PROJET ANDROIDE MASTER

Reconnaissance de nombres contenus sur surfaces sphériques : application au tirage du loto

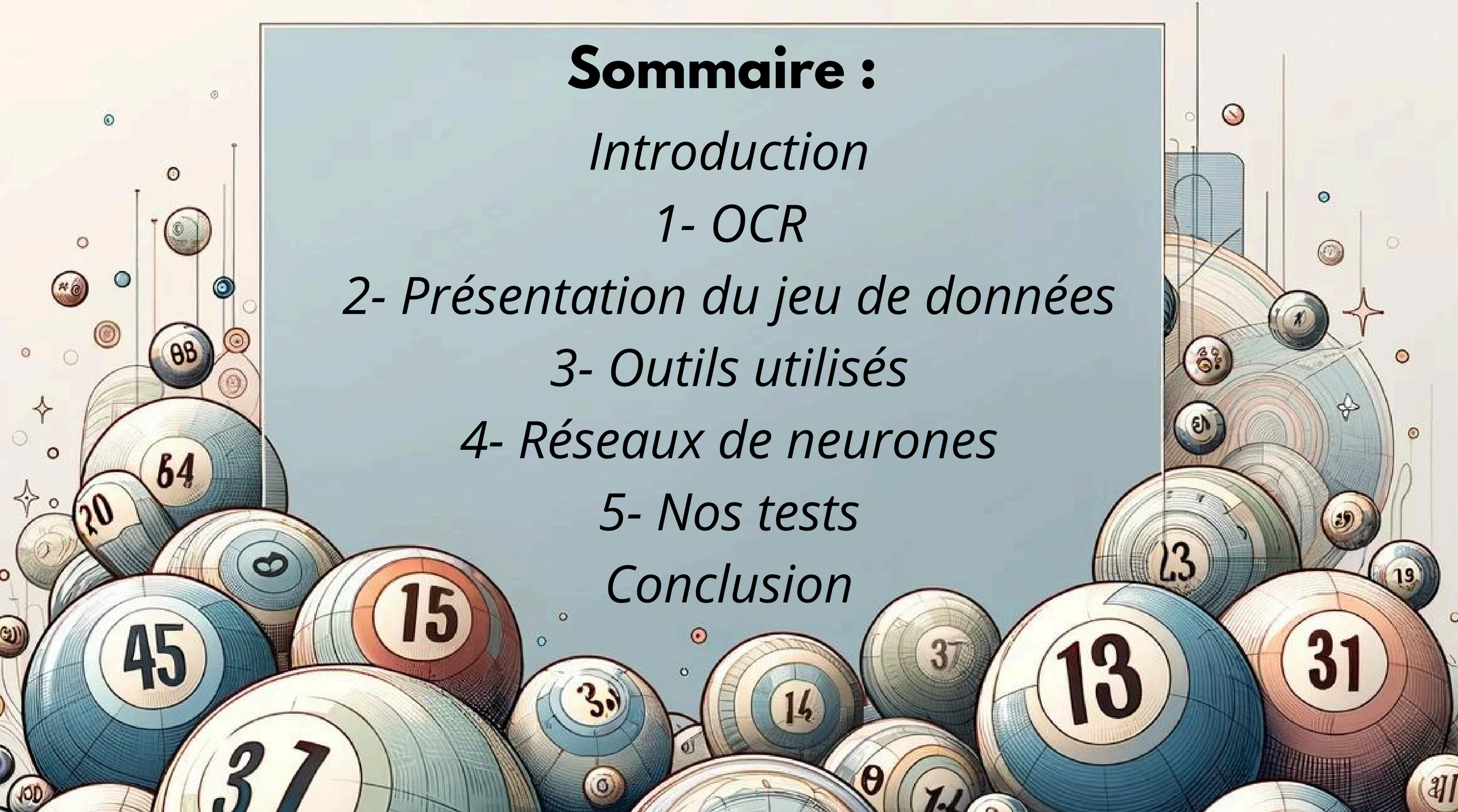
Présenté par

LECLERC Antony - AMOZIEG Shirel - CAI Zhirui

Sous la direction de Thibaut Lust

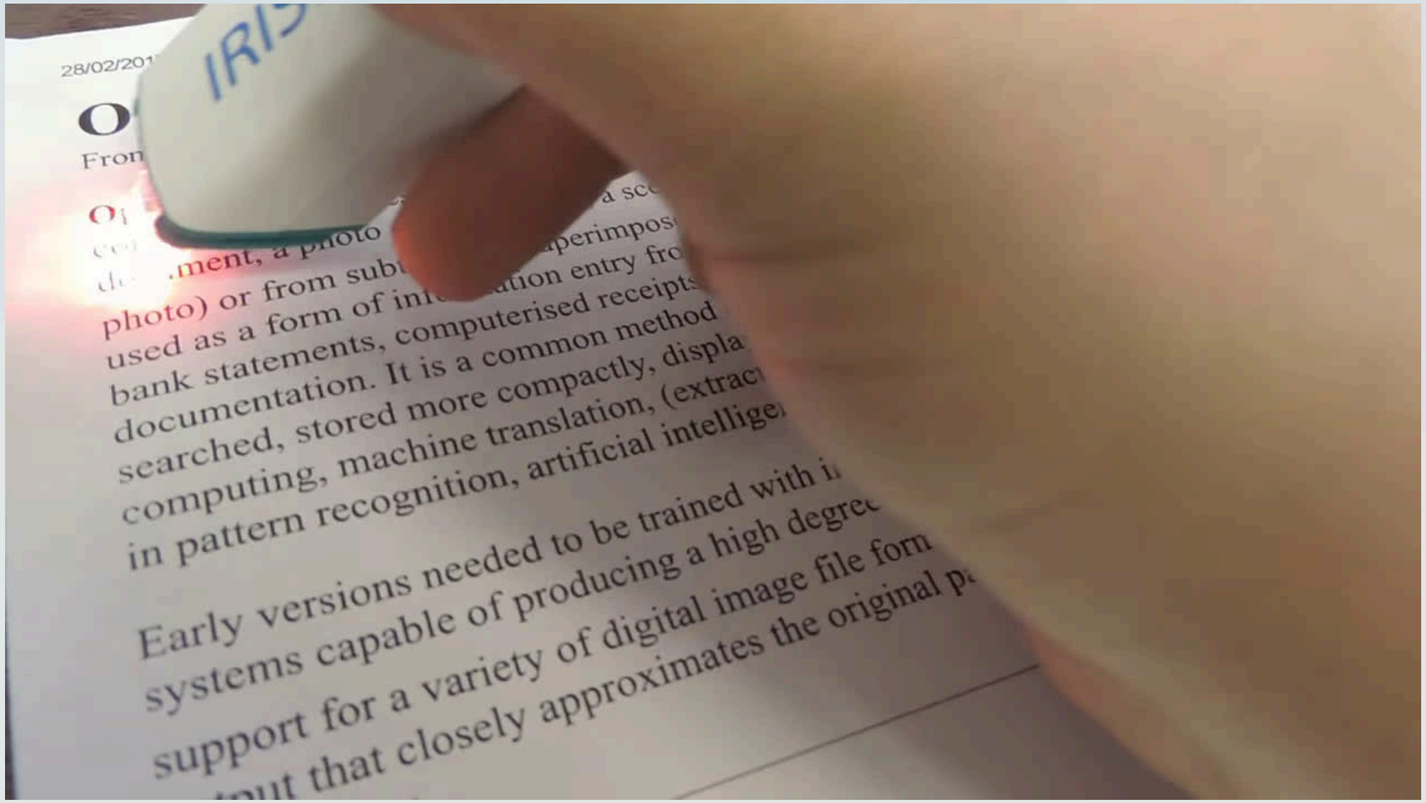
SORBONNE UNIVERSITÉ

Sommaire :



1

Optical Character Recognition (OCR)

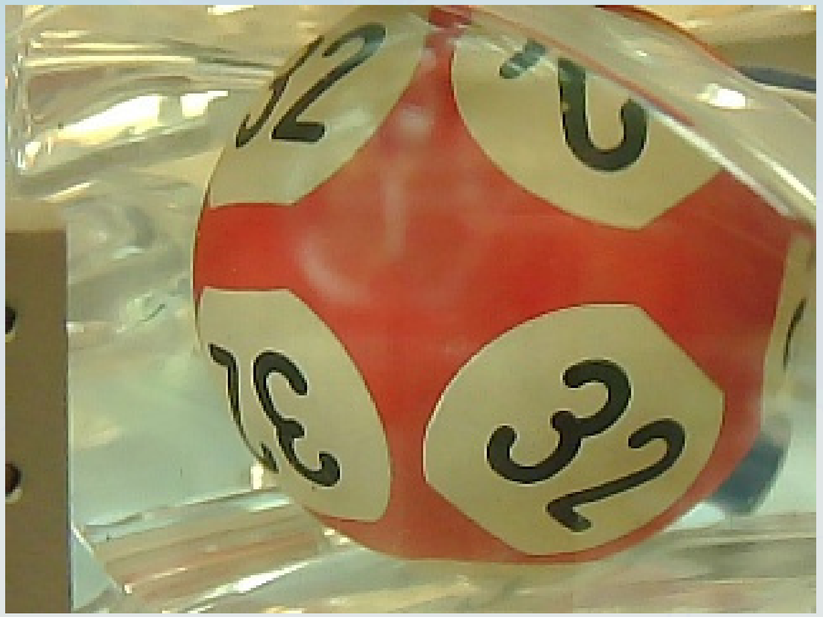


2

Présentation du jeu de données

2

Présentation du jeu de données

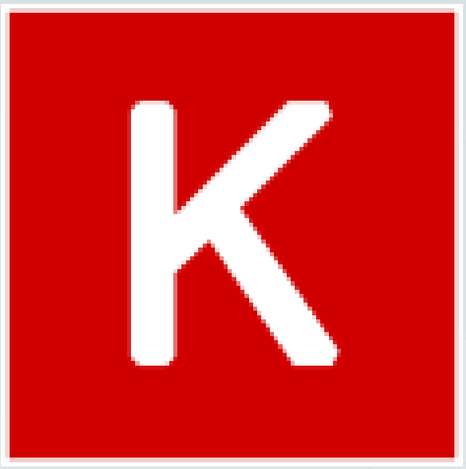


3

Outils utilisés

+

+



Python 3.12

Tensorflow 2.16

Keras 3.3.3

3

Pourquoi Python ?

- Langage simple et accessible
- