département Sciences du Numérique  
2022-2023

## Table des matières

1 Sujet du projet : . . . . .	3
1.1 Sujet choisi : . . . . .	3
1.2 Méthodologie : . . . . .	4
2 Elaboration de la solution : . . . . .	7
2.1 Etapes suivies et Architectures : . . . . .	7
3 Analyse des résultats : . . . . .	12
3.1 Analyse quantitative : . . . . .	12
3.2 Analyse qualitative : . . . . .	14
4 Conclusion : . . . . .	17
4.1 Bilan et problèmes rencontrés : . . . . .	17
4.2 Pistes d'amélioration : . . . . .	17

## 1 Sujet du projet :

### 1.1 Sujet choisi :

#### 1.1.1 Explication illustrée du sujet :

Le sujet de notre projet porte sur la classification de différentes variétés de pâtes, à travers une base de données partagée en 10 classes.

Nous avons choisi ce sujet en raison de notre amour pour les pâtes et de notre envie de pouvoir différencier simplement les différents types de pâtes. Nous avons également été inspiré par les nombreux plats de pâtes différents existants dans le monde entier et avons été curieux de découvrir les différences subtiles entre les variétés.

En utilisant l'apprentissage profond pour classifier les pâtes, nous espérons non seulement en apprendre plus sur les différentes variétés, mais également aider d'autres amateurs de pâtes à mieux comprendre et apprécier la richesse de la cuisine italienne.



FIGURE 1 – "Nous aimons les pâtes"

#### 1.1.2 Lien vers la base de données sur GitHub :

Voici le lien pour accéder à notre base de données :

<https://github.com/MathieuCampan/DeepPasta>

Nous avons également développé un script pour charger nos données dans Google Colab :

[https://colab.research.google.com/drive/15\\_nELci0gMwr5nG-QgX3FOeascwhj2mH?usp=sharing](https://colab.research.google.com/drive/15_nELci0gMwr5nG-QgX3FOeascwhj2mH?usp=sharing)

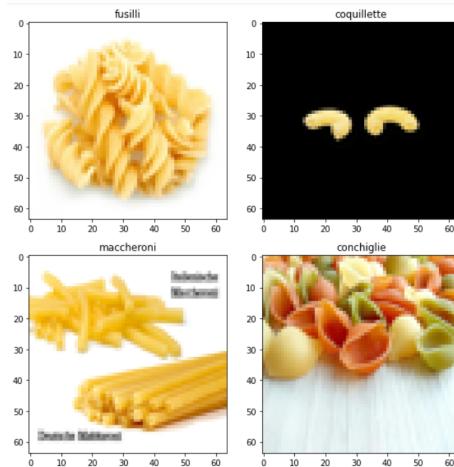


FIGURE 2 – "Exemple de Données Récupérées"

## 1.2 Méthodologie :

### 1.2.1 Acquisition des données :

Pour ce faire, nous avons collecté des données pour chaque classe selon les différentes caractéristiques des pâtes telles que leur forme, leur texture, leur couleur, leur ingrédient de base, etc...

Ces données ont été récupérées par scraping depuis le moteur de recherche google. Nous avons utilisé un script python nommé Google-Image-Scraper, accessible sur github.

Pour chaque classe, nous avons récupéré 200 images pour avoir un total de 2000 images pour notre base de données.

Nous nous sommes répartis 2 classes par personnes, ce qui a fait 400 images à récupérer chacun.

Chacune des images a été sélectionnée pour former au mieux une base d'apprentissage cohérente et complète, notamment en variant le nombre (1 ou 2 pâtes à un plat complet de pâtes nature) mais aussi en y ajoutant quelques ingrédients ou des sauces. De plus, nous avons recadré les images pour nous assurer que les pâtes soient au centre de l'image.

Nous avons également sélectionné les images de bonne qualité et pris en priorité les formats de type .jpg pour faciliter le traitement à venir.

Pour l'annotation des images, nous avons simplement donné comme nom le type de pâtes suivi du numéro de l'image pour pouvoir les différencier rapidement.

Pour ce qui est de la détection de doublons, nous avons utilisé le logiciel gratuit Awesome Duplicate Photo Finder, qui a permis de facilement détecter et supprimer les doublons.

📁 cappelletti	dossier Cappelletti
📁 conchiglie	dossiers renommés pour homogénéité
📁 coquillette	Delete README.md
📁 farfalle	dossiers renommés pour homogénéité
📁 fusilli	dossiers renommés pour homogénéité
📁 lasagne	Delete README.md
📁 maccheroni	dossiers renommés pour homogénéité
📁 penne	dossiers renommés pour homogénéité
📁 spaghetti	dossiers renommés pour homogénéité
📁 tagliatelle	dossier tagliatelle
📄 README.md	dossier img

FIGURE 3 – Dossiers des Classes de pâtes GitHub

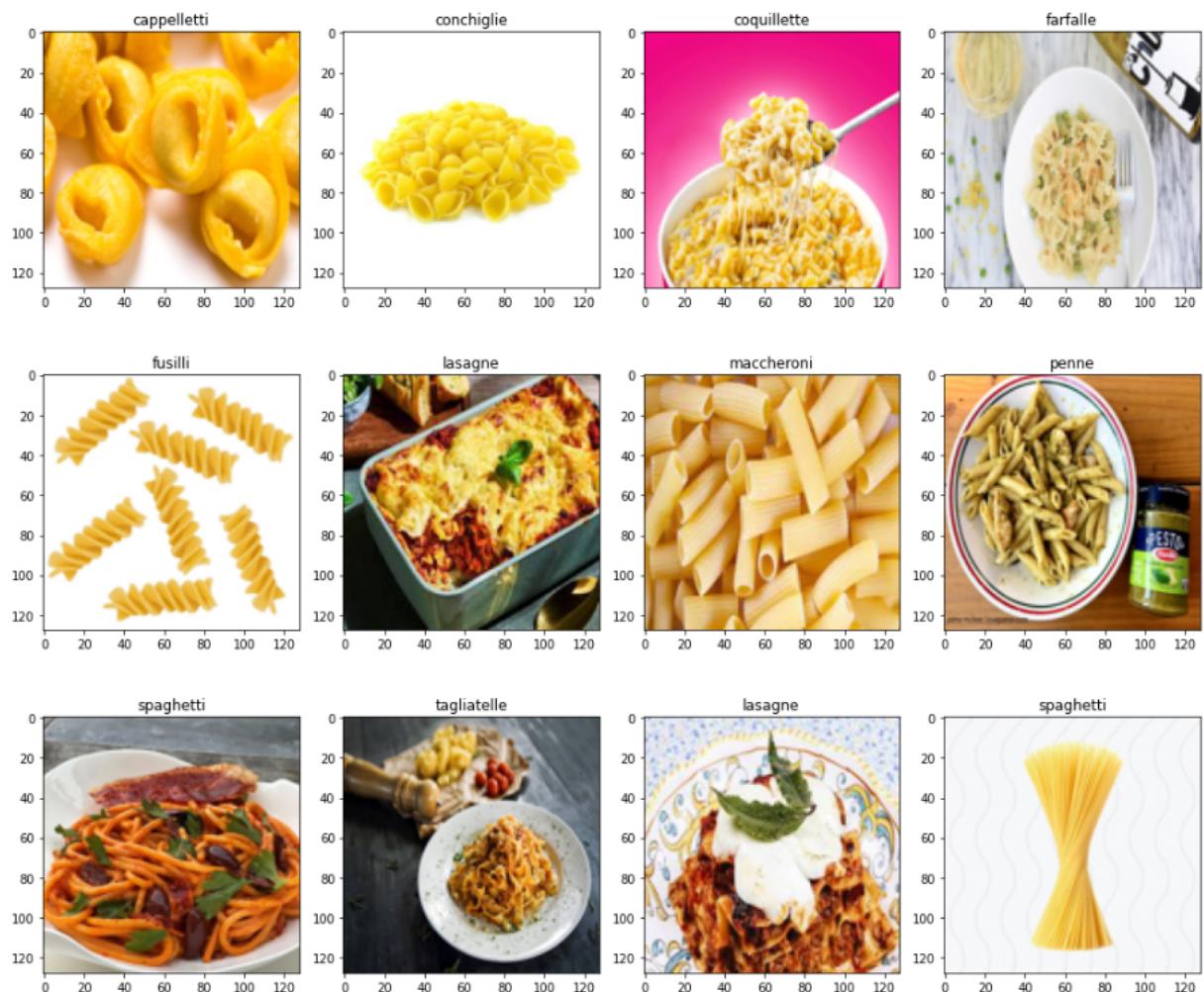


FIGURE 4 – Images issues de notre base de données

### 1.2.2 Répartition des données :

Pour la répartition des données, nous avons décidé de consacrer 80% des images à l'apprentissage, 10% à la validation et 10% au test.

Dans la base d'apprentissage, nous avons mis des images qui montrent un type de pâtes sous différentes formes et perspectives. Pour les tests et la validation, nous avons mis des images un peu plus compliquées pour vérifier si l'algorithme allait bien reconnaître les types. Ces bases comportent notamment des images de plats complets de pâtes, avec de la sauce ou en salade.

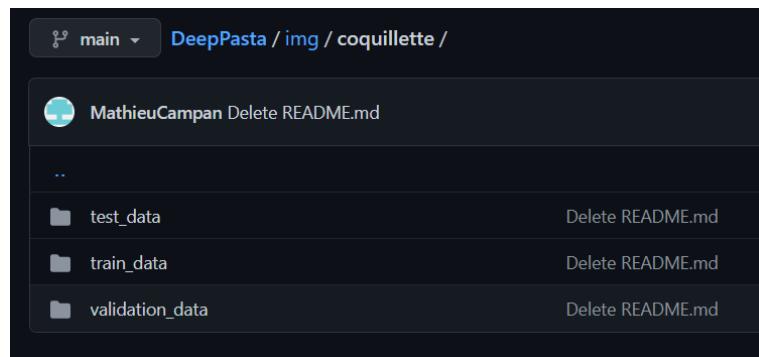


FIGURE 5 – Dossiers des Bases d'Entrainement, de Test et de Validation GitHub

## 2 Elaboration de la solution :

Cette partie décrit de façon détaillée les différentes étapes que nous avons suivies pour résoudre notre problème.

