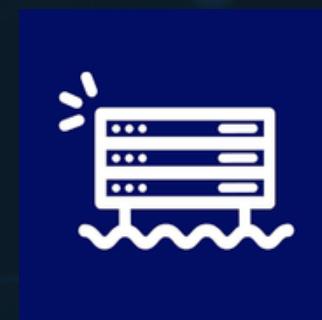


Neural Zoo

# DEEP LEARNING PROJECT



Speaker

Pierre-Alexis Lebair



# Sommaire

## Introduction

Explication du sujet et mise en contexte

## Veille MLP

Résumé de la veille sur le MLP

## Veille CNN

Résumé de la veille sur le CNN

01

02

03

04

05

06

## Pré-traitement des données

Détails sur le traitement des données et visualisation

## Implémentation des modèles

Explication des choix de modèles et de leur paramétrage

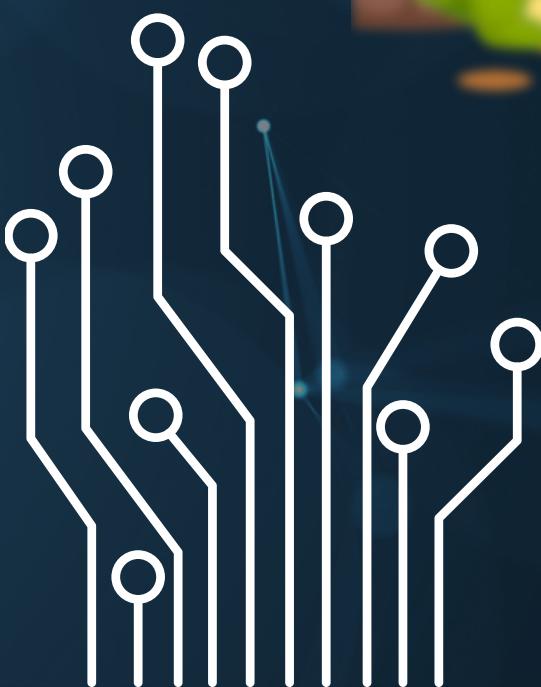
## Conclusion

Conclusion sur la question posé dans le sujet et suggestion sur la marche à suivre

