Projet Science de données

One Shot Learning

Présenté par :
Ghita Benjelloun
Chen Dang
Joachim Dublineau

Encadré par:
Benjamin Negrevergne

Sommaire

- **Cifar10** - Entraînement sur 10 classes
- **Cifar10** - Entraînement sur 6 classes
- **Mnist** - Entraînement sur 5 classes

❑ **Cifar10** - Sur 10 classes

Structure des données:

images 32*32*3

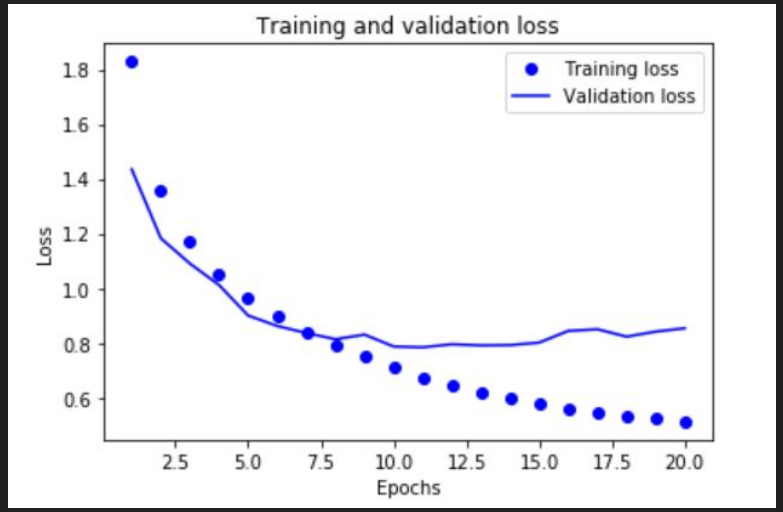
10 Classes:
Airplanes, Auto, Bird, Cat ...

Structure du ConvNet:

Layer (type)	Output Shape	Param #
conv2d_51 (Conv2D)	(None, 32, 32, 100)	2800
activation_71 (Activation)	(None, 32, 32, 100)	0
conv2d_52 (Conv2D)	(None, 32, 32, 100)	90100
activation_72 (Activation)	(None, 32, 32, 100)	0
max_pooling2d_21 (MaxPooling	(None, 16, 16, 100)	0
dropout_41 (Dropout)	(None, 16, 16, 100)	0
conv2d_53 (Conv2D)	(None, 16, 16, 200)	180200
activation_73 (Activation)	(None, 16, 16, 200)	0
conv2d_54 (Conv2D)	(None, 16, 16, 200)	360200
activation_74 (Activation)	(None, 16, 16, 200)	0
conv2d_55 (Conv2D)	(None, 16, 16, 400)	720400
activation_75 (Activation)	(None, 16, 16, 400)	0
max_pooling2d_22 (MaxPooling	(None, 8, 8, 400)	0
dropout_42 (Dropout)	(None, 8, 8, 400)	0
flatten_11 (Flatten)	(None, 25600)	0
dropout_43 (Dropout)	(None, 25600)	0
dense_21 (Dense)	(None, 600)	15360600
activation_76 (Activation)	(None, 600)	0
dropout_44 (Dropout)	(None, 600)	0
dense_22 (Dense)	(None, 6)	3606
activation_77 (Activation)	(None, 6)	0

Total params: 16,717,906

❑ **Cifar10** - Entrainement



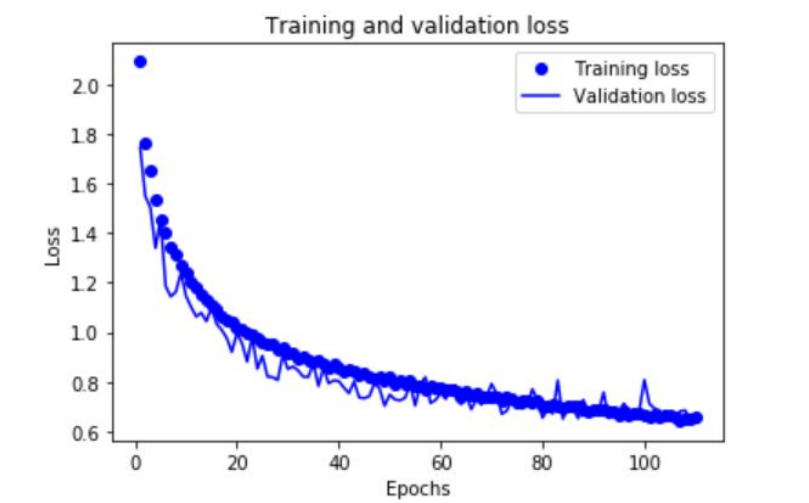
Hyperparamètres:

Learning_rate = 0.0001
Batch_size = 32
weight_penalty = 0.0001
optimizer = Adam
Loss = ‘categorical_crossentropy’

Sur les données du test :

Loss = 0.87
Accuracy = 81%

❑ **Cifar10** - Entrainement - Data Augmentation



Hyperparamètres Data Augmentation:

rotation = 2
width shift = 0.15
height shift = 0.15
shear =0.1
zoom = 0.1

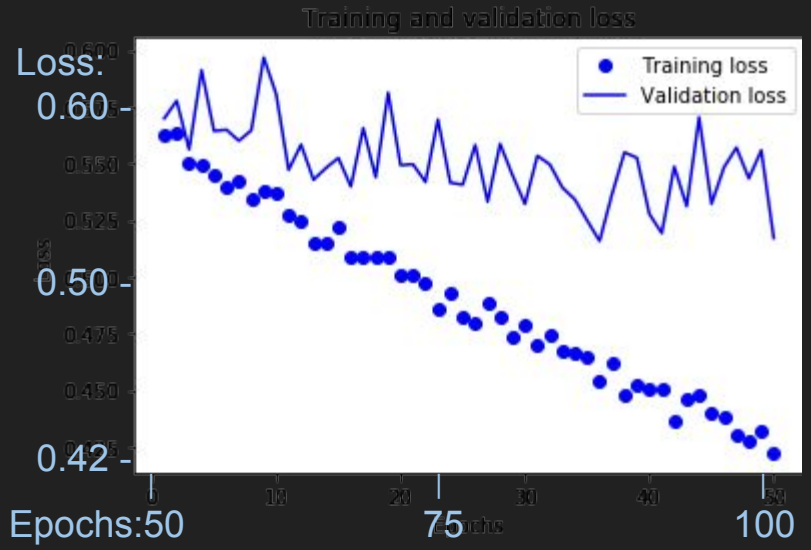
Sur les données du test :

Loss = 0.67
Accuracy = 87%

Utilisation du réseau pour le One-Shot Learning

- Notre réseau Convolutif obtient de bonnes performances sur les 10 classes.
- On va alors l'entraîner sur 6 classes, récupérer l'avant dernière couche qui représentera les informations de l'image.
- Voir si cette représentation permet de bien distinguer les 4 classes suivantes par la méthode du plus proche voisin.