### 2.1 Etapes suivies et Architectures :

Nous avons repris pour commencer les fonctions fournies en Travaux Pratiques pour charger les images de notre bases de données. Nous avons définis des labels comprenant les noms des 10 classes ("tagliatelle", "maccheroni"... ) puis les ensembles d'apprentissage (**x\_train**, **y\_train**), de validation (**x\_val**, **y\_val**) et de tests (**x\_test**, **y\_test**).

Premièrement, nous avons créé un réseau de neurones naïf, traitant des images en  $64 * 64 * 3$  (**IMAGE\_SIZE=64**), pour nous donner une réponse sur une des 10 classes possibles. Nous avons donc, outre les couches de Convolution et de Max-Pooling amenant les données en  $(6 * 6 * 128)$ , une couche dense intermédiaire de 512 neurones et enfin une couche de sortie de 10 neurones (car 10 classes possibles) avec pour fonction d'activation softmax, puisqu'il s'agit d'un problème d'identification de classes.

Model: "sequential"		
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 62, 62, 32)	896
max_pooling2d (MaxPooling2D)	(None, 31, 31, 32)	0
conv2d_1 (Conv2D)	(None, 29, 29, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 14, 14, 64)	0
conv2d_2 (Conv2D)	(None, 12, 12, 96)	55392
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 96)	0
conv2d_3 (Conv2D)	(None, 4, 4, 128)	110720
max_pooling2d_3 (MaxPooling2D)	(None, 2, 2, 128)	0
flatten (Flatten)	(None, 512)	0
dense (Dense)	(None, 512)	262656
dense_1 (Dense)	(None, 10)	5130
<hr/>		
Total params: 453,290		
Trainable params: 453,290		
Non-trainable params: 0		

FIGURE 6 – Résumé de notre premier modèle naïf

Nous avons décidé que pour la fonction de loss de ce modèle, nous prenions la méthode '*'sparse\_categorical\_crossentropy'*' pour déterminer 1 classe parmi 10 autres (donc pas binary, ni mse).

Les tests suivants nous montrent que notre réseau sur-apprend avec un taux de réussite pour validation de 10% (soit une performance équivalente à du hasard).

Nous nous sommes rendu compte qu'un problème venait de notre base de données trop petite pour notre nombre de classes. Nous faisions une répartition 60%-20%-20% pour avoir au moins 20 images de tests et validation par classe.

Nous avons donc agrandi la base de données : 2000 images, 200 par classe avec une répartition 80%-10%-10%. Suite à cette modification, nous avons obtenu un taux de réussite pour validation de **20%**.

Afin d'augmenter ce pourcentage, nous avons effectué trois autres modifications : nous avons utilisé VGG-16 pour utiliser leur réseau pré-entraîné, remplaçant ainsi notre partie de Convolution et MaxPooling.

Model: "model_1"		
Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 128, 128, 3)]	0
block1_conv1 (Conv2D)	(None, 128, 128, 64)	1792
block1_conv2 (Conv2D)	(None, 128, 128, 64)	36928
block1_pool (MaxPooling2D)	(None, 64, 64, 64)	0
block2_conv1 (Conv2D)	(None, 64, 64, 128)	73856
block2_conv2 (Conv2D)	(None, 64, 64, 128)	147584
block2_pool (MaxPooling2D)	(None, 32, 32, 128)	0
block3_conv1 (Conv2D)	(None, 32, 32, 256)	295168
block3_conv2 (Conv2D)	(None, 32, 32, 256)	590080
block3_conv3 (Conv2D)	(None, 32, 32, 256)	590080
block3_pool (MaxPooling2D)	(None, 16, 16, 256)	0
block4_conv1 (Conv2D)	(None, 16, 16, 512)	1180160
block4_conv2 (Conv2D)	(None, 16, 16, 512)	2359808
block4_conv3 (Conv2D)	(None, 16, 16, 512)	2359808
block4_pool (MaxPooling2D)	(None, 8, 8, 512)	0
block5_conv1 (Conv2D)	(None, 8, 8, 512)	2359808
block5_conv2 (Conv2D)	(None, 8, 8, 512)	2359808
block5_conv3 (Conv2D)	(None, 8, 8, 512)	2359808
block5_pool (MaxPooling2D)	(None, 4, 4, 512)	0
flatten_1 (Flatten)	(None, 8192)	0
dense_2 (Dense)	(None, 512)	4194816
dense_3 (Dense)	(None, 10)	5130
<hr/>		
Total params: 18,914,634		
Trainable params: 4,199,946		
Non-trainable params: 14,714,688		

FIGURE 7 – Résumé de notre deuxième modèle VGG 16

Nous avons décidé de garder la même fonction de loss que pour le modèle précédent.

