SEBTI, CLOUP-MARTIN: TP MACHINE LEARNING

(MSE/ISMIN, 2A, 2023-2024)

Adam SEBTI, Timothée CLOUP-MARTIN
adam.sebti@etu.emse.fr, timothee.cloup@etu.emse.fr

February 13, 2024

1 The datasets: CIFAR-3 (4)

1.1 Question 1 (1)

Nous avons mesuré la taille des données contenues dans le dataset "CIFAR-3".

- **X_gray : (18000, 32, 32)**
- **X : (18000, 32, 32, 3)**
- **Y : (18000)**

18000 correspond à la taille du dataset, i.e. le nombre d'images. Chaque image est de taille 32×32 et le 3 correspond aux couleurs RGB pour le dataset avec les images colorées.

La valeur maximale d'un pixel est **255** et la valeur minimale est **0**.

Il est préférable de normaliser les données sur **[0;1]** car cela améliore les performances et la stabilité de l'entraînement du modèle.

1.2 Question 2 (2)

Il est indispensable de séparer la base de donnée entre une base d'entraînement et une base de validation. La base d'entraînement va permettre au modèle d'apprendre à distinguer les images, mais il ne faut pas tester son modèle avec la base d'entraînement car cela introduirait un biais dans l'évaluation.
La base de validation permet de tester notre modèle en conditions réèlles.

1.3 Question 3 (1)

Les bases de données d'entraînement et de test sont bien équilibrées. Cet équilibre est indispensable afin que modèle ait les mêmes peformances pour chaque classes.

```
Classes in Y_train

class_0 4766

class_1 4828

class_2 4806

==================

Classes in Y_test

class_0 1159

class_1 1220

class_2 1221
```

Figure 1: Variance en fonction de la réduction de dimension

2 Dimensionality reduction with the PCA (4)

2.1 Question 1 (1)

Sachant que la dimension original d'une image de X_gray est $32 \times 32 = 1024$, nous avons réalisé la PCA avec les nombres de composantes suivants : 50,100,200,400,1000

2.2 Question 2 (2)

Le but de la réduction de dimension est de rendre le modèle plus rapide. L'objectif est de garder un bon compromis entre la rapidité du modèle et ses performances. Plus un modèle est performant, plus sa variance est élevée. Ainsi, nous avons tracé la variance du modèle en fonction du nombre de composant après réduction.

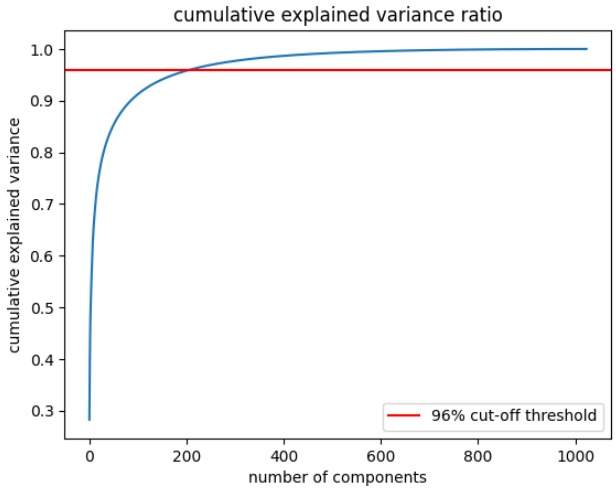


Figure 2: Variance en fonction de la réduction de dimension

On remarque qu'avec un nombre de composante égal à 209, on conserve 96% de la variance à l'origine. En revanche on a divisé par 5 le nombre de composantes. Le système sera bien plus rapide et les performances sont conservées.

2.3 Question 3 (1)

Voici ci-dessous les images restituées après une transformation inverse en fonction du nombre de composantes choisit.

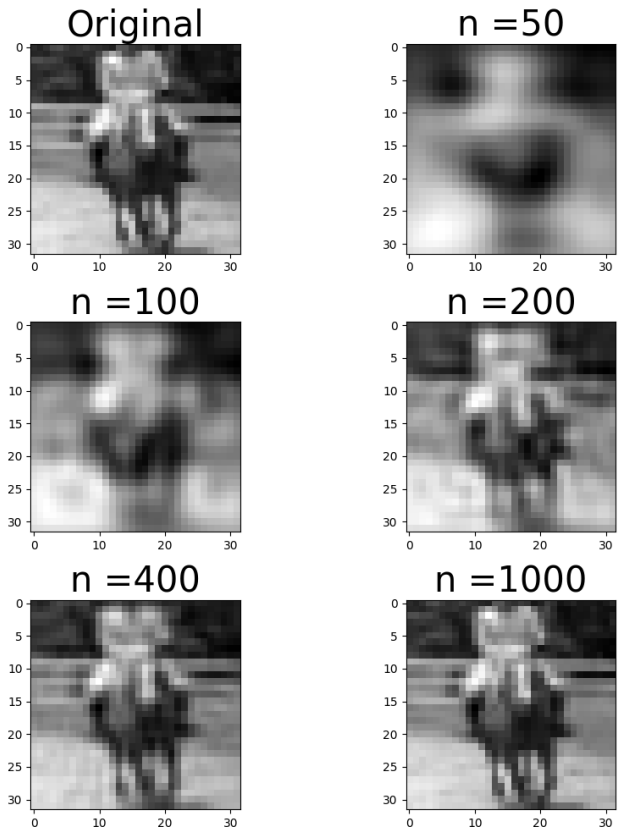


Figure 3: Variance en fonction de la réduction de dimension

On voit bien qu'a partir de n=200, l'image restituée est quasi identique à l'image à l'origine. On a perdu très peu d'information malgré la réduction de dimension. En revanche les images obtenue après une réduction vers les dimensions n=50 ou n=100 sont inexploitables. Finalement, la réduction choisie à la question précédente semble bien utilisable pour réaliser notre modèle.

3 Supervised Machine Learning (28)

3.1 Logistic Regression, Gaussian Naive Bayes Classifier (6)

3.1.1 Question 1 (1)

La régression logistique est un modèle linéaire qui utilise la fonction logistique pour modéliser la probabilité d'appartenance à une classe. Le classifieur naïf bayésien est une méthode non linéaire qui s'appuie sur le théorème de Bayes et l'hypothèse naïve d'indépendance conditionnelle entre les paramètres (dans notre cas il s'agit des pixels). Soit X la base d'entraînement (entrées) et Y le vecteur regroupant les classes (sorties). La méthode du classifieur naïf bayésien consiste à déterminer la probabilité P(X|Y) afin d'en déduire par la relation de Bayes P(Y|X). La méthode de régression logistique consiste à déterminer directement P(Y|X).

3.1.2 Question 2 (2)

Voici ci-dessous les résultats obtenus :

```
Logistic Regression with uncompressed dataset

Testing Accuracy : 0.5937777777777777

Training Accuracy :0.6403703703703704

              precision    recall  f1-score   support

           0       0.68      0.69      0.68      1436
           1       0.55      0.61      0.58      1557
           2       0.55      0.49      0.52      1507

    accuracy                           0.59      4500
   macro avg       0.59      0.60      0.59      4500
weighted avg       0.59      0.59      0.59      4500
```

(a) Résultats de l'entraînement du modèle de regression logistique

```
Gaussian Naïve Bayes Classifier with uncompressed dataset

Testing Accuracy : 0.584

Training Accuracy :0.583925925925926

              precision    recall  f1-score   support

           0       0.64      0.64      0.64      1436
           1       0.54      0.73      0.62      1557
           2       0.58      0.38      0.46      1507

    accuracy                           0.58      4500
   macro avg       0.59      0.58      0.57      4500
weighted avg       0.59      0.58      0.57      4500
```

(b) Résultats de l'entraînement du modèle de classification naïf bayésen

Figure 4: Résultats d'entraînement

Les deux modèles ont une précision de validation proche de 60%. Pour améliorer nos modèles, on pourrait modifier des paramètres tels que le paramètre de lissage pour le classifieur naif bayésien et le paramètre de pénalité pour régulariser le modèle dans le cas de la régression logistique. Cependant ces modifcations ne permettraient pas d'obtenir des résultats assez satisfaisants. Nous allons par la suite ré-entraîner les modèles après avoir réalisé une transformée PCA vers une dimension n=209.

