Deep Learning in Computer Vision

Lab 4 – Report

1. Convolutional Neural Network on MNIST dataset

On essaie différentes architectures sur l'ensemble d'images MNIST et on enregistre les résultats après 10 epochs.

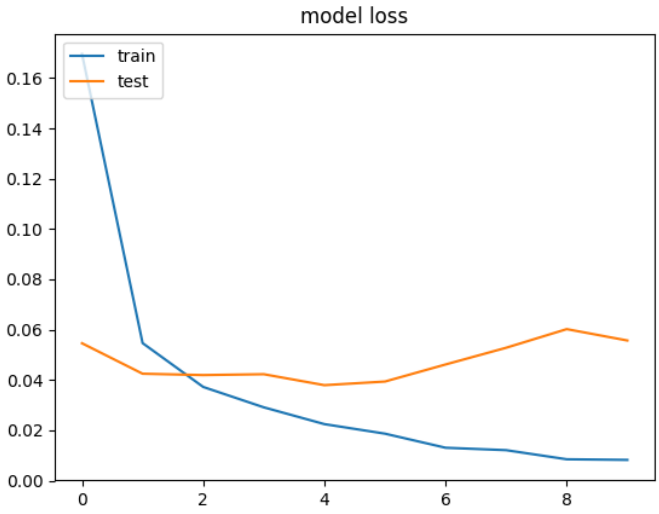
Modèle 1

Optimizer utilisé: adam

Input → Conv2D (64 filtres, relu) → Conv2D (32 filtres, relu) → MaxPooling → Flatten → Dense(softmax)

Accuracy finale sur l'ensemble de test : **0.9866**

On remarque un **overfitting conséquent** après quelques epochs seulement.



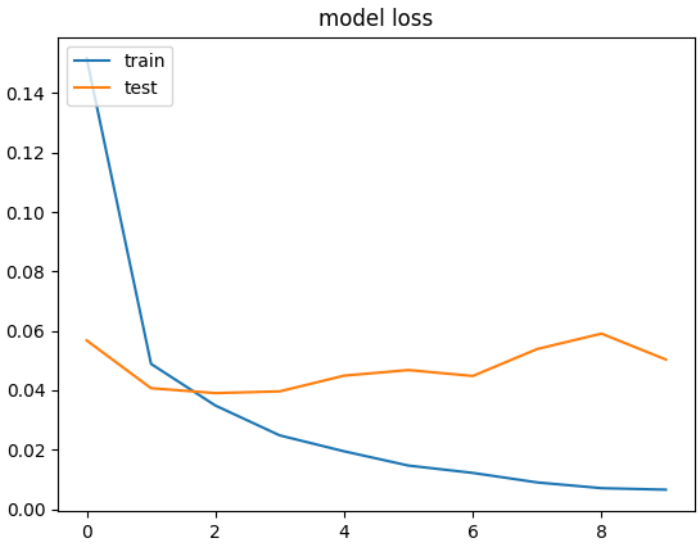
Modèle 2

Optimizer utilisé: adam

Input → Conv2D (64 filtres, relu) → Conv2D (**64 filtres**, relu) → MaxPooling → Flatten → Dense(softmax)

Accuracy finale sur l'ensemble de test : **0.9888**

On remarque **une augmentation de la précision** lorsque le nombre de filtres de la deuxième couche est mis à 64.



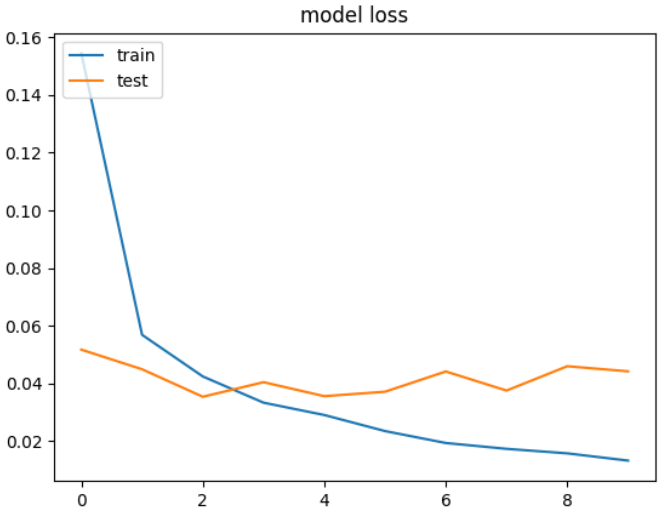
Modèle 3

Optimizer utilisé: adam

Input → Conv2D (**128 filtres**, relu) → Conv2D (64 filtres, relu) → MaxPooling → Flatten → Dense(softmax)

Accuracy finale sur l'ensemble de test : **0.989**

On remarque un **résultat presque équivalent** lorsque l'on augmente encore le nombre de filtres.



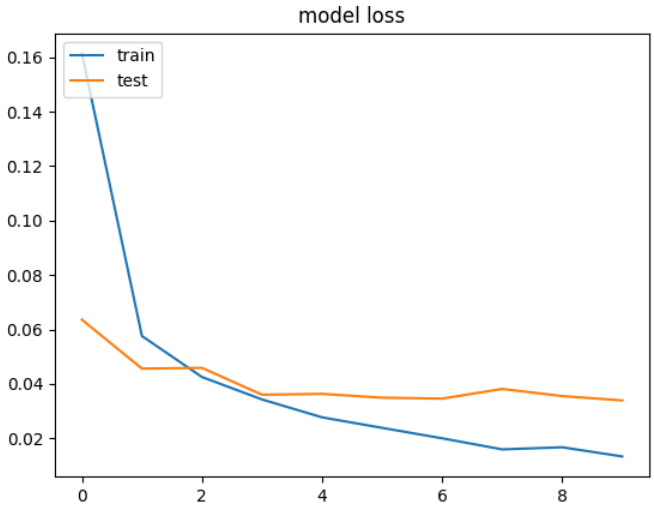
Modèle 4

Optimizer utilisé: adam

Input → Conv2D (64 filtres, relu) → Conv2D (64 filtres, relu) → MaxPooling → **Dropout(0.3)** → Flatten → Dense(softmax)

Accuracy finale sur l'ensemble de test : **0.9911**

On note une **réduction de l'overfitting** avec l'ajout d'un dropout.



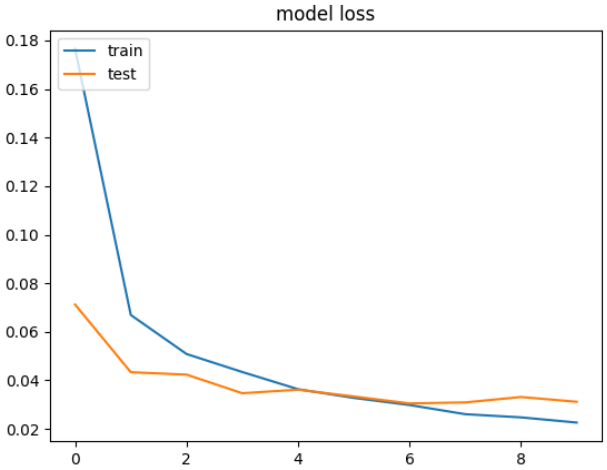
Modèle 5

Optimizer utilisé: adam

Input → Conv2D (64 filtres, relu) → Conv2D (64 filtres, relu) → MaxPooling → Dropout(**0.5**) → Flatten → Dense(softmax)

Accuracy finale sur l'ensemble de test : **0.9906**

On note une **réduction de l'overfitting** mais avec **perte de précision** lorsque l'on augmente encore le dropout.



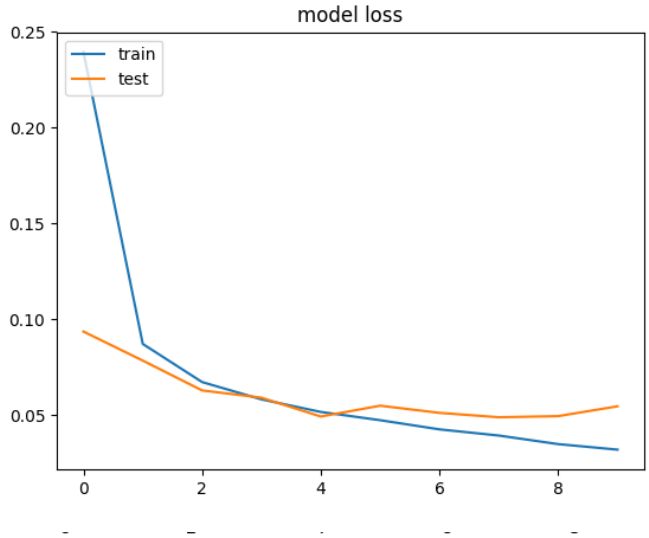
Modèle 6

Optimizer utilisé: adam

Input → **Conv2D (64 filtres, relu)** → **MaxPooling** →Dropout(0.3) → Flatten → Dense(softmax)

Accuracy finale sur l'ensemble de test : **0.9834**

On remarque une forte perte de précision avec la suppression de la deuxième couche Conv2D.



