# RAPPORT PROJET ANNUEL RATTRAPAGE



**Sommaire**

[RAPPORT PROJET ANNUEL RATTRAPAGE 1](#_Toc113483708)

[I. Sujet et problématique 3](#_Toc113483709)

[II. Constitution du Dataset 3](#_Toc113483710)

[III. Modèles et analyse 4](#_Toc113483711)

[1. Création du MLP 5](#_Toc113483712)

[2. Parcours du dataset 6](#_Toc113483713)

[3. Analyses 7](#_Toc113483714)

[4. L’application 10](#_Toc113483715)

[IV. Conclusion 11](#_Toc113483716)

## Sujet et problématique

Reconnaissance d’image : l’objectif était de réaliser une application avec la reconnaissance d’un objet ou aliment. J’avais 3 choix possible de dataset, j’ai alors décidé de prendre le dataset des pizzas, tartes aux pommes et tartes aux fraises

Pour réaliser ce projet, j’ai décidé de reprendre mon MLP python du projet annuel. Je l’ai corrigé et modifié à plusieurs reprises afin de l’adapter à chaque cas de tests et au dataset. Enfin j’ai réalisé un site web ainsi qu’une petite application flask pour prédire les images.

## Constitution du Dataset

Comme dit dans le sujet, mon dataset possède 3 classes : pizzas, tarte aux pommes et tarte aux fraises. Afin de l’utiliser dans mon modèle, je l’ai divisé en 2, une catégorie train et une catégorie test. Dans la dernière version du MLP une catégorie validation est utilisée mais nous utilisons en réalité les mêmes données que les données de train.

Pour le constituer, j’ai utilisé diverses sources :

Dans un premier temps, j’ai utilisé Google Images. Je prenais les images une par une.

En plus de google Image, j’ai utilisé d’autres sources de données :

Une image contenant texte

Description générée automatiquement

J’ai utilisé tout d’abord le site images.cv afin d’essayer de recueillir des datasets de nourriture et d’y sélectionner les pizzas, les tartes aux pommes et les tartes aux fraises. Cependant je n’avais pas d’images intéressantes pour le projet. Les images étaient de mauvaises qualités.

J’ai donc décidé de procédé de la même manière avec Kaggle cette fois-ci



Pour rappel, Kaggle est une plateforme web organisant des compétitions en science des données appartenant à Google. Il y a un grand nombre de données dessus et donc de dataset.

A travers Kaggle, j’ai donc sélectionné que je trouvais intéressantes pour mon dataset et les ai ajoutées.

Ensuite j’ai utilisé diverses librairies d’images pour continuer d’agrandir mon dataset

Mon dataset est donc composé au total de 380 images.

## Modèles et analyse

Durant le premier projet, nous devions réaliser 4 modèles, dont le MLP. Comprenant que nous n’arrivions pas à l’implémenter from scratch, j’avais donc décidé de créer les modèles en partant du Framework Scikit-learn



Scikit-learn est une bibliothèque libre Python destinée à l'apprentissage automatique. Elle est développée par de nombreux contributeurs notamment dans le monde académique par des instituts français d'enseignement supérieur et de recherche comme

J’avais donc d’abord utilisé cette bibliothèque et ensuite je devais créer mes modèles en fonction de cette bibliothèque. Cela avait parfaitement fonctionné pour le modèle linéaire, mais concernant le MLP, c’était un échec. En effet, mon modèle ne parvenait qu’à réussir le cas de test XOR, mais ne réussissait aucun autre cas de test.

Durant ce projet de rattrapage, j’ai décidé de repartir sur ce modèle, le connaissant bien, et je l’ai donc débuggé.

### Création du MLP

Dans un premier temps, je suis donc reparti sur ce MLP. Je me suis aperçu au fur et à mesure de ce qu’il manquait. Tout d’abord, je n’avais qu’une fonction d’activation linéaire. De plus, je n’avais aucun paramètre permettant de spécifier si nous étions sur une classification ou non. J’ai donc alors rajouté ces différents paramètres et la fonction d’activation sigmoïde. Petit à petit, je suis parvenu à valider la majeure partie des cas de test donnés au début du projet.

J’ai alors décidé d’appliquer mon dataset à mon modèle. Mais j’ai dû faire face à plusieurs problèmes, notamment le fait que mon MLP ne fonctionnait pas avec un tableau Numpy et surtout n’allait pas fonctionner avec ma solution choisit qu’est le OneHotEncoder. J’ai donc dû créer une deuxième version de mon MLP afin répondre à ce besoin.