3.1.3 Question 3 (1)

Il est important de tester notre modèle à la fois sur l'échantillon de test et de validation. De cette façon on vérifie déjà que l'apprentisstage de notre modèle soit correct (underfit) mais en plus il nous faudra tester ses performances sur des nouvelles données qui lui sont inconnues pour vérifier qu'il n'ait pas trop calqué les données d'entraînement (overfit). Ainsi tester le modèle de la sorte permet d'évaluer sa performance et surtout sa capacité à généraliser ses prédictions sur toute donnée nouvelle.

3.1.4 Question 4 (2)

```
Logistic Regression with compressed data by PCA method

Testing Accuracy : 0.6142222222222222

Training Accuracy :0.6256296296296296

              precision    recall  f1-score   support

           0       0.70      0.73      0.71      1505
           1       0.57      0.62      0.59      1513
           2       0.57      0.50      0.53      1482

    accuracy                           0.61      4500
   macro avg       0.61      0.61      0.61      4500
weighted avg       0.61      0.61      0.61      4500
```

(a) Résultats de l'entraînement du modèle de régression logistique

```
Gaussian Naïve Bayes Classifier with compressed dataset

Testing Accuracy : 0.6104444444444445

Training Accuracy :0.6091111111111112

              precision    recall  f1-score   support

           0       0.68      0.66      0.67      1505
           1       0.59      0.74      0.66      1513
           2       0.55      0.42      0.48      1482

    accuracy                           0.61      4500
   macro avg       0.61      0.61      0.60      4500
weighted avg       0.61      0.61      0.60      4500
```

(b) Résultats de l'entraînement du modèle de classification naïf bayésien

Figure 5: Résultats d'entraînement

Les deux nouveaux modèles avec une base de donnée compressée ont une précision de validation très légerement supérieure aux modèles précédents. Ces résultats ne sont cependant toujours pas satisfaisant. Nous allons devoir essayer d'autres méthodes de classification.

3.2 Deep Learning: MLP (13)

3.2.1 Question 1 (1)

La taille du tenseur en entrée est (32,32) pour un élement du dataset X_gray qui correspond à la taille d'une image. En sortie la taille du tenseur est de 3, i.e. le nombre de classes (voiture, cheval, cerf).

3.2.2 Question 2 (1)

Pour notre modèle MPL nous avons choisi **epochs = 100**. Le nombre d'epochs représente le nombre de fois que notre modèle aura utilisé la totalité des données d'entraînement pour son apprentisstage.
Nous avons fixé le paramètre **batch = 256**. Ce nombre représente le nombre de lots dans lequel le dataset est subdivisé à chaque entraînement. L'apprentistage se fait par "paquets" dont la taille est égale au batch.

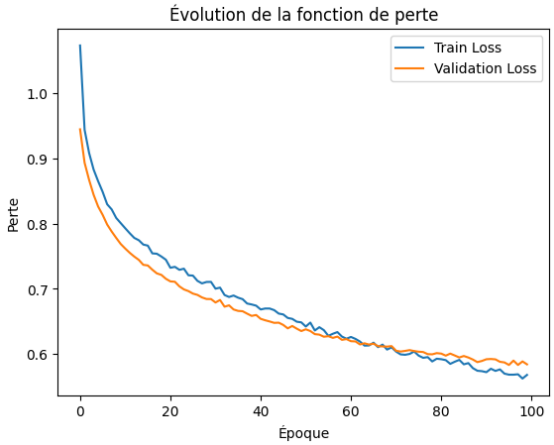
3.2.3 Question 3 (1)

Le dataset de validation est primordial lors de la création d'un modèle d'IA (MLP) car il permet tout d'abord de tester la performance du modèle et vérifier si nous avons de l'overfitting/underfitting.
Ceci est possible car le validation set comprend des données qui sont inconnues au modèle qui va les découvrir pour la première fois.