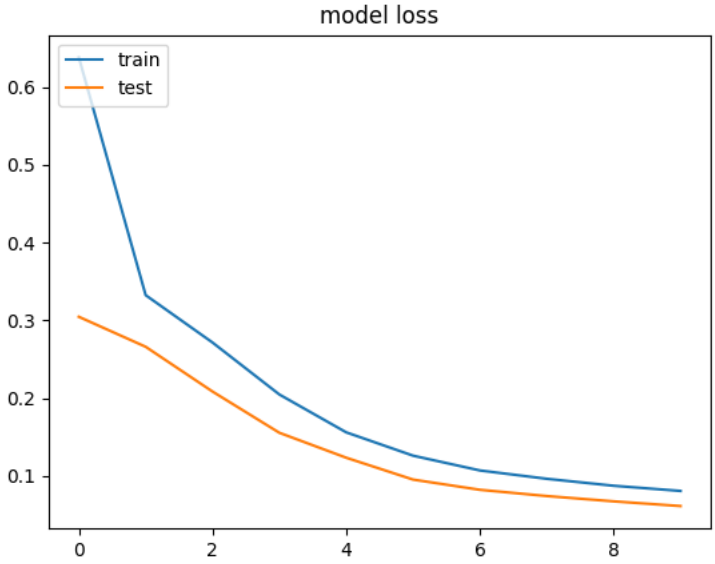
Modèle 7

Optimizer utilisé: **sgd**

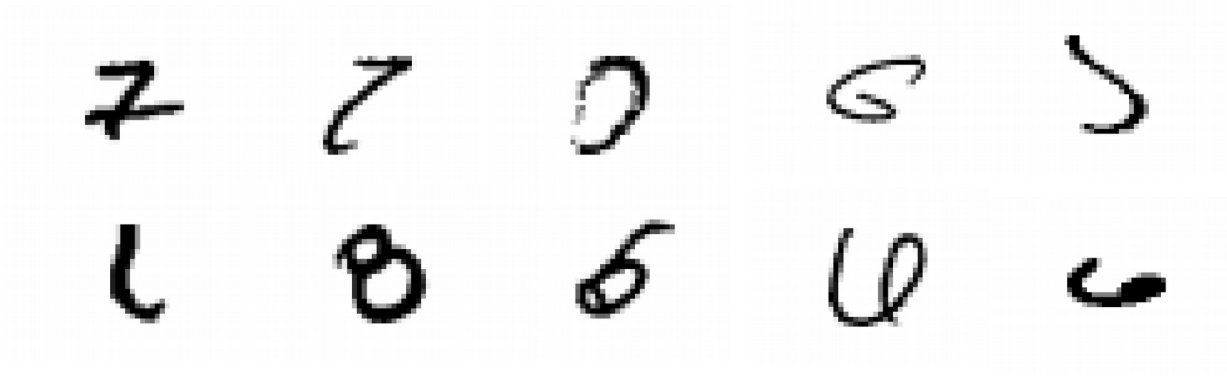
Input → Conv2D (64 filtres, relu) → Conv2D (64 filtres, relu) → MaxPooling →Dropout(0.3) → Flatten → Dense(softmax)

Accuracy finale sur l'ensemble de test : **0.9827**

On remarque une **forte perte de précision** lorsque l'on passe de 'adam' à 'sgd'.



Les dix images les moins bien classées



2. Convolutional Neural Network on CIFAR10 dataset

On utilise l'architecture qui nous a servie pour la partie précédente comme base pour celle-ci. On compare quelques méthodes sur 10 epochs.