❑ **Cifar10** - Entrainement sur 6 classes



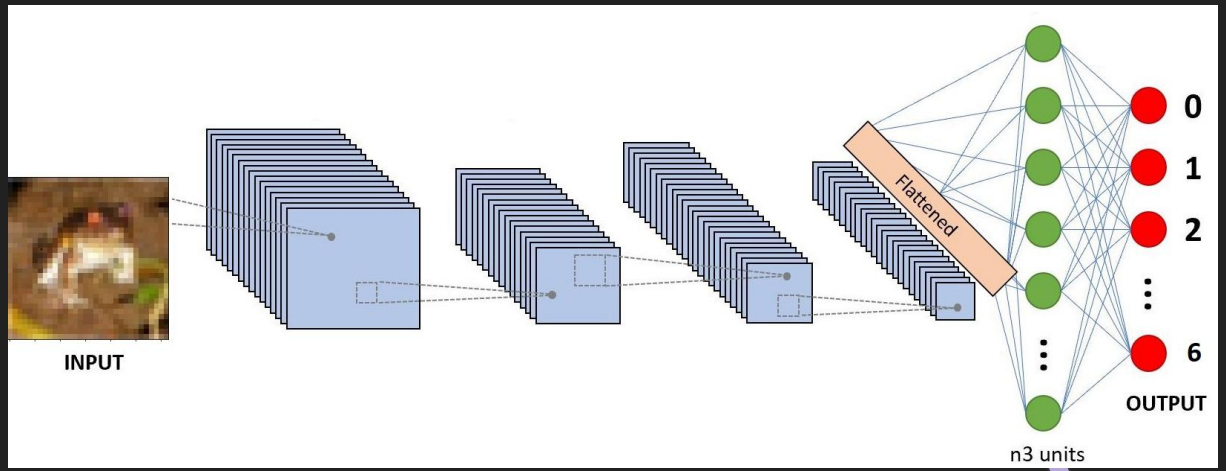
Paramètres:

Learning_rate = 0.0001
Batch_size = 32
weight_penalty = 0.0001
optimizer = Adam
Loss = 'categorical_crossentropy'
+
Data Augmentation

Performance on test: [0.5297104382514953, 0.89]

❑ **Cifar10** - Basic One Shot Learning

On prend 4 représentants: 1 par classes et on récupère leur vecteur de représentation dans notre réseau.

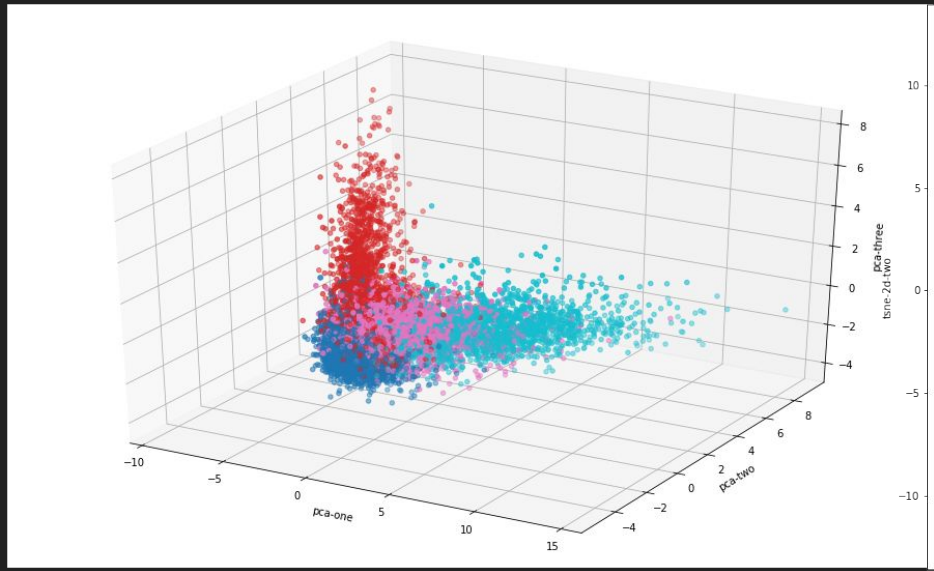


Vecteur de représentation

On attribue à chaque image la classe de son plus proche voisin parmis les représentants dans l'espace de représentation.

❑ **Cifar10** - Basic One Shot Learning - Résultats

Visualisation par réduction de dimension (PCA, t-SNE)



PCA

t-SNE

La représentation apprise par le ConvNet est assez pertinente

❑ **Cifar10** - Basic One Shot Learning - Résultats

Class n° 6 accuracy: 0.6269527896995707
Class n° 7 accuracy: 0.5944921316165951
Class n° 8 accuracy: 0.5984692417739628
Class n° 9 accuracy: 0.5969241773962805

Matrice de Confusion (prédiction, référence):

[[47843 24311 7719 4457]
 [13920 37164 6112 3810]
 [6050 6021 38962 23359]
 [2087 2404 17107 38274]]