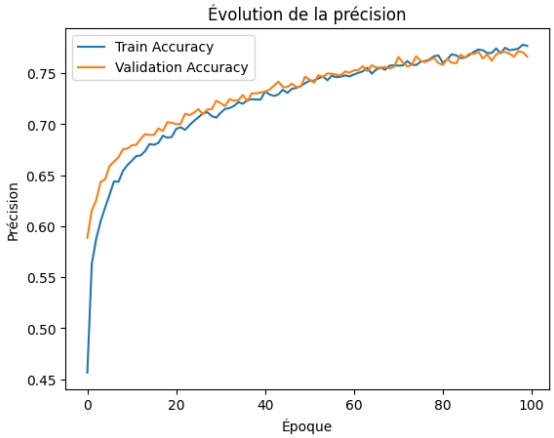
3.2.4 Question 4 (2)

- **Taux d'apprentisstage :** c'est le paramètre **adam** qui l'influence directement, il agit sur le pas avec lequel on effectue la descente de gradient.
- **Dropout** : ce paramètre represente la probabilité que chaque neurone soit éteint lors d'un apprentisstage donné. Le réglage de paramètre permet de contrer de manière efficace l'overfitting.
- **Nombre de neurones :** Il représente la capacité de notre système, il faut choisir un nombre de neurones cohérent avec la taille de la couche d'entrée et de la couche de sortie.

3.2.5 Question 5 (1)



(a) Courbe de précision lors de la phase d'entraînement et de validation



(b) Courbe de perte lors de la phase d'entraînement et de validation

Figure 6: Résultats du modèle MLP

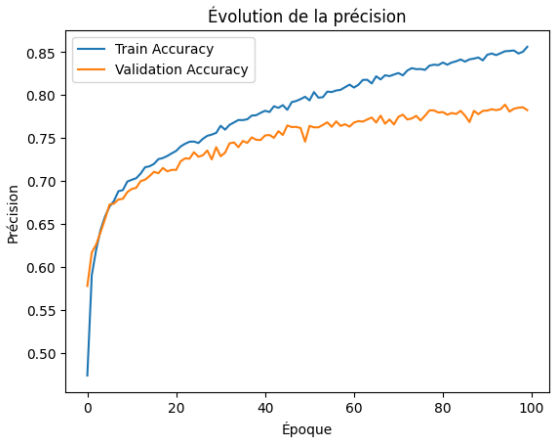
```
Train loss : 0.49080365896224976

Train accuracy : 0.8145833611488342

Validation loss : 0.5841126441955566

Validation accuracy : 0.7661111354827881
```

Figure 7: Résultat de l'entraînement du modèle MLP

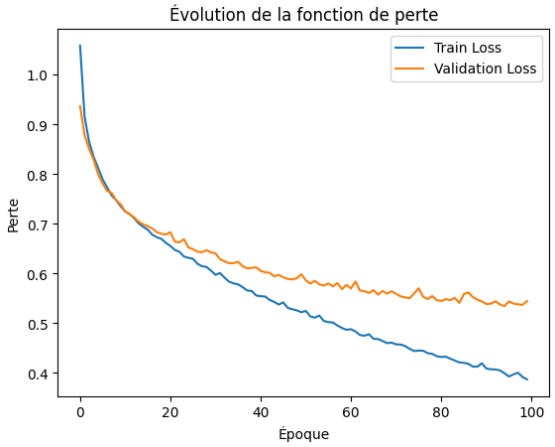
Les résultats montrent que le modèle n'est pas suffisant pour prédire avec précision la catégorie de toutes les images, la précision n'est que de 75%. Toutefois le modèle ne présente ni des problèmes d'overfitting ni d'underfitting.