De plus, nous avons utilisé l'augmentation de données en effectuant des rotations, décalages sur les images. Ces opérations sont justifiées car les pâtes sont principalement identifiables par leurs frontières et non par leur position. Les zooms ne sont alors pas appropriés car ils déforment ces frontières. Nous avons pour cela utilisé le code ci-dessous :

```
ImageDataGenerator(  
    rotation_range=100,  
    width_shift_range=0.2,  
    height_shift_range=0.2,  
    shear_range=0.2,  
    zoom_range=0.0,  
    horizontal_flip=True)
```

C'est lors de cette réflexion que nous nous sommes rendu compte du problème majeur : la pixellisation. Les images pixellisées ne sont pas reconnaissables car nous perdons l'information des frontières, ce que nous avons remarqué en affichant certaines images illisibles (principalement des petites pâtes comme les coquillettes). Nous avons donc augmenté la résolution des images à 128 \* 128 \* 3 (**IMAGE\_SIZE=128**).

Après toutes ces modifications, nous avons atteint un taux de réussite pour validation de **60%**.

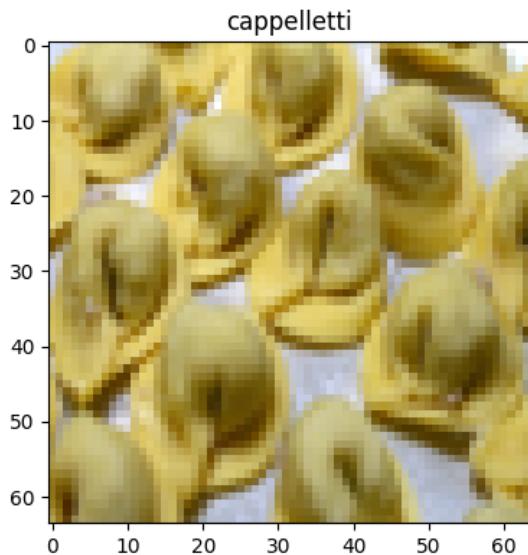


FIGURE 8 – Format des images en 64 pixels

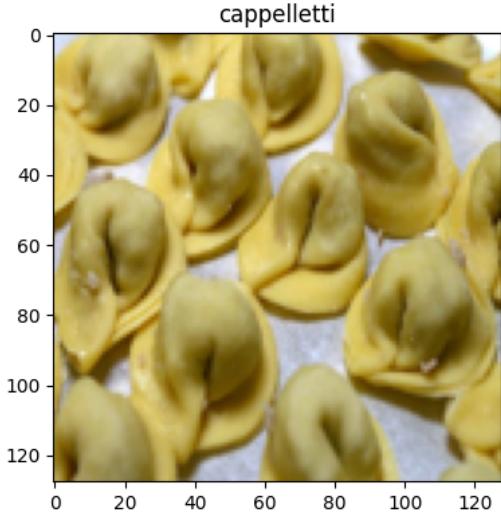


FIGURE 9 – Format des images en 128 pixels

Nous avons également implémenté une matrice de confusion, afin de comprendre quelles classes étaient confondues :

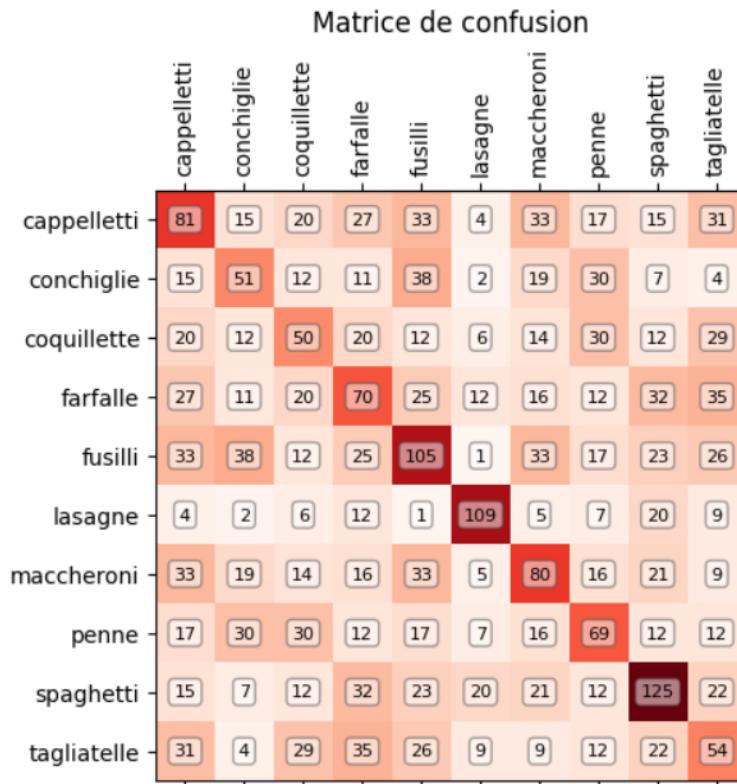


FIGURE 10 – Matrice de confusion avec modèle VGG et résolution de 64, sur l’ensemble d’apprentissage

Avec une résolution de 64 pixels, les images sont très ressemblantes, d'où certaines cases allant jusqu'à plus de 30 mauvaises classifications. Cela signifie qu'il y a beaucoup de confusion entre les classes par notre modèle (nos images ont été prises après les avoir recadrées donc les classes restent bien devinées à 50%).