	Rappel	Précision
Class 6	0.68	0.57
Class 7	0.53	0.61
Class 8	0.56	0.52
Class 9	0.55	0.64

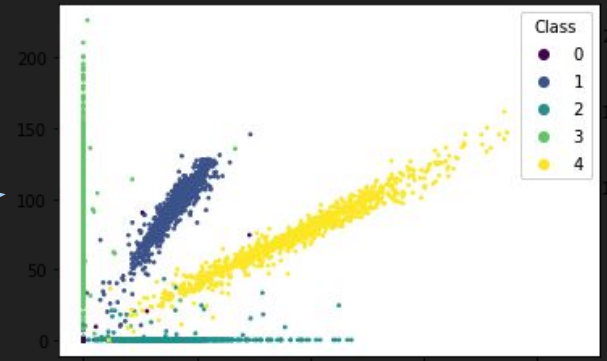
- Beaucoup de confusions entre 6 et 7 (grenouilles et chevaux)
- Beaucoup de confusions entre 9 et 10 (bateaux et camions)

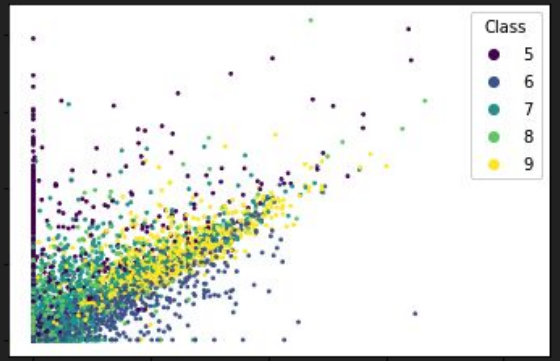
❑ **MNIST** - Embeddings avec un réseau dense

NN

Layer (type)	Output Shape
reshape_2 (Reshape)	(None, 784)
dense_5 (Dense)	(None, 128)
dense_6 (Dense)	(None, 128)
dense_7 (Dense)	(None, 2)
dense_8 (Dense)	(None, 10)

Total params: 117,280
Trainable params: 117,280
Non-trainable params: 0

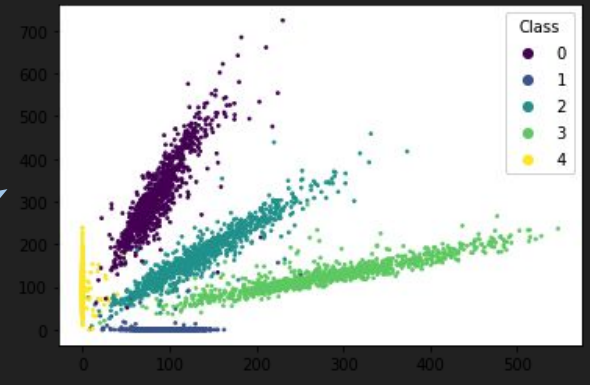


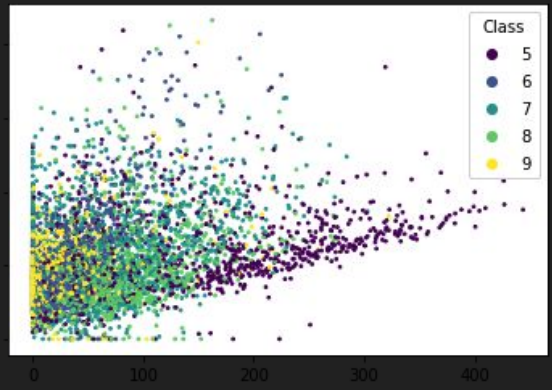


CNN

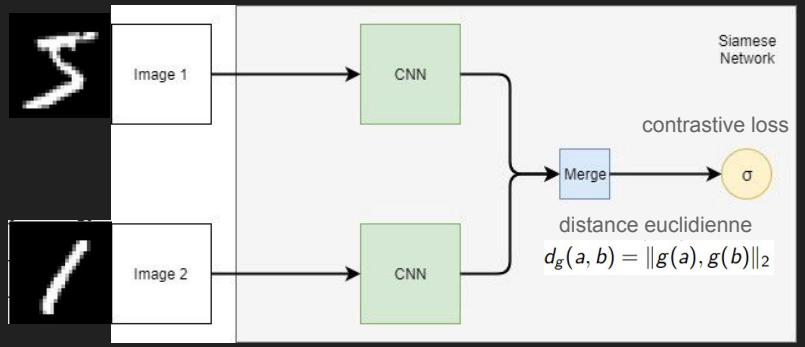
Layer (type)	Output Shape
conv2d_1 (Conv2D)	(None, 28, 28, 64)
max_pooling2d_1 (MaxPooling2	(None, 14, 14, 64)
dropout_1 (Dropout)	(None, 14, 14, 64)
conv2d_2 (Conv2D)	(None, 14, 14, 64)
max_pooling2d_2 (MaxPooling2	(None, 7, 7, 64)
dropout_2 (Dropout)	(None, 7, 7, 64)
flatten_1 (Flatten)	(None, 3136)
dense_1 (Dense)	(None, 2)
dense_2 (Dense)	(None, 10)

Total params: 43,872
Trainable params: 43,872
Non-trainable params: 0





❑ **MNIST** - Embeddings avec un réseau siamois



$$L(a,b,g) = y_{ab} \cdot \frac{1}{2} \cdot d_g(a,b)^2 \quad + \quad (1-y_{ab}) \cdot \frac{1}{2} \cdot \max(0, m - d_g(a,b))^2$$

Layer (type)	Output Shape
input1 (InputLayer)	(None, 28, 28, 1)
input2 (InputLayer)	(None, 28, 28, 1)
sequential_2 (Sequential)	(None, 2)
lambda_2 (Lambda)	(None, 1)

Total params: 39,426
Trainable params: 39,426
Non-trainable params: 0

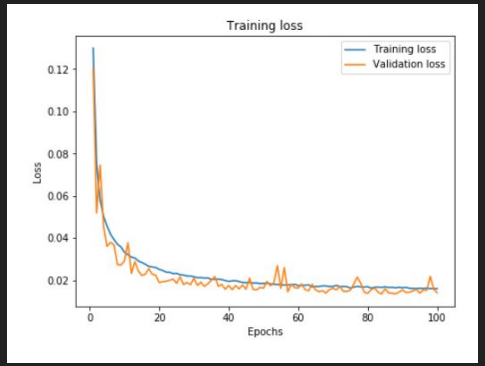
Layer (type)	Output Shape
conv2d_3 (Conv2D)	(None, 28, 28, 32)
max_pooling2d_3 (MaxPooling2	(None, 14, 14, 32)
dropout_3 (Dropout)	(None, 14, 14, 32)
conv2d_4 (Conv2D)	(None, 14, 14, 64)
max_pooling2d_4 (MaxPooling2	(None, 7, 7, 64)
dropout_4 (Dropout)	(None, 7, 7, 64)
flatten_2 (Flatten)	(None, 3136)
dense_2 (Dense)	(None, 2)

Total params: 39,426
Trainable params: 39,426
Non-trainable params: 0

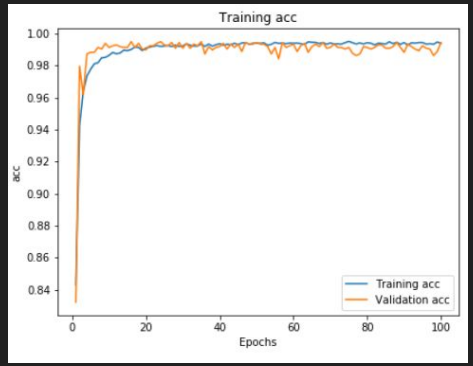
❑ **MNIST** - Embeddings avec un réseau siamois

Entraînement sur 5 classes

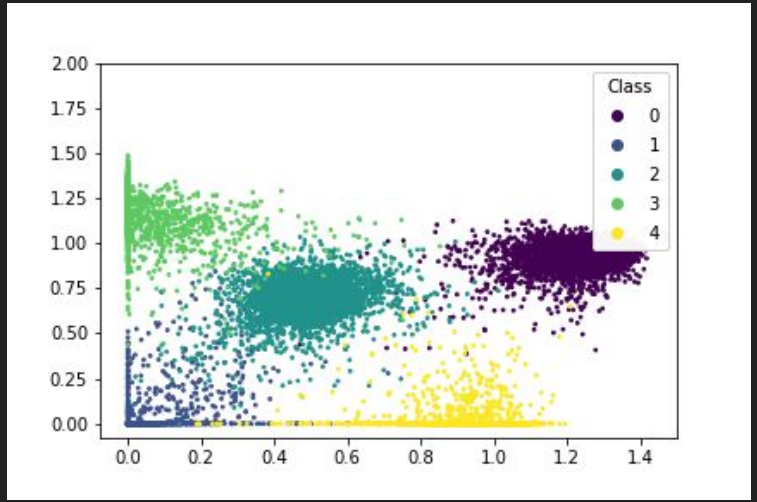
Training and validation loss



Training and validation accuracy



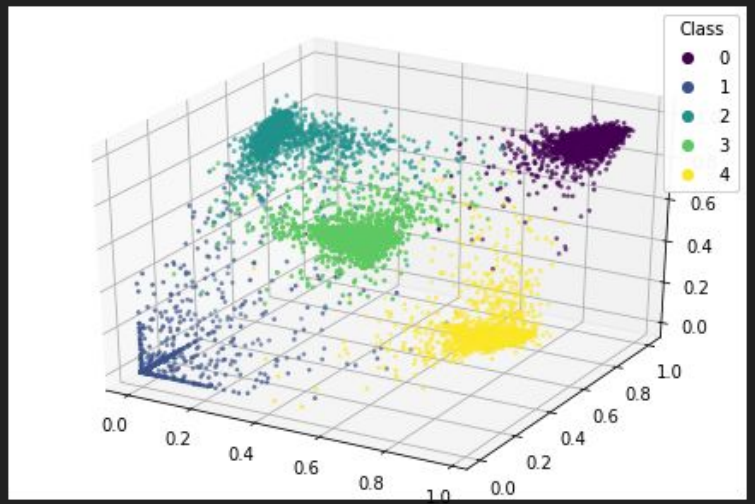
Embeddings 2D de 0 à 4



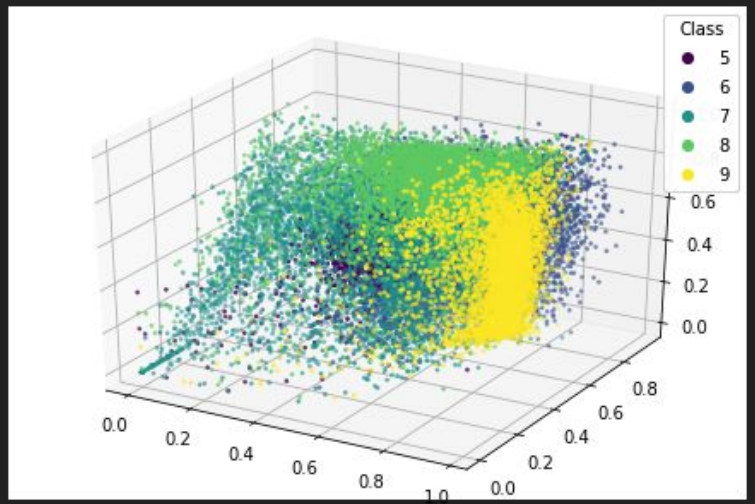
❑ **MNIST** - Embeddings avec un réseau siamois

Visualisation 3D

Embeddings de 0 à 4



Embeddings de 5 à 9



- confusions entre 4 et 9
- confusions entre 1 et 7

Loss et accuracy sur le test set

```
10842/10842 [==============================] - 2s 167us/step
[0.16905730505355543, 0.7595462092194823]
```

Conclusions