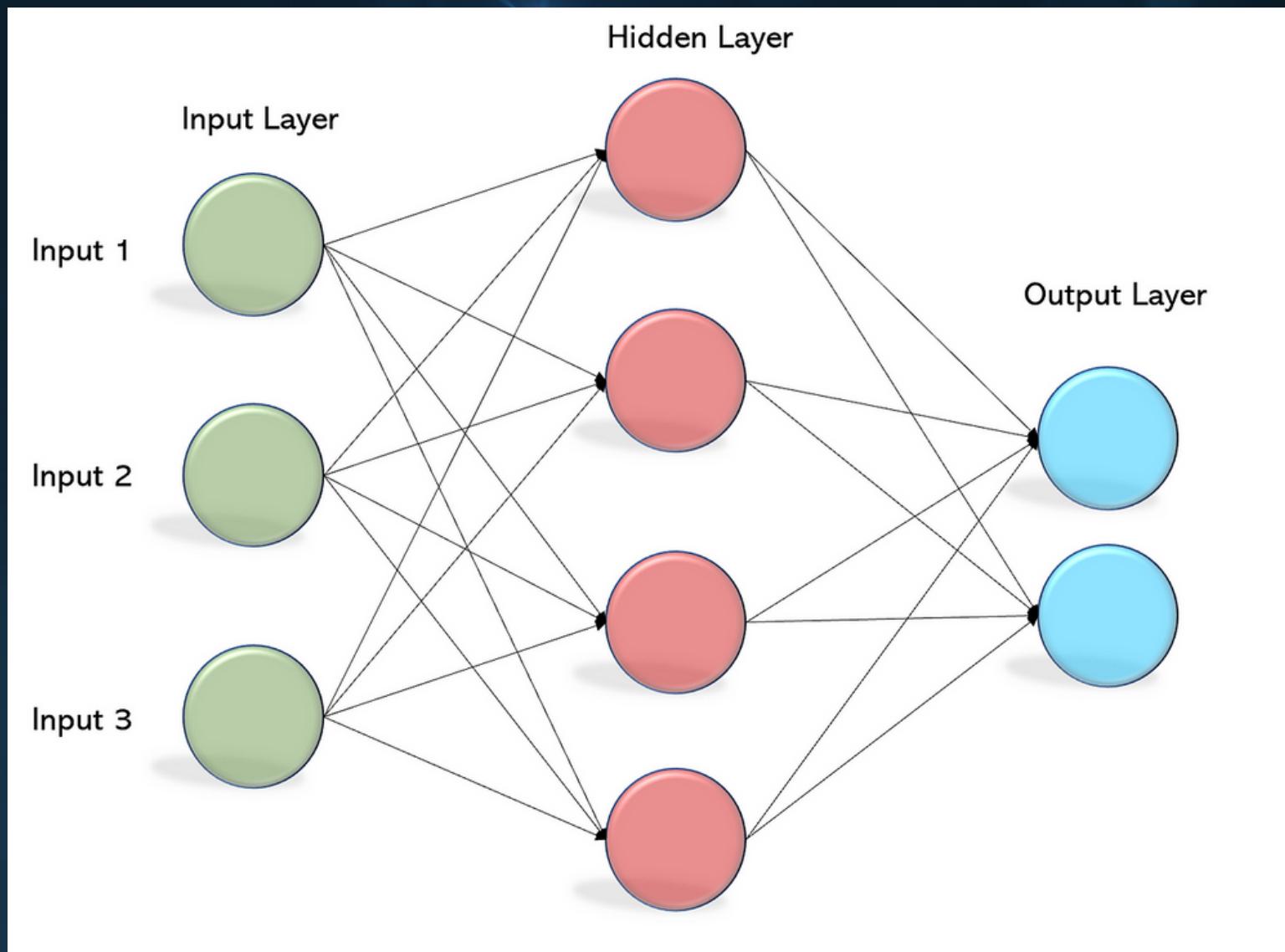
# Introduction

- Deep Learning
- Classification d'images
- Réseau de neurones

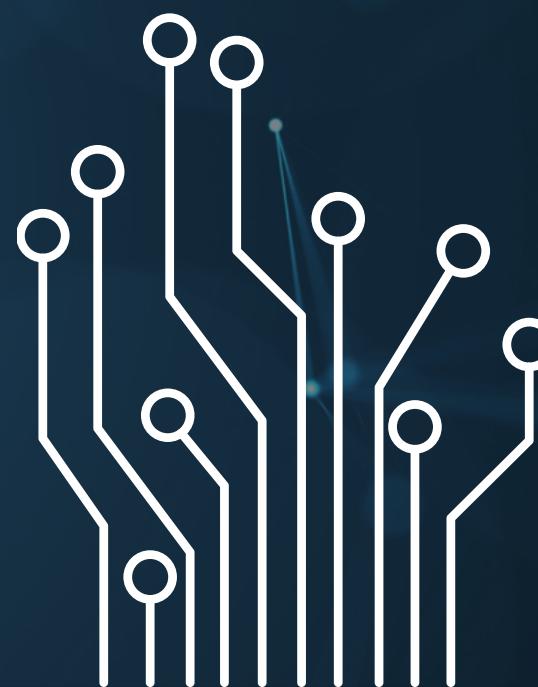
Le Deep Learning est une catégorie d'intelligence artificielle qui exploite des réseaux de neurones artificiels, inspirés directement du fonctionnement des neurones humains, pour assimiler de nouvelles connaissances. Ces réseaux sont constitués de plusieurs couches de neurones artificiels connectés entre eux.



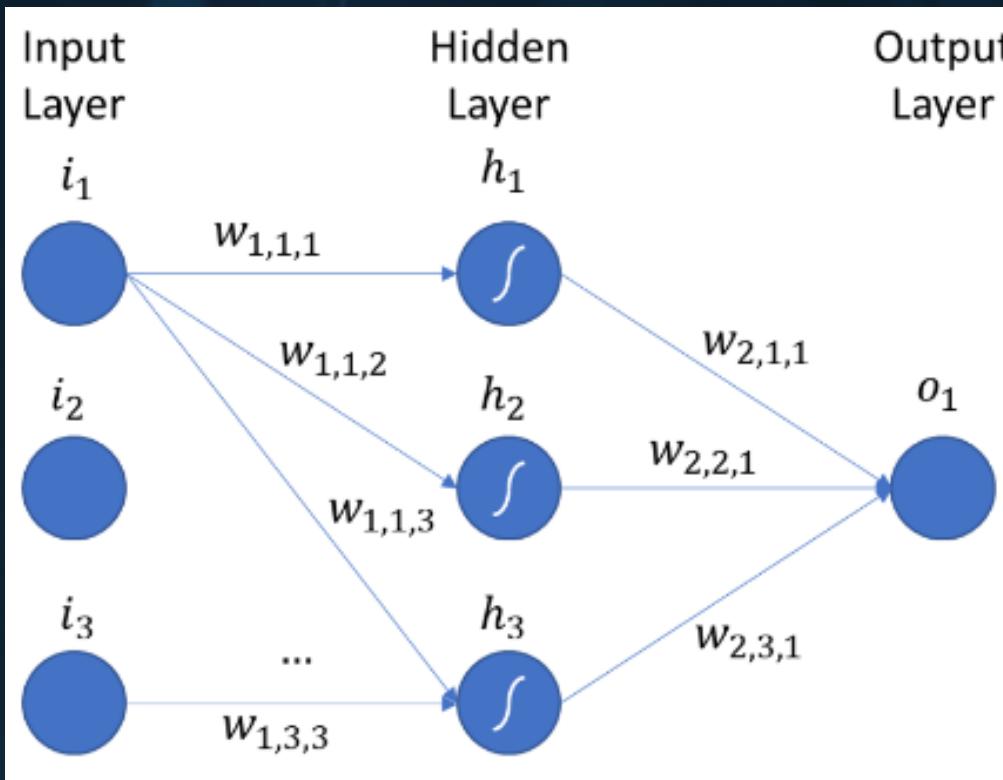
# Veille sur le MLP



le perceptron multicouche (multilayer perceptron MLP en anglais) est un type de réseau neuronal artificiel organisé en plusieurs couches. Un perceptron multicouche possède au moins trois couches : une couche d'entrée, au moins une couche cachée, et une couche de sortie

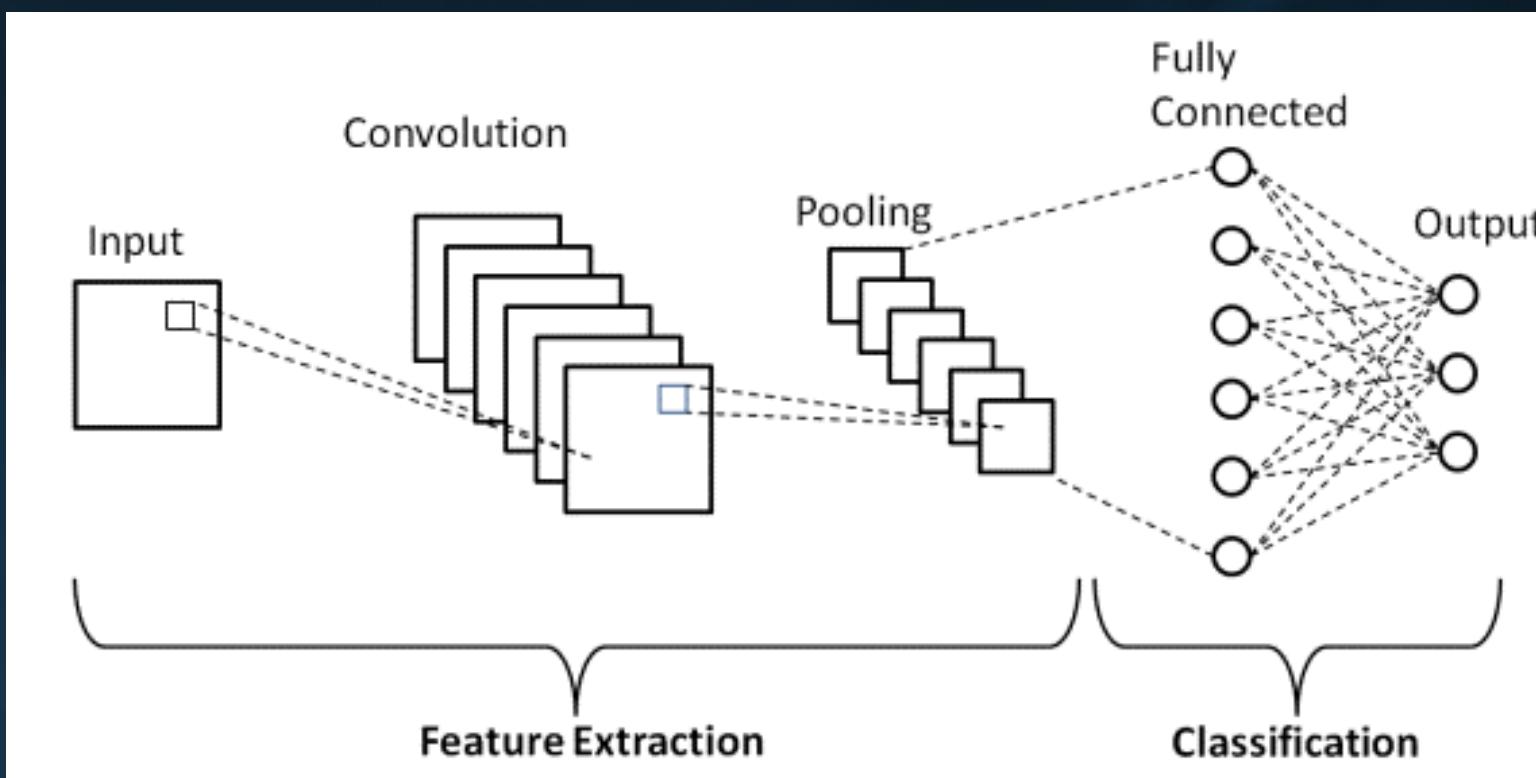


# Veille sur le MLP

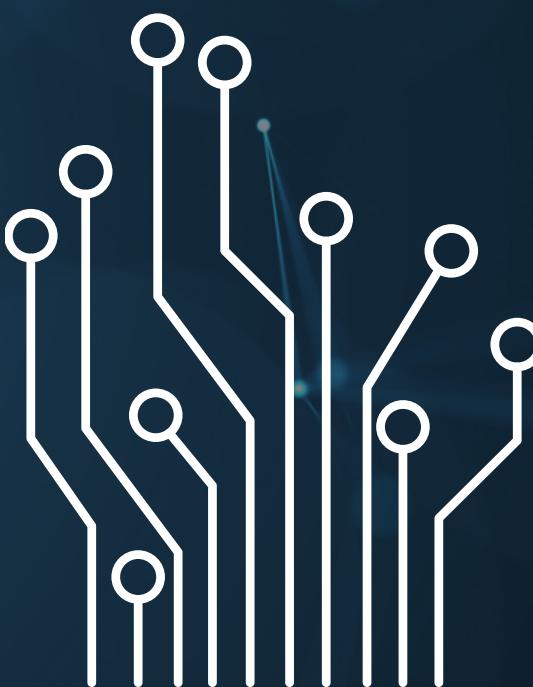


- Nombre de couches cachées
- Nombre de neurones par couche cachée
- Fonction d'activation
- Taux d'apprentissage
- Nombre d'itérations (epochs)
- Architecture
- Fonction de perte

# Veille sur le CNN



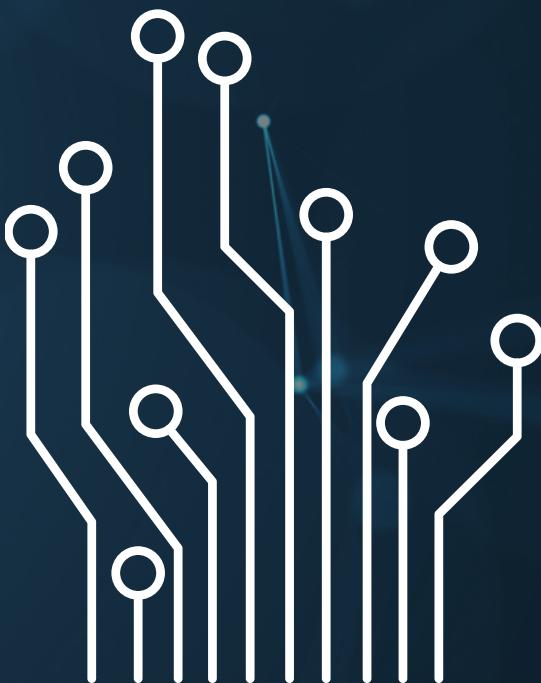
Les réseaux de neurones artificiels de type convolutifs (Convolutional Neural Networks ou CNNs) sont une classe de modèles d'apprentissage profond couramment utilisés dans le domaine de la vision par ordinateur. Ils ont révolutionné le traitement des images et sont largement utilisés dans des tâches telles que la classification d'images, la détection d'objets et la segmentation sémantique.



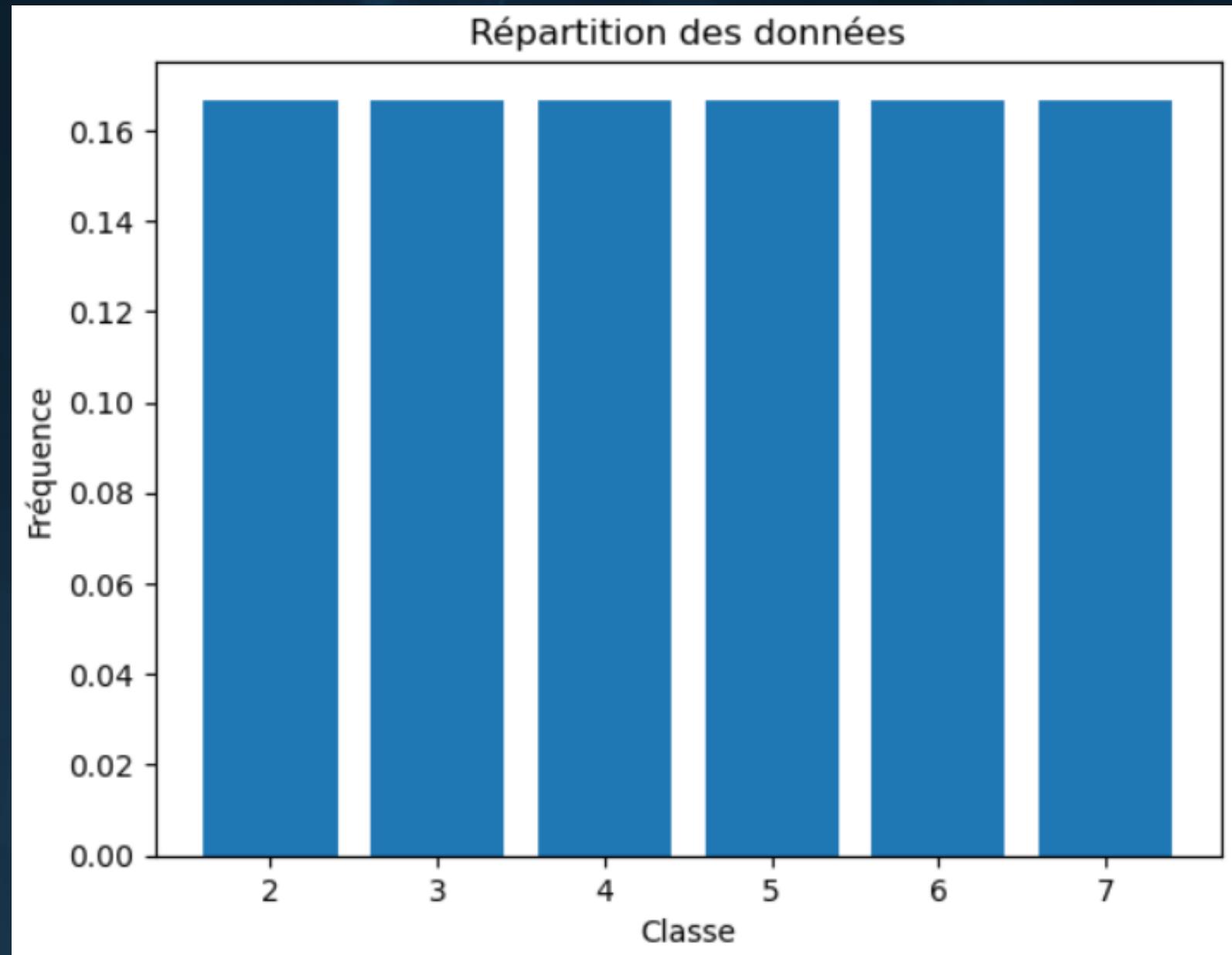
# Veille sur le CNN



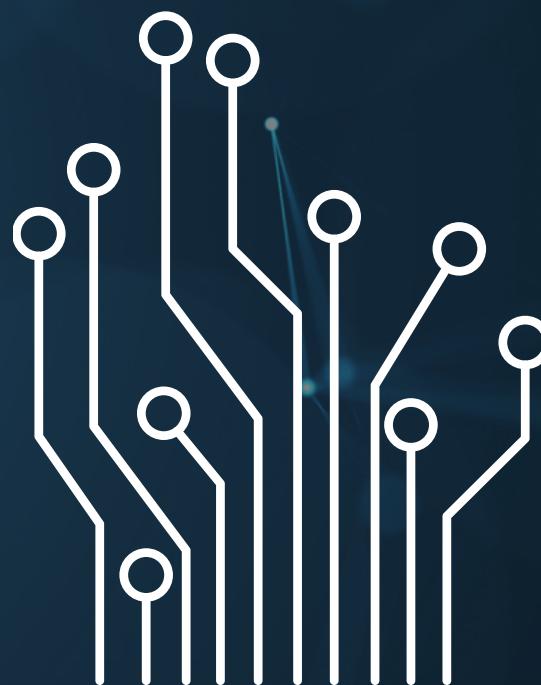
- Couches de convolution
- Couches de pooling
- Taille du noyau de convolution et nombre de filtres
- Taille et pas du pooling
- Couches de dropout



# Analyse des données



- Répartition équilibrée des données
- 50000 images de 32\*32 en (RGB)
- Quantité de données suffisante



```
Entrée [38]: x_train_full.shape  
Out[38]: (50000, 32, 32, 3)  
  
Entrée [39]: y_train_full.shape  
Out[39]: (50000, 1)
```

# Pré-traitement des données



## NORMALISATION

Valeur des pxls entre 0 et 255

```
# Prétraitement des données
x_train = x_train.astype("float32") / 255.0
x_val = x_val.astype("float32") / 255.0
x_test_filtered = x_test_filtered.astype("float32") / 255.0
```

# Pré-traitement des données

```
y_train_flat = y_train.reshape(-1, 1)
y_val_flat = y_val.reshape(-1, 1)
y_test_flat = y_test_filtered.reshape(-1, 1)

enc = OneHotEncoder(categories=[animal_classes], sparse=False)
enc.fit(y_train_flat)

y_train = enc.transform(y_train_flat)
y_val = enc.transform(y_val_flat)
y_test_filtered = enc.transform(y_test_flat)

y_train.shape
(24000, 6)

x_train.shape
(24000, 32, 32, 3)
```



## ONEHOTENCODER

Encoder les labels sous forme catégorielle

# Pré-traitement des données

```
: y_train_filtered  
  
array([[6],  
       [4],  
       [2],  
       ...,  
       [2],  
       [2],  
       [6]], dtype=uint8)
```



