

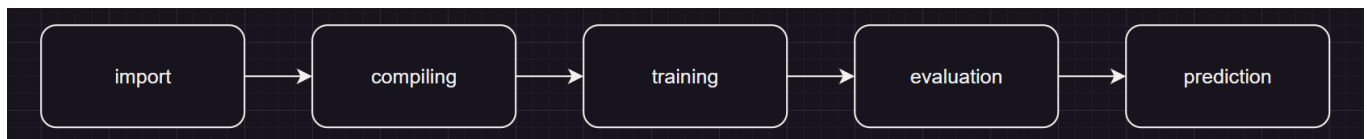
# MNIST

## Explication de haut niveau

- Le but principal du modèle est de reconnaître les chiffres de 0 à 9 avec la plus grande précision possible.
  - Pour ce faire, nous avons du programmer en python, lire de la documentation sur le sujets, reviser les notes de cours pour s'assurer d'avoir une vision globale des AI et leurs modeles
- 
- 

## La base de donnees MNIST

- La base de données MNIST (Modified NIST) comprend 60 000 images d'entraînement et 10 000 images de test.
- Les images de test peuvent être considérées comme des "réponses" et ne devraient jamais être jointes aux images d'entraînement, sans quoi cela fausserait l'estimation de la précision.
- Cela reviendrait à l'équivalent de l'analogie suivante :
  - Donner les réponses à un élève que l'on essaie de tester sur ses connaissances.
  - L'élève n'apprendrait pas à effectuer de nouveaux liens, car il n'a pas appris, mais seulement mémorisé la matière.
- **Le "pipeline" de données ressemble à l'image suivante :**



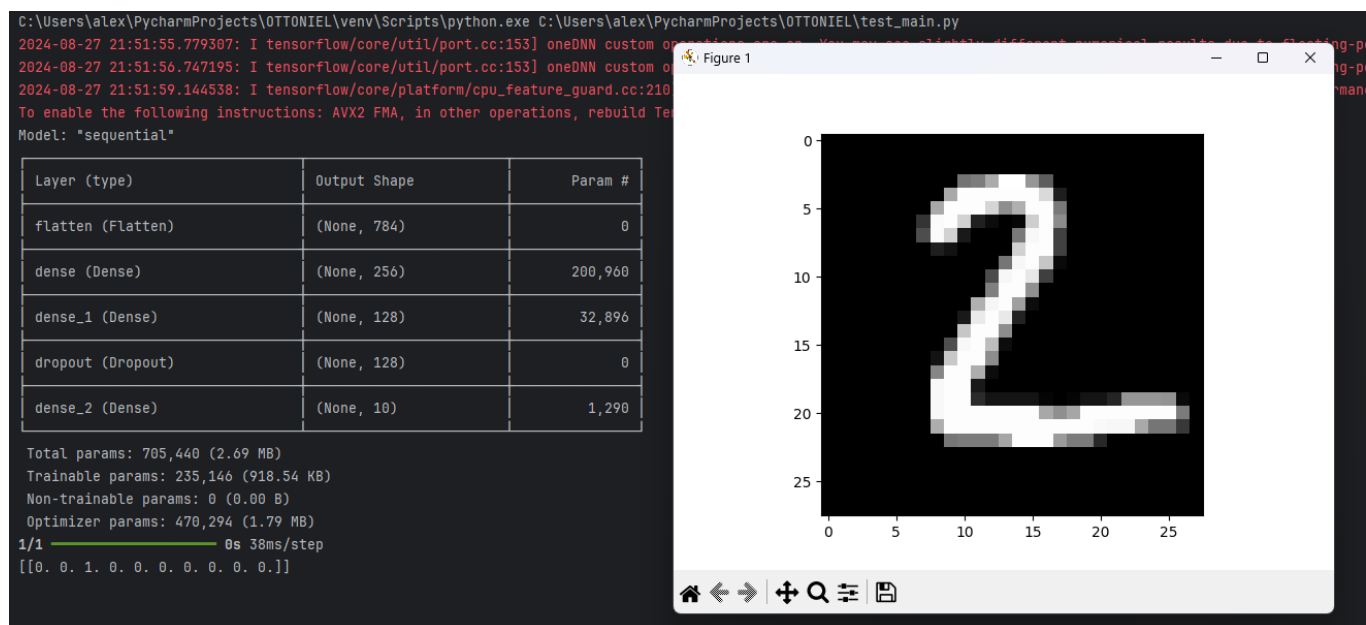
---

## Les prédictions du modèle et leurs résultats

1. Plusieurs paramètres sont en mesure d'affecter la prédiction du modèle et son apprentissage.
2. J'ai décrit ces paramètres sous les résultats.

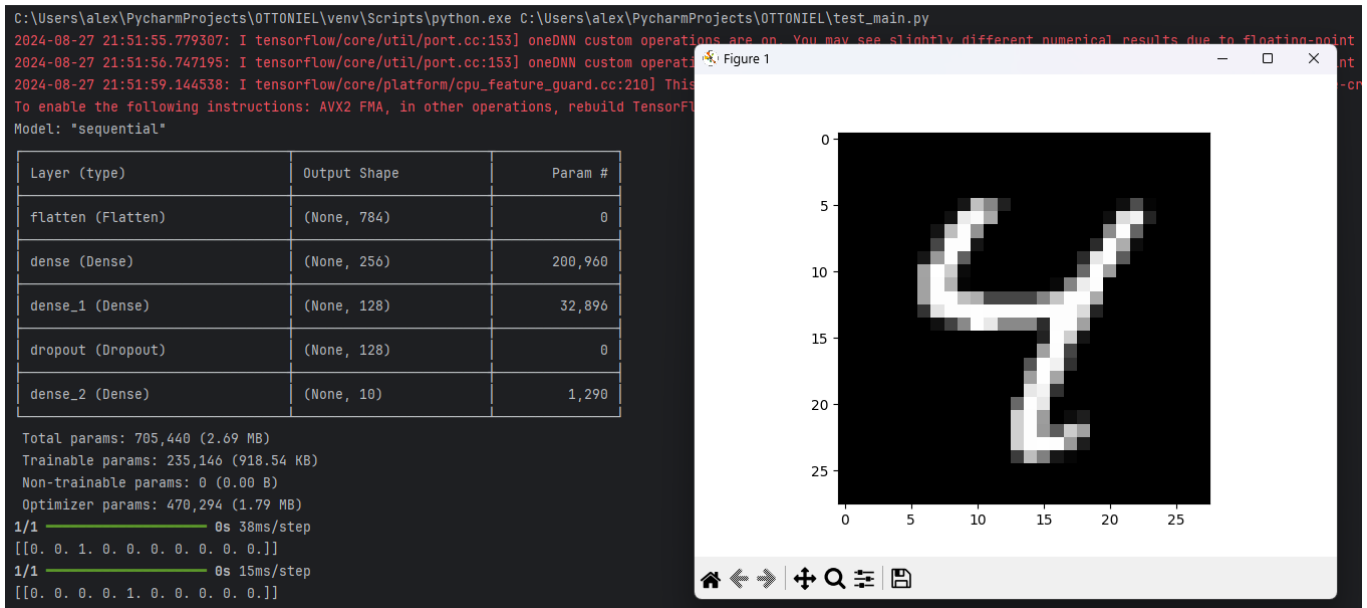
- **Voici les résultats que nous avons obtenus.**

### x\_test[1]



- Le résultat ci-dessus est précis et correct dans sa prédiction, le chiffre affiché est bien le 2, continuons aux prochains exemples.
-

## x\_test[6]



- Le résultat de "x\_test[6]" est 4, on peut voir que la 5ème position est enclenchée (1), ce qui signifie que le modèle a une fois de plus bien prédit.

## x\_test[3513]



- Encore une fois, le modèle a prédit de façon précise les chiffres qui apparaissent à l'écran.
- C'est long de devoir confirmer les résultats un par un, mais c'est nécessaire.
- Autrement, notre entraînement pourrait s'en trouver compromis et mener à une baisse de précision.

## x\_test[145]

To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.

Model: "sequential"

Layer (type)	Output Shape	Param #
flatten (Flatten)	(None, 784)	0
dense (Dense)	(None, 256)	200,960
dense_1 (Dense)	(None, 128)	32,896
dropout (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 10)	1,290

Total params: 705,440 (2.69 MB)

Trainable params: 235,146 (918.54 KB)

Non-trainable params: 0 (0.00 B)

Optimizer params: 470,294 (1.79 MB)

1/1 — 0s 38ms/step

[[0. 0. 1. 0. 0. 0. 0. 0. 0.]]

1/1 — 0s 15ms/step

[[0. 0. 0. 0. 1. 0. 0. 0. 0.]]

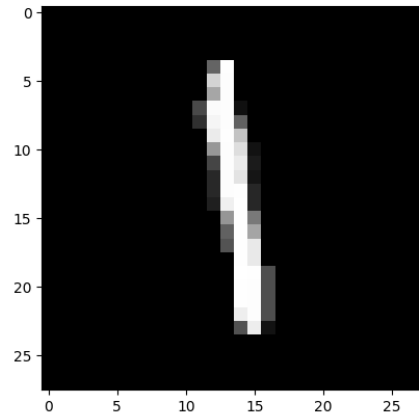
1/1 — 0s 33ms/step

[[0. 0. 1. 0. 0. 0. 0. 0. 0.]]

1/1 — 0s 14ms/step

[[0. 1. 0. 0. 0. 0. 0. 0. 0.]]

Figure 1



- Le deuxième neurone qui représente le 1 est activé, le modèle est précis.

## x\_test[458]

Model: "sequential"

Layer (type)	Output Shape	Param #
flatten (Flatten)	(None, 784)	0
dense (Dense)	(None, 256)	200,960
dense_1 (Dense)	(None, 128)	32,896
dropout (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 10)	1,290

Total params: 705,440 (2.69 MB)

Trainable params: 235,146 (918.54 KB)

Non-trainable params: 0 (0.00 B)

Optimizer params: 470,294 (1.79 MB)

1/1 — 0s 38ms/step

[[0. 0. 1. 0. 0. 0. 0. 0. 0.]]

1/1 — 0s 15ms/step

[[0. 0. 0. 0. 1. 0. 0. 0. 0.]]

1/1 — 0s 33ms/step

[[0. 0. 1. 0. 0. 0. 0. 0. 0.]]

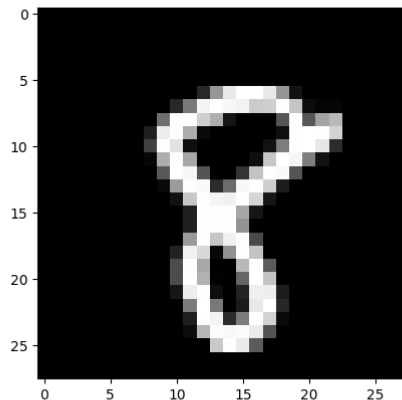
1/1 — 0s 14ms/step

[[0. 1. 0. 0. 0. 0. 0. 0. 0.]]

1/1 — 0s 14ms/step

[[0. 0. 0. 0. 0. 0. 0. 0. 1.]]

Figure 1



- Le cinquième test s'achève avec brio, le 9ème neurone est activé et l'image à l'écran est bien un 8.

- 
- Plus bas, plusieurs concepts ont été séparés et expliqué en contexte avec notre projet.
- 

## "Normalisation"

1. La normalisation permet de convertir les pixels de nos "Training Data" (une fois aplatis) de 0 à 1 plutôt que de 0 à 255.
  2. Sans la normalisation, le modèle pourrait avoir des difficultés à apprendre dans les cas où le delta ( $\Delta$ ) entre les différentes valeurs (pixels) est tellement grand que le modèle ne les reconnaît pas comme importantes ou importantes lorsqu'elles ne le sont pas, ce qui peut fausser les données.
  3. Cela permet également de préserver de l'espace au niveau des données (Poids du modèles).
- Concrètement, il est plus facile de différencier le blanc du noir que de différencier plusieurs teintes de gris, c'est également plus rapide! (Plus performant)
- 

## "Dropout Layer"









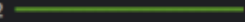


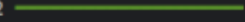

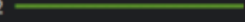
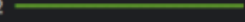
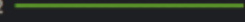









- La couche de dropout permet de désactiver temporairement certains neurones afin de permettre au modèle de s'entraîner de manière plus robuste.
- Le "Dropout Layer" fonctionne de la façon suivante :
  1. Un pourcentage de neurones est désactivé de manière aléatoire.
  2. Les informations d'entraînement passent par les autres connexions neuronales.
  3. Cela permet aux autres neurones (qui auraient potentiellement été sous-utilisés) de développer leur plein potentiel.
  4. Les neurones qui on "Dropout" changent à chaque "Epoch" (aléatoirement).
- Le "Dropout Layer" est considerer comme un "hidden layer" car il est dans le sandwich de notre modeles
-


## Analogies :

- Quelques analogies pour montrer notre compréhension :
    1. L'ajout d'un handicap à un athlète lors de son entraînement (Goku)
    2. Un aveugle qui, avec le temps, apprend à aiguïser ses autres sens (Daredevil)
    3. Un chemin peu exploré, mais que l'on se rappelle de mieux en mieux à chaque nouvelle exploration.
  - L'objectif ultime de cette technique est d'aider le réseau neuronal à réagir de manière appropriée lorsqu'il est confronté à de nouvelles informations (haute précision).
- 