Modèle 1

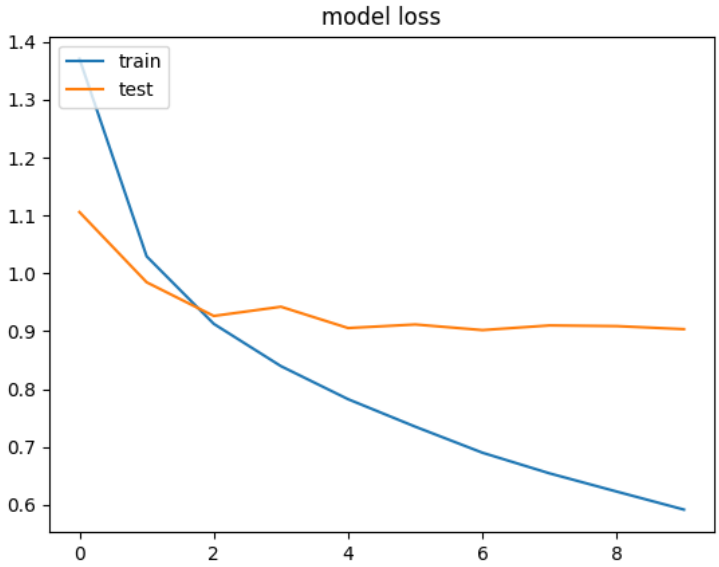
Optimizer utilisé: adam

Input → Conv2D (64 filtres, relu) → Conv2D (64 filtres, relu) → MaxPooling →Dropout(0.3) → Flatten → Dense(softmax)

Accuracy finale sur l'ensemble de test : **0.7047**

On remarque une **forte perte de précision** par rapport aux images précédentes que l'on attribue à la présence des trois canaux de couleurs.

16,26 cm



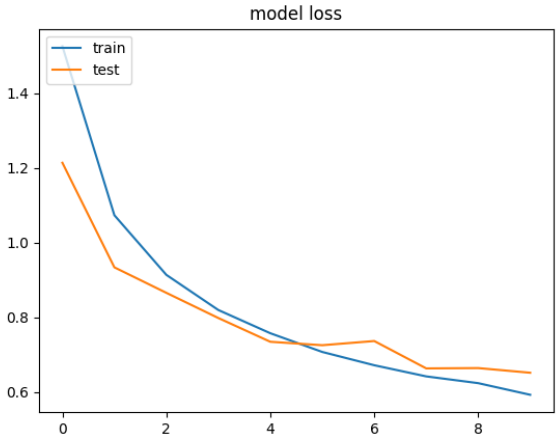
Modèle 2

Optimizer utilisé: adam

Input → Conv2D (64 filtres, relu) → Conv2D (64 filtres, relu) → MaxPooling →Dropout(0.3) → **Conv2D (64 filtres, relu) → Conv2D (64 filtres, relu) → MaxPooling →Dropout(0.3)** → Flatten → Dense(softmax)

Accuracy finale sur l'ensemble de test : **0.7749**

On remarque que la duplication des couches déjà présentes apporte une **augmentation de notre précision finale** et une **réduction de l'overfitting**.



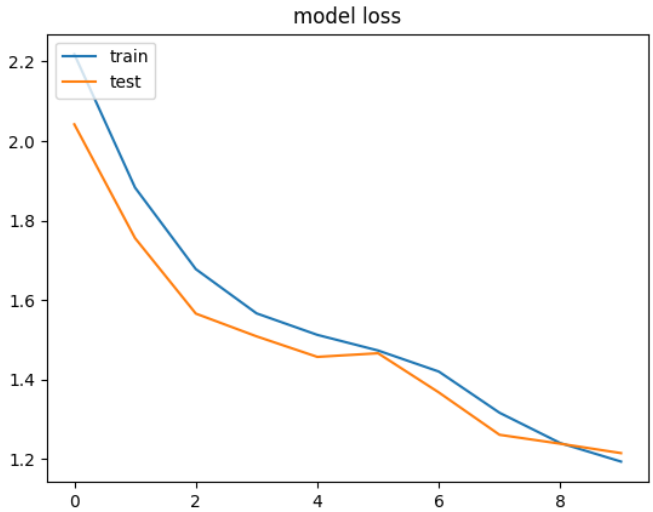
Modèle 3

Optimizer utilisé: adam

Input → Conv2D (64 filtres, **sigmoid**) → Conv2D (64 filtres, **sigmoid**) → MaxPooling →Dropout(0.3) → Flatten → Dense(softmax)

Accuracy finale sur l'ensemble de test : **0.5786**

On remarque une **forte perte de précision** lors de l'utilisation de la fonction d'activation sigmoid.



Les dix images les moins bien classées