Nous avons donc générée la matrice de confusion en utilisant des images de meilleure résolution, et observons ainsi, qu'il y a confusion entre les spaghetti et tagliatelles (qui a diminué après l'amélioration de la résolution des images). Cependant, la principale confusion se trouve être entre les pennes et les maccheronis, ce qui est compréhensible car ces pâtes sont presque identiques, et différenciables uniquement par leurs extrémités, ce qui peut être impossible en fonction de leur position et de leur rotation :

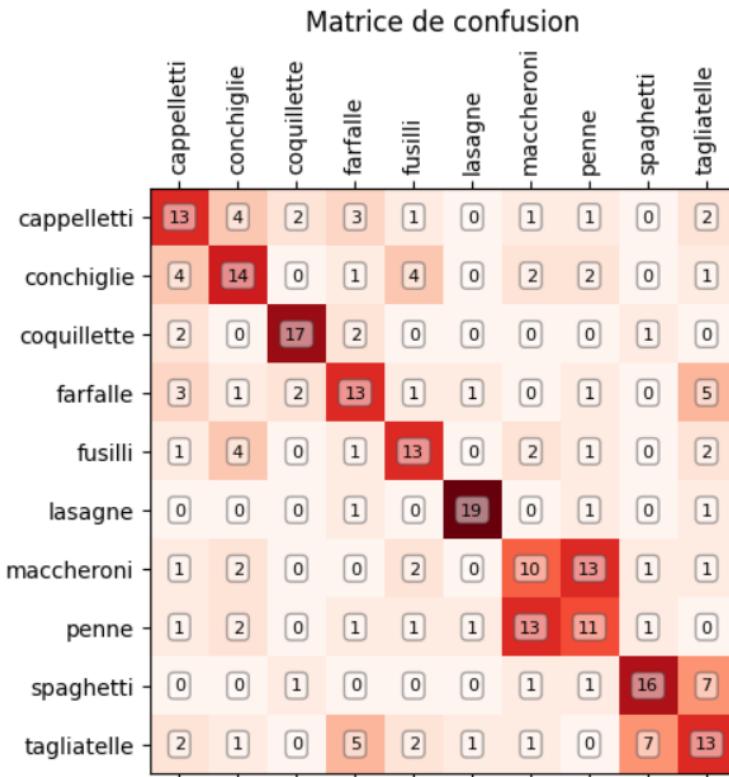


FIGURE 11 – Matrice de confusion avec modèle VGG et résolution de 128, sur l’ensemble de validation

Cet affichage d’images nous a également montré une autre possibilité d’amélioration : le cadrage des images. En effet, nous n’étions pas amplement satisfait de l’intégralité de certaines images qui avaient trop d’informations inutiles. Par exemple, un fond blanc prenant une place non négligeable par rapport aux pâtes qui n’occupaient qu’un quart de l’espace.

Cela étant, nous avons pris la décision de recadrer l’intégralité des images pour ne garder que les informations utiles : les pâtes et plats de pâtes.

Après cette étape, en retestant l’ensemble de notre base de données, nous obtenions des résultats variant entre **65 et 70%** de précision pour la validation.

Suite à ces tests, nous avons augmenté une nouvelle fois la qualité des images : 256 \* 256 \* 3 (**IMAGE\_SIZE=256**). Nous obtenons cette fois un taux de réussite pour validation entre **70 et 75%**.

### 3 Analyse des résultats :

Dans cette partie, nous allons présenter une analyse détaillée de nos résultats qui présentera notre précision globale avec les différentes méthodes utilisées et aussi des exemples illustrés des réussites et des échecs avec les explications associés.

#### 3.1 Analyse quantitative :

##### 3.1.1 Précision globale :

Suite à nos dernières modifications, nous avons lancé notre réseau sur l'ensemble de validation pendant 25 epochs avec un batch-size de 10. Il entraîne puis valide pour chaque epoch les résultats. Pour ce test, la moyenne de temps de chaque epoch se situe aux alentours de 26s avec un step de 160ms. Cela fait un total d'attente de 10 min, ce qui peut sembler long mais raisonnable par rapport à notre ensemble de données et la taille des images.

Nous obtenons au final une précision globale de **74%**, cette précision oscillant entre 70% et 75% pendant les derniers epochs. Pour l'apprentissage, la précision monte jusqu'à **90%** et le training loss diminue et arrive aux alentours de 0.3. Nous avons également essayé d'augmenter le nombre d'epochs, mais là où la précision de l'ensemble de test augmente encore, la précision de l'ensemble de validation stagne toujours entre 70 et 75%.