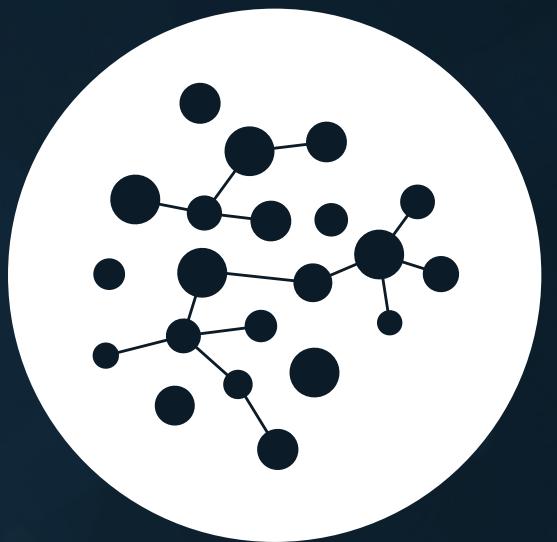
```
array([[0., 1., 0., 0., 0., 0.],  
       [0., 0., 0., 1., 0., 0.],  
       [0., 0., 0., 1., 0., 0.],  
       ...,  
       [0., 0., 1., 0., 0., 0.],  
       [0., 1., 0., 0., 0., 0.],  
       [0., 0., 0., 0., 1., 0.]])
```



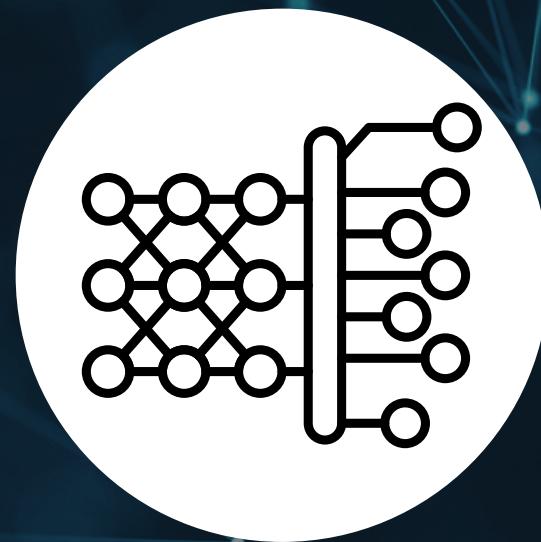
## ONEHOTENCODER

Encoder les labels sous forme catégorielle

# Implémentation des modèles



MLP

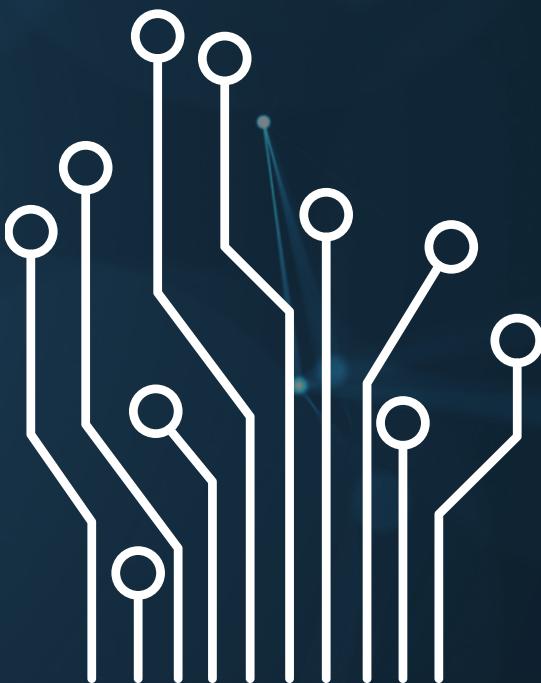


CNN

# MLP

```
# Construction du modèle MLP
mlp_model = keras.Sequential([
    keras.layers.Flatten(input_shape=(32, 32, 3)),
    keras.layers.Dense(128, activation="relu"),
    keras.layers.BatchNormalization(),
    keras.layers.Dense(64, activation="relu"),
    keras.layers.BatchNormalization(),
    keras.layers.Dense(6, activation="softmax")
])

# Compilation et entraînement du modèle MLP
mlp_model.compile(optimizer="adam", loss="categorical_crossentropy", metrics=["accuracy"])
mlp_history = mlp_model.fit(x_train, y_train, epochs=20, validation_data=(x_val, y_val))
```



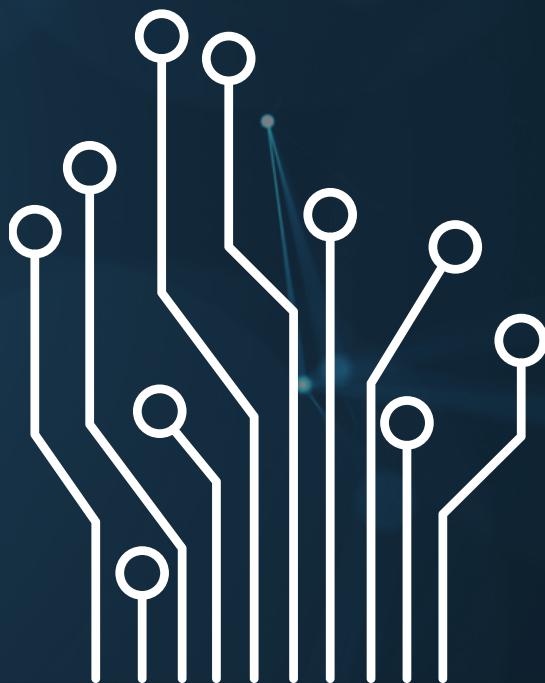
- Batch normalisation
- Adam optimizer
- categorical crossentropy loss function