3.2.6 Question 6 (4)

Le modèle précedemment entraîné ne présente pas de problèmes d'overfitting. Nous allons donc intentionnellement modifier un paramètre lors de l'entraînement d'un nouveau modèle afin de provoquer l'overfitting et examiner les répercussions sur les résultats obtenus. Pour accentuer l'overfitting, il faut minimiser la régularisation du modèle. Dans le cadre du modèle MLP, le paramètre de régularisation est le taux de dropout. Voici ci-dessous les résultats obtenus pour un taux de dropout de 10% :



(a) Courbe de précision lors de la phase d'entraînement et de validation



(b) Courbe de perte lors de la phase d'entraînement et de validation

Figure 8: Résultats du modèle MLP

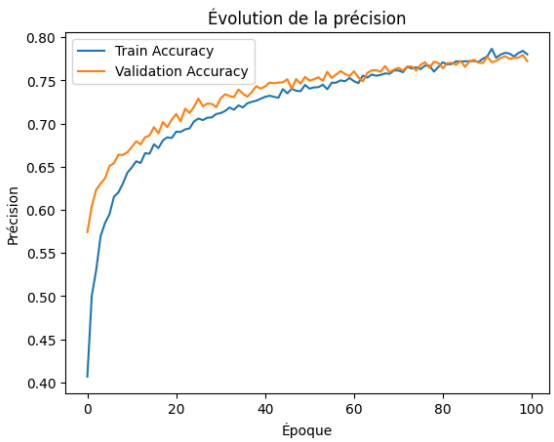
```
Train loss : 0.3811133801937103

Train accuracy : 0.8593055605888367

Validation loss : 0.5446105599403381

Validation accuracy : 0.7825000286102295
```

Figure 9: Résultat de l'entraînement du modèle MLP

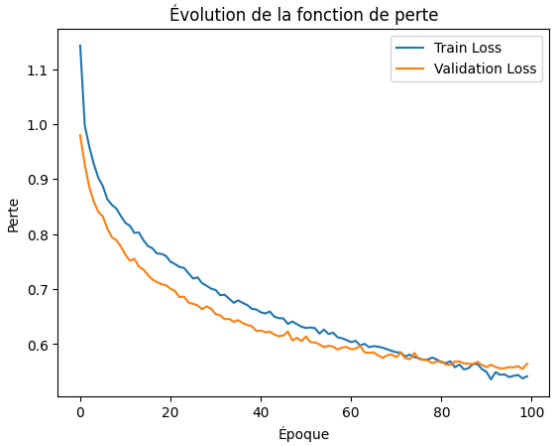
L'effet d'overfitting est très clairement perceptible sur les deux courbes. Cet effet traduit le fait que le modèle s'est surentraîné et ne s'est pas généralisé. Le modèle va comparer les données de validation avec les données d'entraînement dans les moindres détails. C'est pourquoi on retrouve au bout de 100 époques un écart considérable entre la courbe de training loss et celle de data loss.

3.2.7 Question 7 (3)

Avec ce nouveau modèle, nous avons rajouté une couche intermédiaire et réadapté les hyperparamètre à savoir le taux de dropout et le taux d'apprentissage via le paramètre ADAM.



(a) Courbe de précision lors de la phase d'entraînement et de validation



(b) Courbe de perte lors de la phase d'entraînement et de validation

Figure 10: Résultats du modèle MLP amélioré

```
Train loss : 0.3811133801937103

Train accuracy : 0.8593055605888367

Validation loss : 0.5446105599403381

Validation accuracy : 0.7825000286102295
```