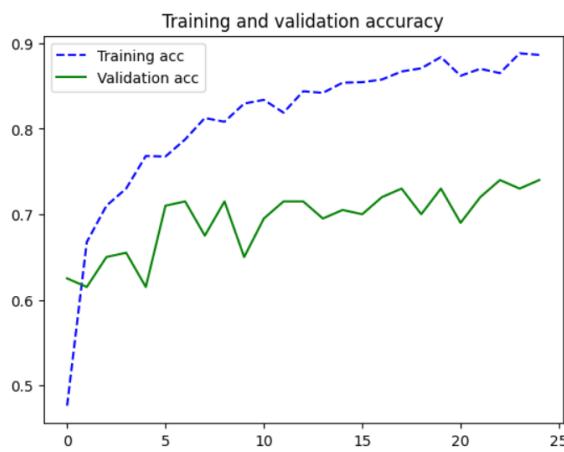


FIGURE 12 – Training and Validation accuracy

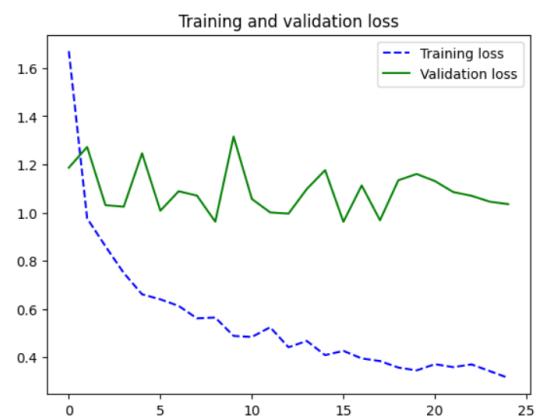


FIGURE 13 – Training and Validation loss

La précision par classe reste à 20-25% d'erreur pour la majorité des classes, mais certaines ont un plus grand taux d'erreur : les tagliatelles (30%), les fusilis (35%) et les maccheronis (40%).

74.0
[ [ 5. 20.]
[ 4. 20.]
[ 4. 20.]
[ 4. 20.]
[ 7. 20.]
[ 5. 20.]
[ 8. 20.]
[ 4. 20.]
[ 5. 20.]
[ 6. 20.] ]

FIGURE 14 – Precision globale et par classe : [Nombre d'erreurs, Nombre d'images ]

### 3.1.2 Matrice de confusion :

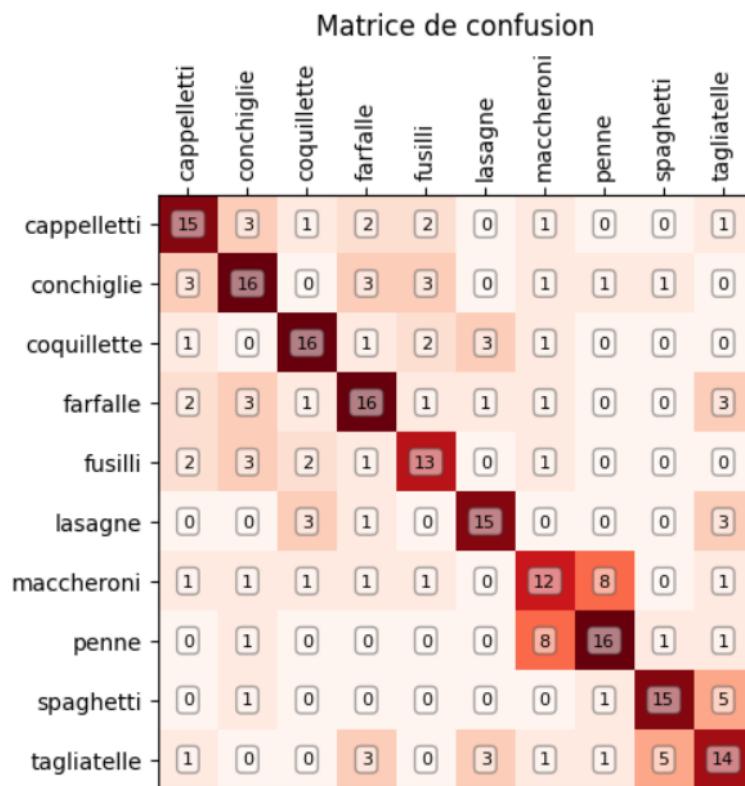


FIGURE 15 – Matrice de confusion

Nous pouvons constater si dessus la matrice de confusion qui affiche les correspondances et les erreurs entre les différentes classes. Si nous faisons la somme des colonnes sous une diagonale, nous obtenons bien 200 images, qui correspondent effectivement à nos 200 images de test.

Il y a également correspondance entre la diagonale de la matrice et le nombre d'erreurs de chaque classe (Figure 14) pour les images de tests : leur somme fait bien 20. (ex : cappelliti : 15 correspondances + 5 erreurs = 20 images de test).

D'autre part, nous pouvons observer qu'il y a une grande confusion entre les maccheroni et les pennes (2/5 de confusion) mais également entre les spaghetti et les tagliatelles (1/4 de confusion).