J’ai donc repris mon MLP pour l’adapter. Cela m’a pris un peu de temps. Lorsque j’ai eu terminé de mettre à jours mon MLP, je me suis dés le départ aperçu que certains cas de tests marchaient moins bien voire ne marchaient plus du tout. L’algo était souvent en sous-apprentissage, et cela même en modifiant les paramètres. Il était alors extrêmement compliqué de trouver les bons paramètres, et les bonnes fonctions d’activation à utiliser pour chaque cas. En effet, sur ce deuxième MLP j’avais mis à jour mes fonctions d’activation Linear et Sigmoïd et j’avais également rajouté la fonction d’activation Relu.   
Par la suite, j’ai essayé d’appliquer mon dataset à mon Modèle. Lorsque j’appliquais 2 classes uniquement, le modèle était en sous-apprentissage mais parvenait à apprendre en modifiant les paramètres. Seulement, avec 3 classes, le modèle ne parvenait toujours pas apprendre et cela quelque soit les paramètres. De plus, ce modèle n’utilisait aucune librairie comme Numpy et était donc extrêmement lent. Le traitement prenait plusieurs heures malgré une taille d’image réduite.  
J’ai alors rapidement compris que ce MLP ne pourrait fonctionner convenablement et j’ai donc pris la décision de recréer un MLP en utilisant celui déjà produit, et en y intégrant Numpy.

L’implémentation de ce nouveau modèle me pris un peu de temps. Après cela, je l’ai tout d’abord testé sur mes cas de tests. Il arrivait à bien apprendre et à bien prédire. L’erreur et l’accuracy étaient correctes.

J’ai alors testé le dataset, et malgré le fait que le modèle était plus rapide, il rencontrait toujours des difficultés à apprendre sur le dataset. Cela étant principalement dû au fait que les images se ressemblaient : en effet elles avaient la même forme et parfois des couleurs très proches. De plus, les images n’étant pas découpées, et donc certains objets à côté pouvaient fausser l’apprentissage. Par exemple une cuillère.

J’ai alors remodifié plusieurs fois mon MLP afin d’obtenir de meilleurs résultats, et j’ai revu certains paramètres.

Pour gagner du temps, j’ai développé une fonction qui allait exécuter l’entrainement mais avec différents paramètres de manière automatique. Cela permettait de facilement comparer les résultats et de prendre alors les meilleurs paramètres pour mon apprentissage.

J’ai fini par obtenir des résultats plus intéressants même s’ils sont encore améliorables. Je pourrais les améliorer en modifiant certains paramètres, et en rajoutant des images à mon dataset. Je n’ai pu réaliser ces améliorations, faute de temps.

### Parcours du dataset

J’ai créé une fonction prenant en paramètre un tableau de lien pour les images. Cela me permettait de facilement passer de 2 à 3 classes sans modifier ma fonction. Pour entrainer mon modèle et faire fonctionner mon apprentissage, j’ai utilisé la méthode OneHotEncoder.

Cela me permettait d’envoyer un tableau dans un tableau à mon MLP. Exemple : [1,0,0]

Ensuite j’ai créé une fonction dans mon notebook affichant pour chaque image, la prédiction ainsi que le pourcentage de précision. Ce pourcentage sert à démontrer à quel point le modèle est certain qu’une image est une pizza, une tarte aux pommes ou une tarte aux fraises.

J’ai également créé une fonction save dans mon MLP permettant de sauvegarder mon entrainement et de le réutiliser ultérieurement dans mon notebook ou dans une application.

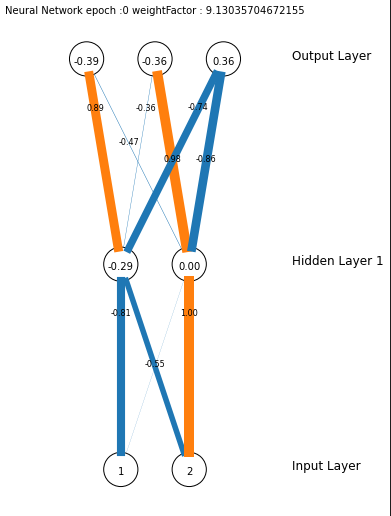
### Analyses

Afin d’évaluer l’efficacité de l’apprentissage de mon modèle, j’affichais dés le départ l’erreur afin de voir si celle-ci diminuait. Par la suite, j’ai rajouté l’accuracy.



Comme nous pouvons voir sur cet exemple, dans le cas présent, l’accuracy est à 1 ce qui signifie que le modèle ne se trompe pas sur les données

Par la suite, et afin de mieux comprendre comment fonctionnait mes réseaux de neurones, j’ai réalisé une représentation graphique de mon réseau de neurones



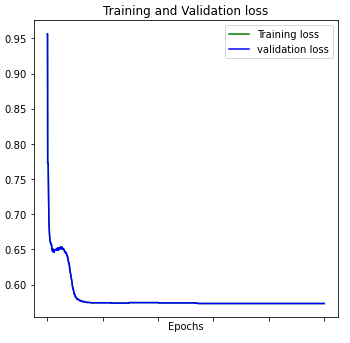
Concernant ce schéma : Tout en bas, nous avons la couche d’entrée, input layer. En haut, nous avons la couche de sortie, Output layer. Et entre les deux, se sont les couches cachées, Hidden layer. Chaque couche possède un certain nombre de neurones.