## "Epochs"

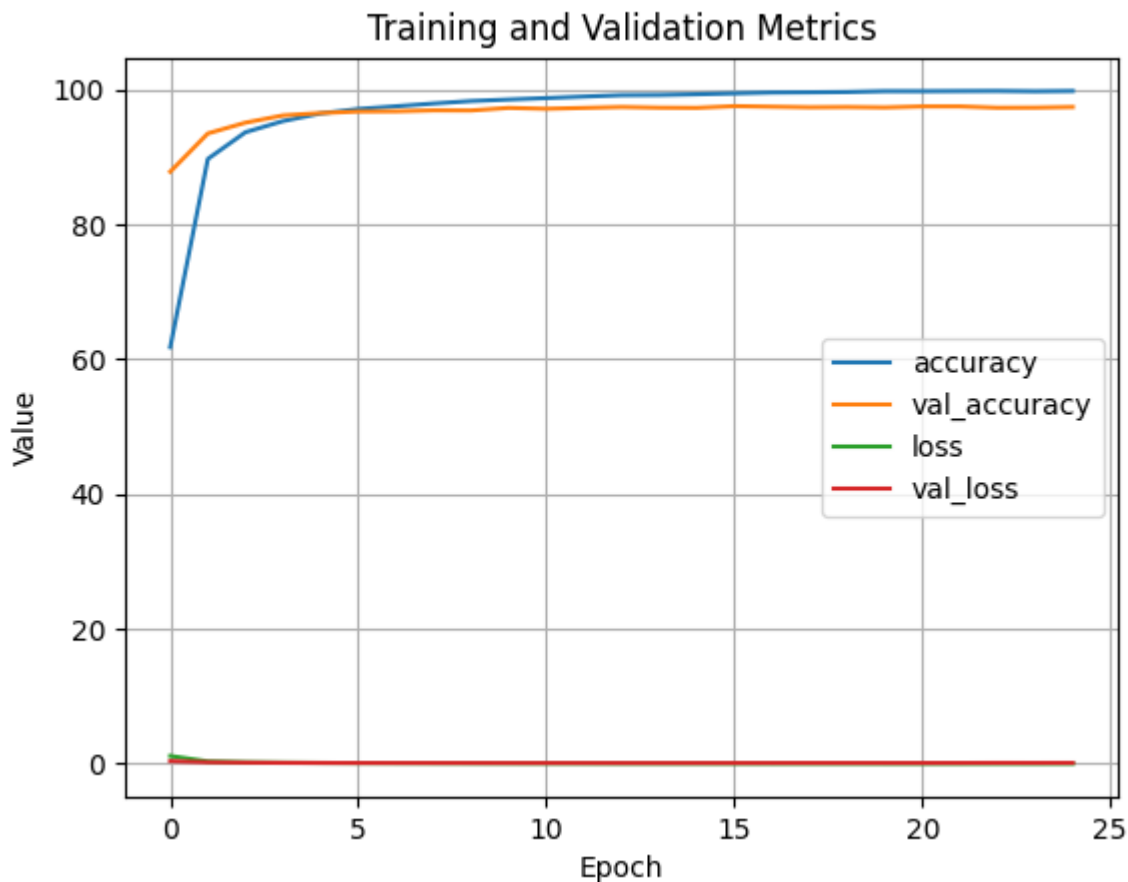
- L'"Epochs" est un hyperparamètre qui consiste en un passage complet du "dataset" d'entraînement à travers le réseau neuronal.
- Le nombre d'Epochs permet de déterminer combien de fois les données d'entraînement sont utilisées (Full Pass).
- On peut trouver le nombre d'itérations que contient une Epoch en divisant le nombre total de données d'entraînement par le nombre de données dans une itération (batch).
- Typiquement, on doit configurer le nombre d'Epochs pour qu'il "challenge" le réseau jusqu'à l'épuisement des gains :
  1. La précision n'augmente plus OU
  2. Elle augmente trop peu pour justifier le coût des ressources technologiques.
- Sur l'image suivante, on peut voir l'effet progressif des "Epoch" sur la précision de notre modèle.

```
Epoch 1/25
12/12  1s 38ms/step - accuracy: 0.4819 - loss: 1.4439 - val_accuracy: 0.9869 - val_loss: 0.3054
Epoch 2/25
12/12  0s 34ms/step - accuracy: 0.8994 - loss: 0.3330 - val_accuracy: 0.9394 - val_loss: 0.1967
Epoch 3/25
12/12  1s 37ms/step - accuracy: 0.9379 - loss: 0.2041 - val_accuracy: 0.9545 - val_loss: 0.1517
Epoch 4/25
12/12  0s 34ms/step - accuracy: 0.9569 - loss: 0.1467 - val_accuracy: 0.9621 - val_loss: 0.1282
Epoch 5/25
12/12  1s 31ms/step - accuracy: 0.9652 - loss: 0.1162 - val_accuracy: 0.9647 - val_loss: 0.1176
Epoch 6/25
12/12  1s 34ms/step - accuracy: 0.9710 - loss: 0.0972 - val_accuracy: 0.9669 - val_loss: 0.1109
Epoch 7/25
12/12  1s 35ms/step - accuracy: 0.9769 - loss: 0.0803 - val_accuracy: 0.9705 - val_loss: 0.1020
Epoch 8/25
12/12  0s 29ms/step - accuracy: 0.9817 - loss: 0.0627 - val_accuracy: 0.9706 - val_loss: 0.1003
Epoch 9/25
12/12  0s 35ms/step - accuracy: 0.9842 - loss: 0.0546 - val_accuracy: 0.9726 - val_loss: 0.0954
Epoch 10/25
12/12  0s 35ms/step - accuracy: 0.9873 - loss: 0.0429 - val_accuracy: 0.9722 - val_loss: 0.0963
Epoch 11/25
12/12  1s 35ms/step - accuracy: 0.9893 - loss: 0.0369 - val_accuracy: 0.9733 - val_loss: 0.0936
Epoch 12/25
12/12  1s 34ms/step - accuracy: 0.9912 - loss: 0.0297 - val_accuracy: 0.9755 - val_loss: 0.0927
Epoch 13/25
12/12  1s 32ms/step - accuracy: 0.9932 - loss: 0.0248 - val_accuracy: 0.9760 - val_loss: 0.0943
Epoch 14/25
12/12  1s 30ms/step - accuracy: 0.9929 - loss: 0.0242 - val_accuracy: 0.9751 - val_loss: 0.0987
Epoch 15/25
12/12  0s 32ms/step - accuracy: 0.9948 - loss: 0.0185 - val_accuracy: 0.9769 - val_loss: 0.0992
Epoch 16/25
12/12  1s 33ms/step - accuracy: 0.9949 - loss: 0.0162 - val_accuracy: 0.9768 - val_loss: 0.0998
Epoch 17/25
12/12  0s 36ms/step - accuracy: 0.9967 - loss: 0.0135 - val_accuracy: 0.9744 - val_loss: 0.1090
Epoch 18/25
12/12  1s 48ms/step - accuracy: 0.9961 - loss: 0.0139 - val_accuracy: 0.9715 - val_loss: 0.1187
Epoch 19/25
12/12  0s 32ms/step - accuracy: 0.9949 - loss: 0.0160 - val_accuracy: 0.9760 - val_loss: 0.1053
Epoch 20/25
12/12  0s 38ms/step - accuracy: 0.9967 - loss: 0.0112 - val_accuracy: 0.9764 - val_loss: 0.1070
Epoch 21/25
12/12  1s 28ms/step - accuracy: 0.9970 - loss: 0.0102 - val_accuracy: 0.9745 - val_loss: 0.1128
Epoch 22/25
12/12  1s 30ms/step - accuracy: 0.9971 - loss: 0.0095 - val_accuracy: 0.9769 - val_loss: 0.1154
Epoch 23/25
12/12  0s 29ms/step - accuracy: 0.9979 - loss: 0.0075 - val_accuracy: 0.9770 - val_loss: 0.1132
Epoch 24/25
12/12  1s 29ms/step - accuracy: 0.9982 - loss: 0.0063 - val_accuracy: 0.9765 - val_loss: 0.1160
Epoch 25/25
12/12  1s 30ms/step - accuracy: 0.9983 - loss: 0.0059 - val_accuracy: 0.9770 - val_loss: 0.1209

Évaluer le nouveau modèle sur l'ensemble de test :
3/3  0s 11ms/step - accuracy: 0.9771 - loss: 0.1044
```