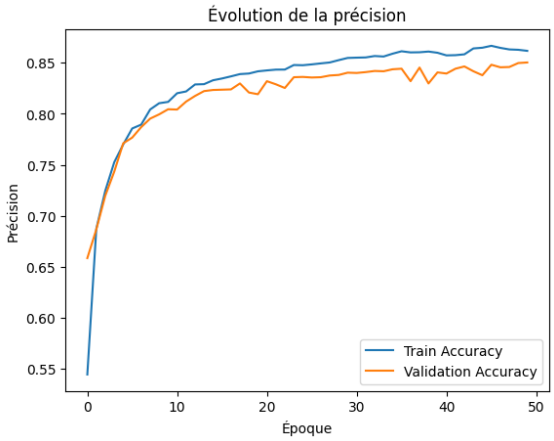
Figure 11: Résultat de l'entraînement du modèle MLP amélioré

3.3 Deep Learning: CNN (11)

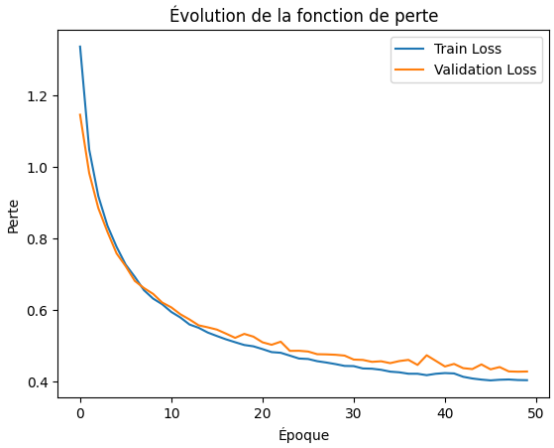
3.3.1 Question 1 (1)

La taille d'un tenseur d'entrée est (32,32,3) avec le modèle CNN. Elle diffère du modèle précédent parce que l'on prend cette fois-ci la base de donnée composée d'images colorées (3 couleurs RGB).

3.3.2 Question 2 (1)



(a) Courbe de précision lors de la phase d'entraînement et de validation



(b) Courbe de perte lors de la phase d'entraînement et de validation

Figure 12: Résultats du modèle CNN

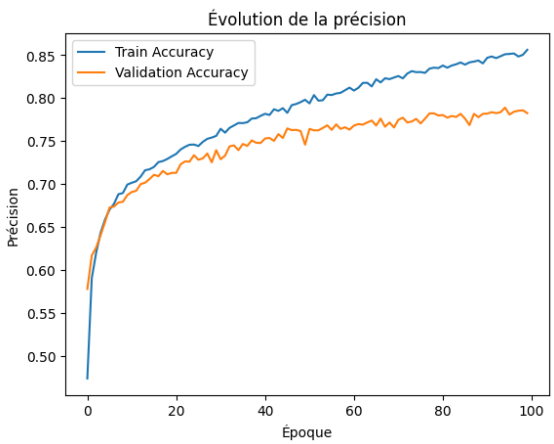
```
Train loss : 0.39384403824806213

Train accuracy : 0.8687499761581421

Validation loss : 0.4271392822265625

Validation accuracy : 0.8502777814865112
```

Figure 13: Résultat de l'entraînement du modèle CNN

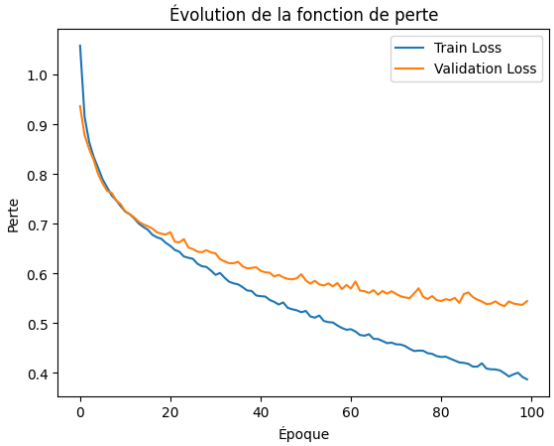
Les résultats sont très satsfaisant. On n'observe pas de phénomènes d'overfitting (surajsutement) ni d'underfitting (sousajsutement). La précision du test atteint 86% ce qui est très satisfaisant.