Nous analyserons qualitativement ces résultats dans la section suivante.

### 3.2 Analyse qualitative :

#### 3.2.1 Exemples :



FIGURE 16 – Confusion entre Maccheroni et Penne

Nous remarquons ici les confusions entre penne et maccheroni. En effet, ces pâtes sont presque identiques à l'oeil nu. Or, leur differences reposant presque entièrement sur la distinction des frontières cela explique la confusion existante entre ces deux classes dans notre réseau.

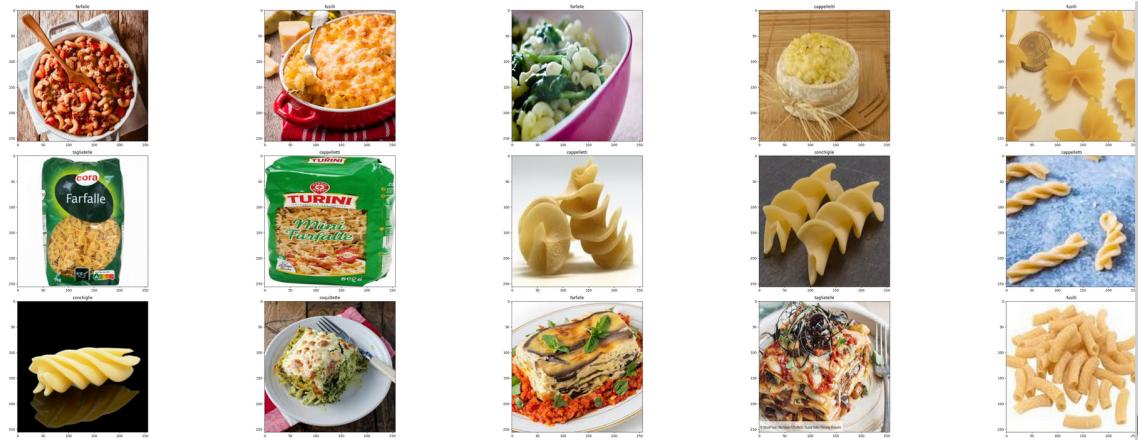


FIGURE 17 – Erreurs de labelisation

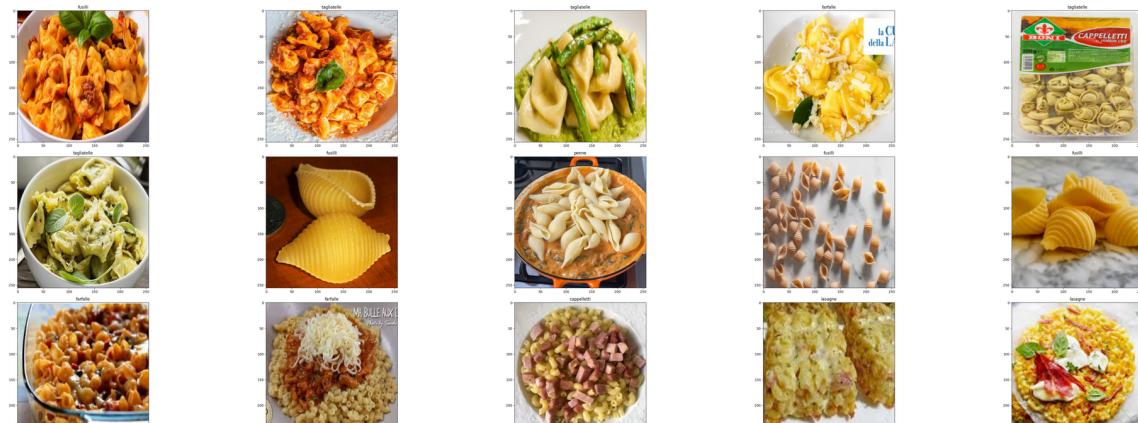


FIGURE 18 – Erreurs de labelisation

Nous remarquons que dans les images qui sont mal reconnues il y a des images qui sont difficilement reconnaissable même par des humains, le problème vient donc dans ce cas de notre base de donnée. Cependant il y a aussi des problèmes sur la reconnaissance de pâtes crues, sans sauce et en gros plan, qui sont donc très facilement reconnaissables. Dans ce cas-là, le problème vient peut-être que la base de données, qui ne contient pas assez de pâtes crues pour apprendre correctement.

### 3.2.2 Tentative de suppression des classes problématiques :

Comme nous avons remarqué que le réseau confondait les pennes et les maccheronis, nous avons décidé de tester l'entraînement du réseau en enlevant une des 2 classes pour vérifier si cela améliorerait les performances. Nous avons repris le réseau VGG 16 avec l'augmentation de données et nous l'avons fait apprendre sur 25 epochs. Nous obtenons la matrice de confusion de la figure 19.