# Graphiques Confondue

- À partir de 12 "Epoch", la valeur "accuracy" n'augmente presque plus et l'entraînement s'en trouve ralenti.
- Le temps de calcul en "GPU Time" nécessaire devient presque exponentiel, ce qui réduit les gains de façon dramatique."
- On peut voir une representation visuel sur le graphique suivant :



1. Dans le graphique ci-dessus, les valeurs "val\_accuracy" et "val\_loss" indiquent les valeurs de validation qui sont utilisées pour valider que nos valeurs "accuracy" et "loss" sont typiques.
2. Les données "Loss" représentent l'habileté du modèle (à travers le temps) à minimiser la différence entre la prédiction et le résultat.
3. Les données "Accuracy" représentent l'habileté du modèle à reconnaître et classifier les variables, c'est le ratio direct de réussite globale.



---

## "Validation Split"

- Le "dataset" est souvent séparé en 3 parties : (training set, validation set, & test set.)
- On peut utiliser le "validation set" pour évaluer la performance.
- Cela permet de réduire les biais de résultat car les résultats ne dépendent pas des résultats environnants.
- Le "Validation Split" permet de prévenir les problèmes de "overfitting".
  - "Overfitting" est le concept dans lequel l'IA n'est pas en mesure de bien généraliser l'information.