3.3.3 Question 3 (4)

Étant donné l'absence d'overfitting dans nos expériences précédentes, nous allons ajuster un paramètre lors de l'entraînement d'un nouveau modèle afin de délibérément induire un phénomène d'overfitting et voir ses conséquences sur les résultats. Pour accentuer l'overfitting, il faut minimiser la régularisation du modèle. Dans le cadre du modèle CNN, le paramètre de régularisation est le paramètre de pénalité L2. Voici ci-dessous les résutlats obtenus avec le paramètre de pénalité L2 nul :



(a) Courbe de précision lors de la phase d'entraînement et de validation



(b) Courbe de perte lors de la phase d'entraînement et de validation

Figure 14: Résultats du modèle CNN

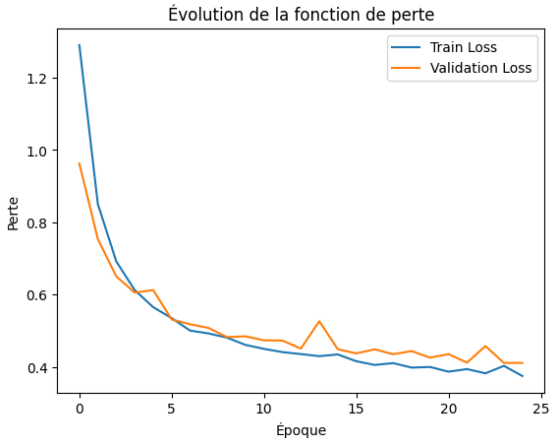
```
Train loss : 0.3811133801937103

Train accuracy : 0.8593055605888367

Validation loss : 0.5446105599403381

Validation accuracy : 0.7825000286102295
```

Figure 15: Résultat de l'entraînement du modèle CNN

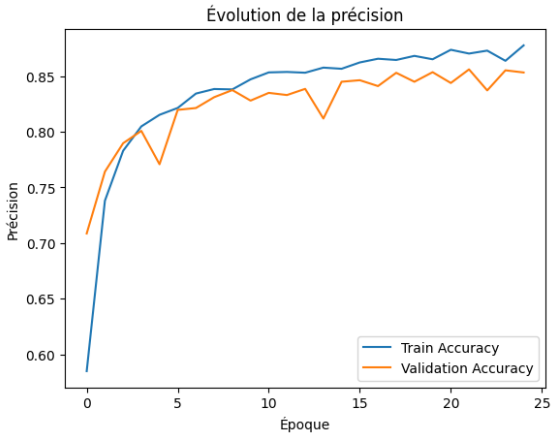
L'impact de l'overfitting est nettement discernable sur les deux courbes. Après 100 époques, on observe une disparité significative entre la courbe de training loss et celle de testing loss.

3.3.4 Question 4 (3)

Nous avons essayé tout d'abord d'ajouter une couche de convolution supplémentaire, mais sans un gain significatif de précision.
Nous nous sommes centré sur les hyperparamètres de la penalité et adam en les réglant plus finement.



(a) Courbe de précision lors de la phase d'entraînement et de validation



(b) Courbe de perte lors de la phase d'entraînement et de validation

Figure 16: Résultats du modèle CNN

Malgré les modifcations on atteint seulement une précision de 87%, de plus, la limitation de la puissance de calcul de nos appareils restreint le nombre d'essais possibles en raison de leur grande durée.

3.3.5 Question 5 (2)

Au cours de ce TP, nous avons créé un modèle de deep learning en combinant à la fois MLP et CNN. Les modèles créés avec CNN se sont avérés être plus performants dans la tâche de classification d'images. Cette différence de performance s'explique par le fait que CNN parvient à extraire davantage d'informations des images, notamment en ce qui concerne la couleur, ce qui se traduit par une meilleure précision tant lors de l'entraînement que lors de la validation, sans toutefois entraîner de surapprentissage. L'inconvénient du CNN réside dans son temps d'entraînement plus long par rapport à celui du MLP.