Chaque neurone est relié à tous les neurones de la couche inférieur par les liens pondérés par des poids. Se sont les chiffres en légende). Lorsque le trait est bleu, c’est que le poids est négatif. S’il est orange, alors le poids est positif.

L’activation d’un neurone se fait alors par la somme des signaux des neurones de la couche inférieure multipliés par les poids respectifs et plus le biais (nombre dans le neurone).

Cette somme passe ensuite par la fonction d'activation choisie pour la couche relu, Linear ou Sigmoïd. C'est le signal que ce neurone renverra lui-même à la couche supérieure.  Et ainsi de suite. A travers ce schéma, nous pouvons dire si un neurone est activé ou non si le nombre est positif. Cela permet alors de voir comment fonctionne notre réseau de neurone et peut aider à améliorer nos paramètres d’apprentissage

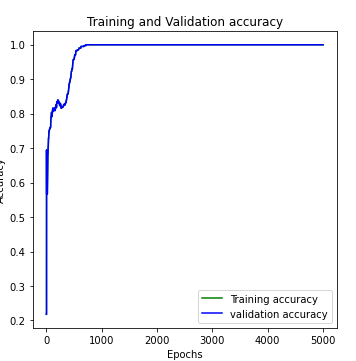
Ensuite, j’ai réalisé des courbes pour visualiser l’évolution de l’erreur :



Sur ce graphique, nous avons une courbe montrant l’évolution de notre erreur en fonction du nombre d’itérations (Epochs).

Ce que nous pouvons voir ici c’est que l’erreur baisse très rapidement et finit par ne plus baisser. Dans le cas présent de ce cas de test, l’accuracy était à 1. A savoir que la courbe validation loss et training loss reçoivent les mêmes données, ce qui explique qu’elles soient identiques.

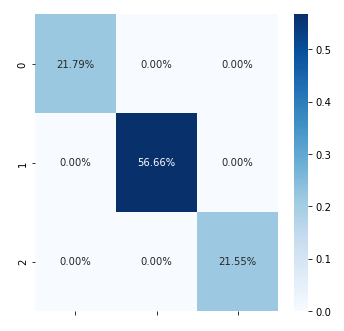
Ensuite, j’ai réalisé des courbes pour visualiser l’évolution de la précision :



Sur ce graphique, nous pouvoir voir la courbe montrant l’évolution de la précision de notre entrainement.

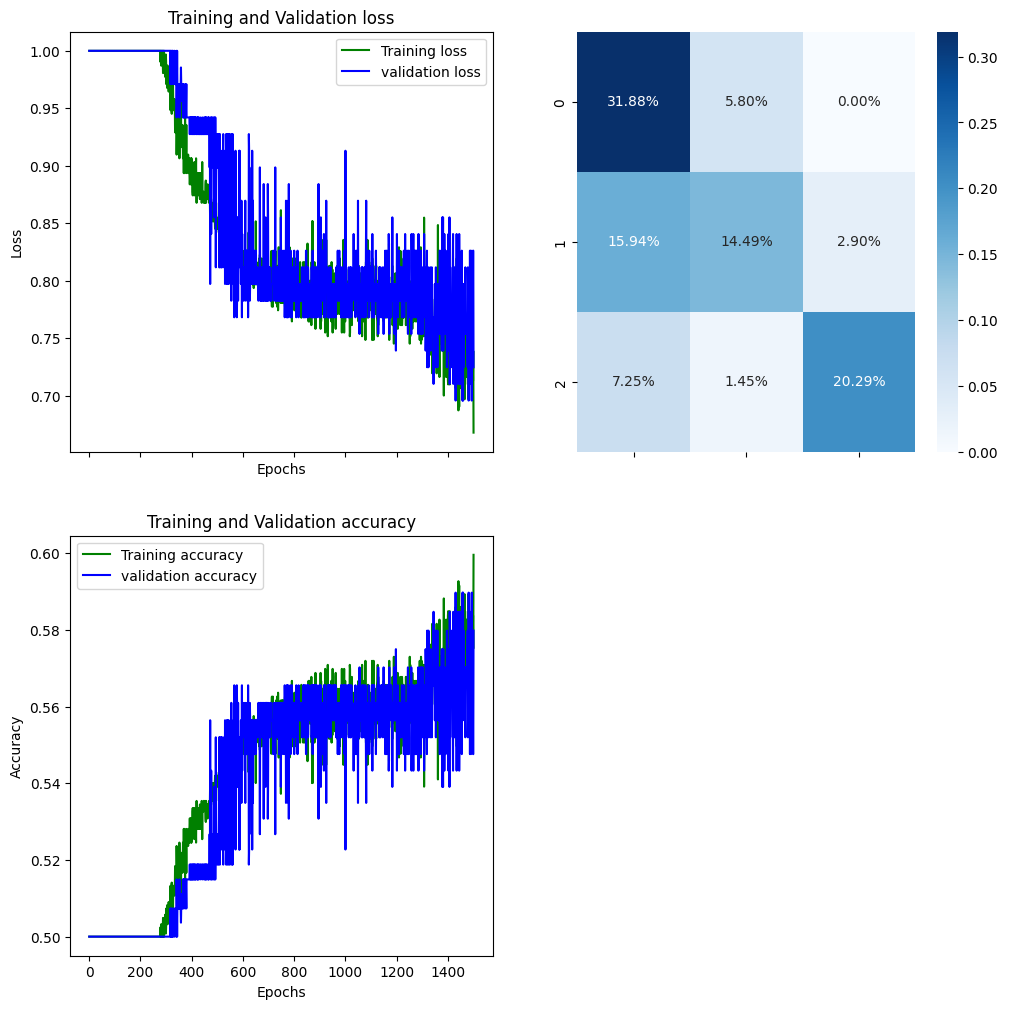
Ce que nous pouvons voir ici, c’est que l’accuracy augmente très rapidement pour atteindre 1.0 ce qui signifie que notre modèle ne se trompe pas.