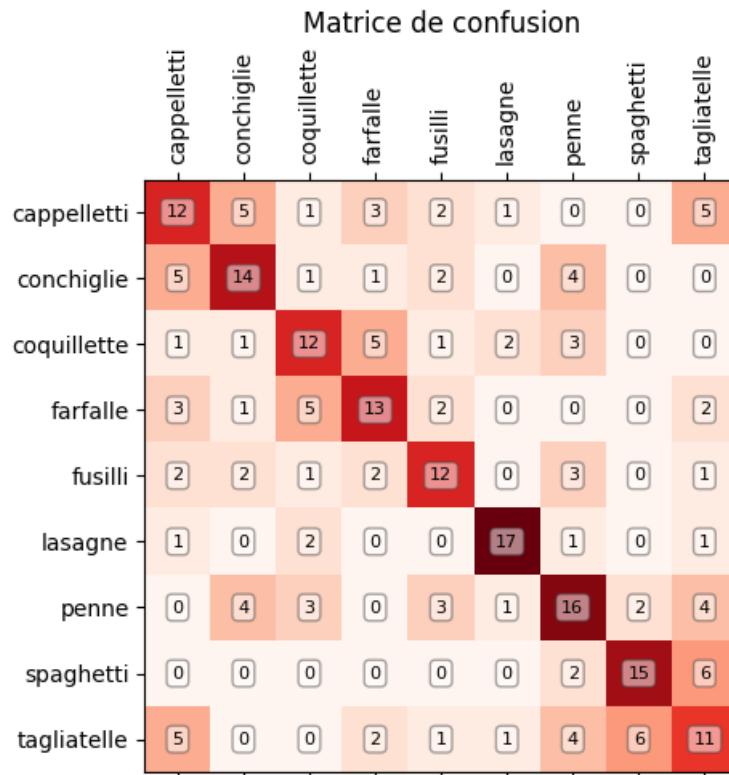


FIGURE 19 – Matrice de confusion avec modèle VGG et résolution de 128, sur l'ensemble de validation sans la classe maccheroni

Nous observons bien une précision plus importante pour les pennes, qui sont bien mieux reconnus. Il y a toujours des classes qui sont moins bien reconnues que d'autre, cependant, il y a une reconnaissance d'au moins 50% pour chacune des classes, ce qui est une nette amélioration. Le retrait de la classe problématique n'a pas eu vraiment d'impact sur les autres classes (sauf sur les pennes). Ce qui est positif, puisque nous avions peur en l'enlevant que l'apprentissage soit plus dur étant donné que la taille de la base de données diminuait. Cela ne s'est pas produit, la simplification en enlevant qu'une classe a compensé cette perte de données.

## 4 Conclusion :

### 4.1 Bilan et problèmes rencontrés :

Finalement, nous sommes assez satisfait de notre réseau au global étant donné que nous avons pu atteindre jusqu'à 75% de bonnes prédictions. Néanmoins, le problème principal qui nous a suivi durant tout le projet, repose sur le fait que puisque certaines formes de pâtes se ressemblent énormement (exemple : Penne/Maccheroni). Nous nous retrouvons avec des prédictions sur ces classes bien plus confuses qu'avec la précision globale.

### 4.2 Pistes d'amélioration :

Lors de ce projet, nous nous sommes souvent aperçus que notre base de données ne convenait pas complètement à nos attentes pour une base d'entraînement et de test.

En effet comme décrit dans le projet, nous avons dû souvent revenir sur celle-ci, changer les images pixellisées, doubler le nombre d'images, redimensionner en adaptant le zoom ou en retirant les zones inutiles pour se concentrer uniquement sur les pâtes. Nous avons également dû augmenter la qualité des images pour obtenir des résultats plus satisfaisants.

Nous pensons que nous pourrions encore peaufiner cette base de données car certaines pâtes sont, même pour un être humain, quasi impossibles à discerner, selon la quantité d'ingrédients dans le plat. Par conséquent, nos résultats par classe peuvent aussi beaucoup varier selon les tests (nous avons obtenu sur deux tests différents, une accuracy globale allant de 72.5% à 74% mais pour les maccheroni 40% d'erreurs (8/20) contre 75% d'erreurs (15/20) ).

	72.5
74.0	[[ 6. 20.]
[[ 5. 20.]	[ 4. 20.]
[ 4. 20.]	[ 9. 20.]
[ 4. 20.]	[ 3. 20.]
[ 4. 20.]	[ 4. 20.]
[ 7. 20.]	[ 3. 20.]
[ 5. 20.]	[ 15. 20.]
[ 8. 20.]	[ 1. 20.]
[ 4. 20.]	[ 1. 20.]
[ 5. 20.]	[ 9. 20.]
[ 6. 20.]]	

FIGURE 20 – Premier test avec 74% d’accuracy

FIGURE 21 – Deuxième test avec 72.5% d’accuracy

Comme nous l’avons décrit plus tôt, le problème de nos classes se reposait principalement sur la distinction des formes, étant donné que les pâtes ont la même couleur. Cependant, nous avons quelques images avec des pâtes de couleurs différentes (ou des sauces changeant un peu leur couleur), nous supposons qu’une piste d’amélioration pourrait être de tester en changeant la couleur des images (même si nous n’attendons pas de changement conséquent).