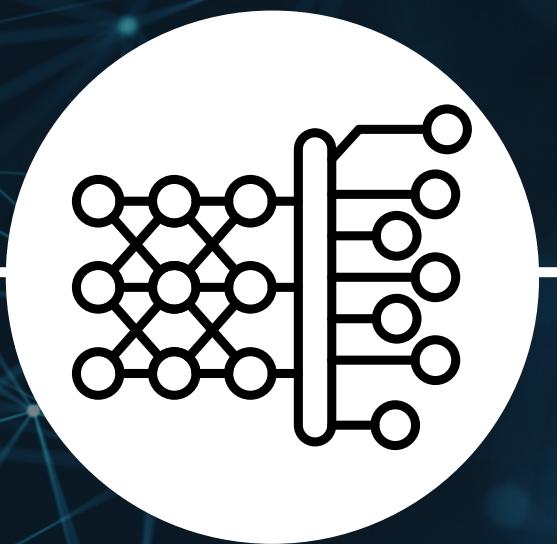
# CNN

```
# Construction du modèle CNN
cnn_model = keras.Sequential([
    keras.layers.Conv2D(32, (3, 3), activation="relu", input_shape=(32, 32, 3)),
    keras.layers.MaxPooling2D((2, 2)),
    keras.layers.Flatten(),
    keras.layers.Dense(64, activation="relu"),
    keras.layers.BatchNormalization(),
    keras.layers.Dense(6, activation="softmax")
])

# Compilation et entraînement du modèle CNN
cnn_model.compile(optimizer="adam", loss="categorical_crossentropy", metrics=["accuracy"])
cnn_history = cnn_model.fit(x_train, y_train, epochs=20, validation_data=(x_val, y_val))
```



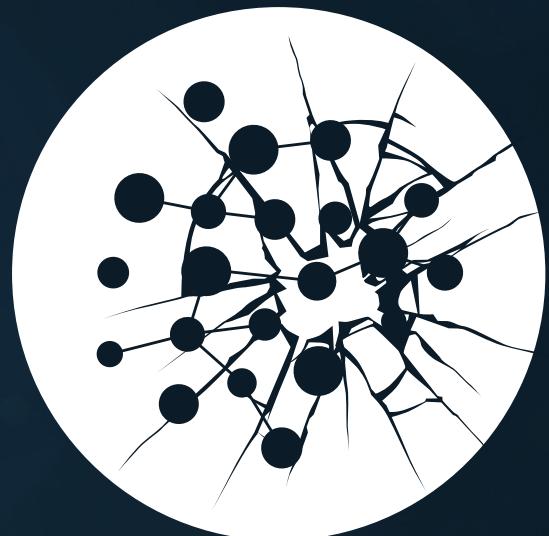
- Batch normalisation
- Adam optimizer
- categorical crossentropy loss function



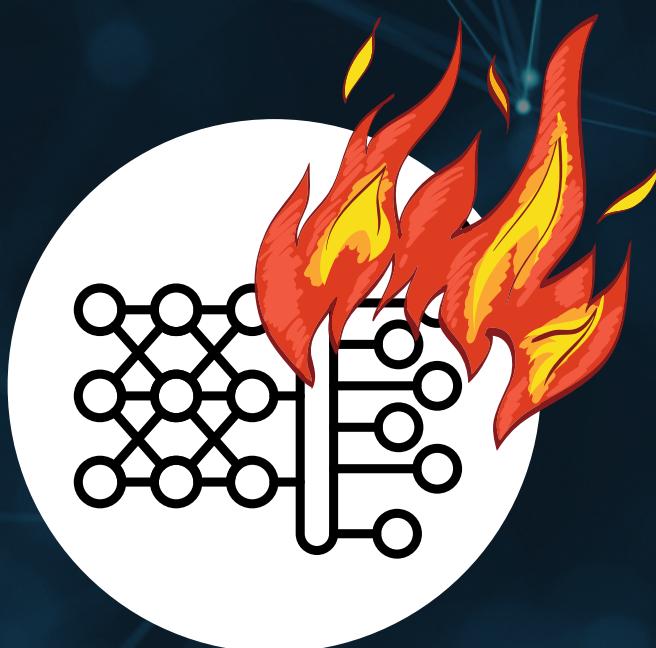
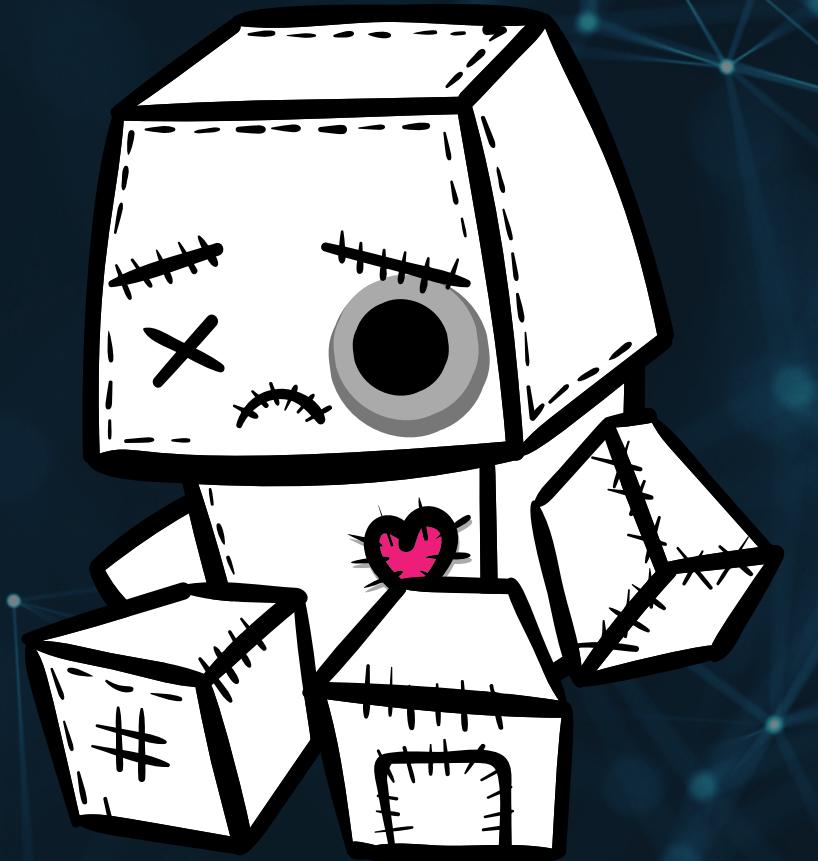
# Comparaison des résultats

---

Les résultats des deux modèles sont assez médiocres et le CNN souffre d'un problème d'overfitting