Enfin, j’ai réalisé un schéma montrant la matrice de confusion :



Ce que nous pouvons voir ici c’est que ce schéma confirme notre accuracy de 1.0 puisque notre modèle ne se trompe pas. Tout est sur la diagonale, et aucune autre case n’est au-dessus de 0%

Jusqu’à présent, nous avons vu d’excellents résultats d’apprentissage, mais cela n’était pas toujours le cas comme nous pouvons le voir sur ces résultats d’entrainement de mon dataset :



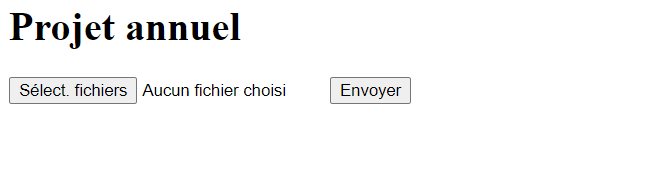
Comme nous pouvons voir, l’erreur baisse mais de manière très instable. Nous pouvons aussi voir que l’accuracy augmente mais également de manière instable. Nous voyons que notre modèle commet des erreurs même s’il arrive tout de même à prédire. C’est ce que j’ai essayé d’améliorer en modifiant les paramètres.

### L’application

Dans un premier temps, l’objectif était de réaliser une site web, permettant de présenter le projet et de tester des images sur notre apprentissage.

Seulement, le développement de ce MLP m’a pris beaucoup de temps, et je n’ai pas eu le temps d’ajouter l’application directement au site. J’ai donc opté pour une autre chose.

J’ai développé une petite API en python utilisant flask. Il n’y a qu’une fonction predict. Elle vérifie d’abord si des images ont été proposées. Dans le cas contraire, on affiche un formulaire permettant d’ajouter une ou des images. En effet, nous pouvons sélectionner plusieurs images à tester.



L’API va alors se servir des données d’apprentissage sauvegardées pour prédire les images. Ensuite, on nous renvoi un json affichant si c’est une pizza, une tarte aux pommes ou bien une tarte aux fraises.

L’objectif sera peut-être dans le futur d’ajouter l’application directement au site web et de ne pas passer par l’application flask.

## Conclusion

Après la déception du projet annuel, il était compliqué de se relancer dans un rattrapage de ce projet.

De plus, il nous fallait repartir de 0 pour le dataset et tout cela dans un délai assez restreint. Néanmoins je suis assez fier du rendu final de ce projet. En effet j’ai pu réaliser beaucoup de choses, et cela malgré le manque de temps et les différentes difficultés rencontrées durant tout le projet. Le MLP est fonctionnel, parvient à apprendre sur le dataset et à prédire et j’ai pu faire une petite application pour prédire une image.

Globalement, ce projet de rattrapage aura été compliqué à réaliser principalement à cause du délai.

Pour la suite, je songe à ajouter l’application directement au site web afin de faire une vraie implémentation de l’application et faire quelque chose de plus qualitatif. Et cela même si le projet est terminé.