---

## "Learning Rate"

- Le "Learning Rate" est un hyperparamètre qui détermine le pas (gain) effectué à chaque itération pendant l'apprentissage.
- Comme nous l'avons appris en classe, il existe un compromis (trade-off) entre la possibilité de "surapprentissage" (overfitting) et de "sous-apprentissage" (underfitting).
  - Lorsque le taux d'apprentissage est trop élevé, il y a une plus grande possibilité de "surapprentissage" (overfitting).
  - Lorsque le taux d'apprentissage est trop bas, la convergence est lente et prend beaucoup de temps.
- Il est recommandé d'ajuster le taux d'apprentissage progressivement pour observer les impacts sur le "training".

---

## "Effet Miroir"

- L'effet miroir, c'est la tendance d'un réseau neuronal à avoir un penchant (biais) pour la symétrie.
  - On brise l'effet miroir en introduisant des irrégularités, ce qui nous permet de différencier et de tester l'information avec un biais intentionnel.
- Briser "l'effet miroir" permet de renforcer l'apprentissage du modèle dans un grand pourcentage des cas.

---

## "Layers"

- Les couches cachées sont des couches qui ne sont pas visibles par l'utilisateur, elles sont situées après la couche de flattening.
  - Il peut y avoir plusieurs couches cachées.
    - Dans le cas de notre modèle en question, il y a 3 couches cachées.
    - Il y a 5 couches au total dans le modèle en question.
  - Sur la dernière couche, il y a 10 neurones (nœuds) qui représentent chacun des 10 chiffres.
-

# Tests Unitaires

1. Le test n°1 effectue une vérification implicite des couches du modèle ; on vérifie si le nombre de couches programmées sont présentes au moment de la compilation.
2. Le test n°2 vérifie si l'entraînement a bien été effectué avec un sous-ensemble de test, un petit échantillon est prélevé et testé pour double-vérifier que les données d'entraînement sont bien valides.
3. Le test n°3 vérifie une fois de plus l'entraînement sur un petit échantillon, cette fois-ci avec un test de précision.

```
Testing started at 10:03 p.m. ...
```

```
Launching unittests with arguments python -m unittest
```

```
C:\Users\alex\PycharmProjects\OTTONIEL\test_unitaire.py in
```

```
C:\Users\alex\PycharmProjects\OTTONIEL
```

```
2024-08-27 22:03:54.089555: I tensorflow/core/util/port.cc:153] oneDNN
custom operations are on. You may see slightly different numerical results
due to floating-point round-off errors from different computation orders. To
turn them off, set the environment variable TF_ENABLE_ONEDNN_OPTS=0.
```

```
2024-08-27 22:03:55.010221: I tensorflow/core/util/port.cc:153] oneDNN
custom operations are on. You may see slightly different numerical results
due to floating-point round-off errors from different computation orders. To
turn them off, set the environment variable TF_ENABLE_ONEDNN_OPTS=0.
```

```
2024-08-27 22:03:56.941425: I
tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is
optimized to use available CPU instructions in performance-critical
operations.
```

```
To enable the following instructions: AVX2 FMA, in other operations, rebuild
TensorFlow with the appropriate compiler flags.
```

```
32/32 _____ 0s 1ms/step - accuracy: 0.9863 - loss: 0.0566
```

```
32/32 _____ 0s 1ms/step - accuracy: 0.8674 - loss: 0.8956
```

```
Ran 3 tests in 0.940s
```

```
OK
```

```
Process finished with exit code 0
```

- Le github est public : [https://github.com/OttonielCA/AI\\_TP1](https://github.com/OttonielCA/AI_TP1)
- Merci d'avoir lue notre travail!

Ottoniel et Sean