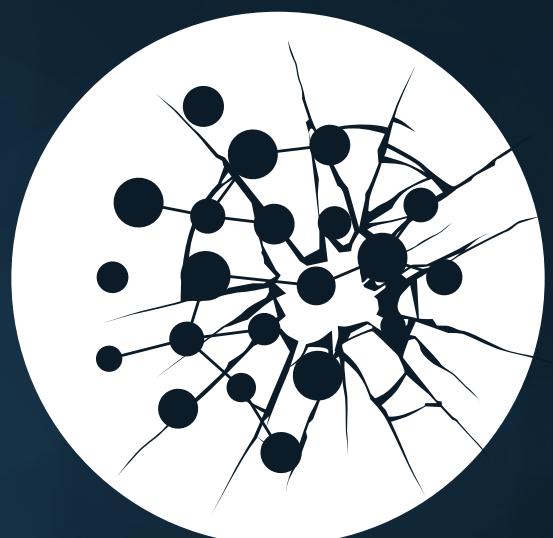
MLP



CNN

# Comparaison des résultats

Les résultats des deux modèles sont assez médiocres et le CNN souffre d'un problème d'overfitting



MLP

Matrice de confusion MLP :  
[[411 156 76 114 145 98]  
 [ 89 421 60 200 174 56]  
 [179 132 309 89 172 119]  
 [ 72 262 39 440 107 80]  
 [ 78 128 66 58 643 27]  
 [ 71 125 64 91 57 592]]

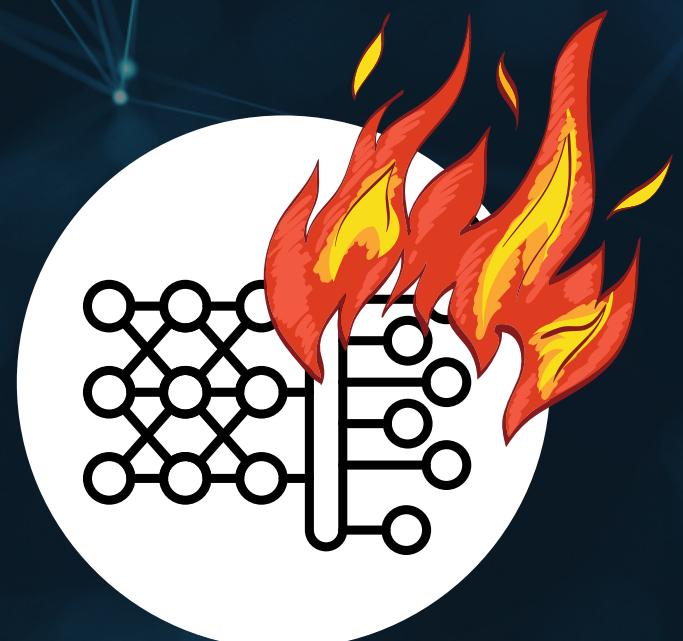
	precision	recall	f1-score	support
0	0.46	0.41	0.43	1000
1	0.34	0.42	0.38	1000
2	0.50	0.31	0.38	1000
3	0.44	0.44	0.44	1000
4	0.50	0.64	0.56	1000
5	0.61	0.59	0.60	1000
accuracy			0.47	6000
macro avg	0.48	0.47	0.47	6000
weighted avg	0.48	0.47	0.47	6000

# Comparaison des résultats

Les résultats des deux modèles sont assez médiocres et le CNN souffre d'un problème d'overfitting

```
Matrice de confusion CNN :  
[[ 589 111 111  57  73  59 ]  
 [ 135 507  94 118  86  60 ]  
 [ 131 127 551  47  70  74 ]  
 [ 139 268  54 430  36  73 ]  
 [ 100 103  70  25 681  21 ]  
 [  96   80  96  54  12 662 ]]
```

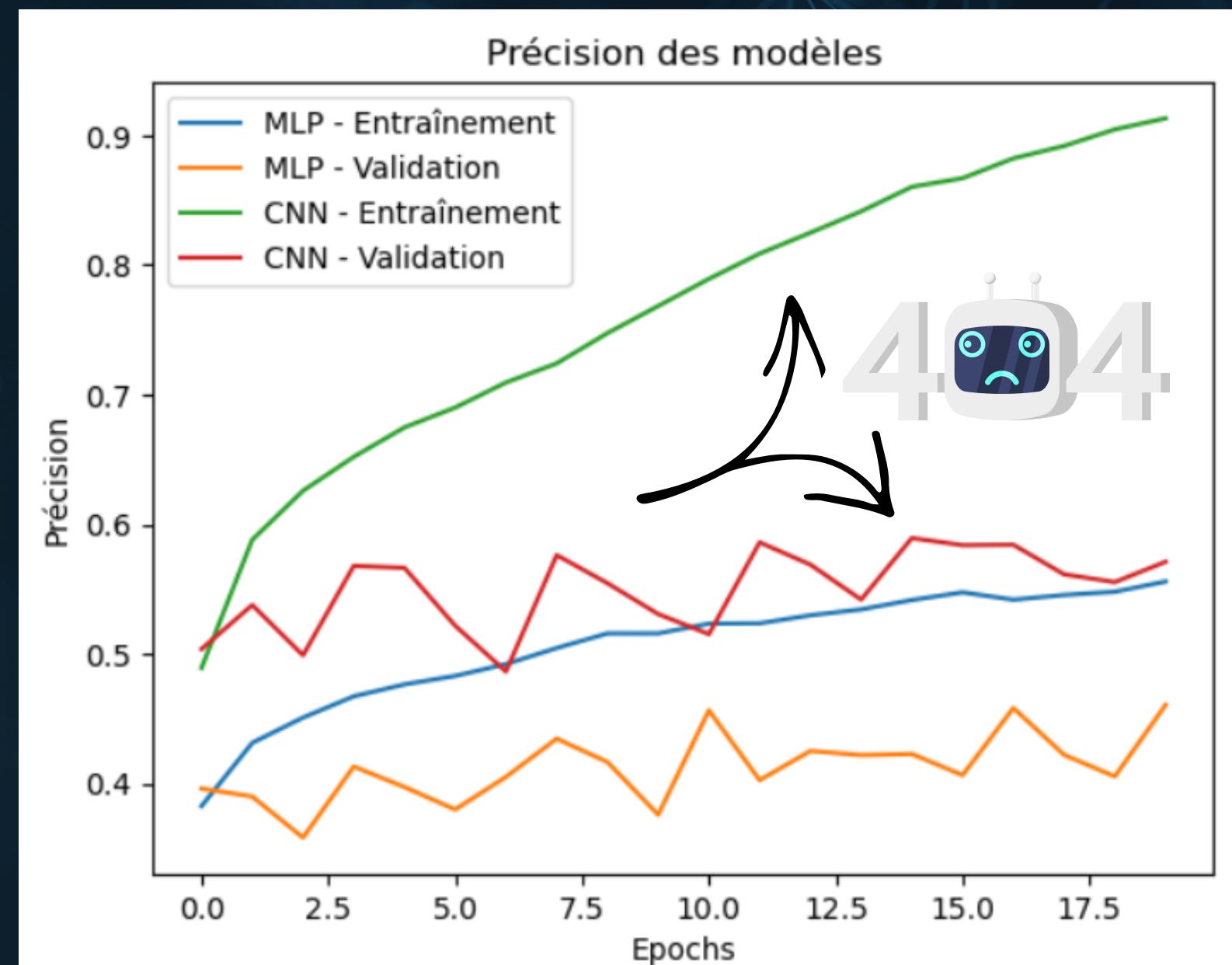
	precision	recall	f1-score	support
0	0.49	0.59	0.54	1000
1	0.42	0.51	0.46	1000
2	0.56	0.55	0.56	1000
3	0.59	0.43	0.50	1000
4	0.71	0.68	0.70	1000
5	0.70	0.66	0.68	1000
accuracy			0.57	6000
macro avg	0.58	0.57	0.57	6000
weighted avg	0.58	0.57	0.57	6000



CNN

# Comparaison des résultats

Les résultats des deux modèles sont assez médiocres et le CNN souffre d'un problème d'overfitting



# Amélioration du CNN



## DROPOUT

Ignorer des parties du réseau de neurones  
afin de limiter le surapprentissage



## EARLY STOPPING

Permet au modèle de s'arrêter avant le  
nombre d'epoch définie en cas de  
convergence  
ou début d'overfitting

# Amélioration du CNN

```
# Remédier à l'overfitting en utilisant le dropout et l'early stopping
dropout_model = keras.Sequential([
    keras.layers.Conv2D(32, (3, 3), activation="relu", input_shape=(32, 32, 3)),
    keras.layers.MaxPooling2D((2, 2)),
    keras.layers.Flatten(),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(64, activation="relu"),
    keras.layers.BatchNormalization(),
    keras.layers.Dense(6, activation="softmax")
])

early_stopping = keras.callbacks.EarlyStopping(patience=3)

dropout_model.compile(optimizer="adam", loss="categorical_crossentropy", metrics=["accuracy"])
dropout_history = dropout_model.fit(x_train, y_train, epochs=100, validation_data=(x_val, y_val), callbacks=[early_stopping])
```

# Résultats du CNN amélioré

---

```
Accuracy du modèle MLP sur l'ensemble de validation : 0.45366665720939636
Accuracy du modèle MLP sur l'ensemble de test : 0.4568333327770233
Accuracy du modèle CNN sur l'ensemble de validation : 0.4893333315849304
Accuracy du modèle CNN sur l'ensemble de test : 0.4883333444595337
Accuracy du modèle avec dropout sur l'ensemble de validation : 0.612666666507721
Accuracy du modèle avec dropout sur l'ensemble de test : 0.6048333048820496
```

# Conclusion

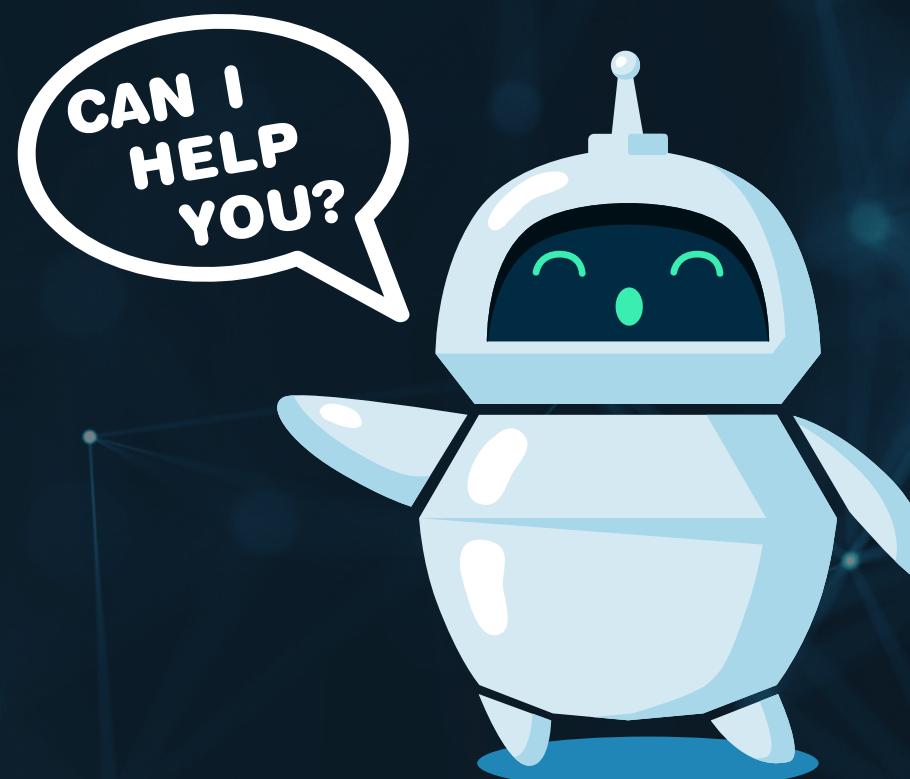
---

- CNN plus adapté à la problématique
- Résultats pouvant être amélioré
- Modification des hyperparamètres et de l'architecture



# Thank You!

Des questions ?



Pierre-Alexis Lebair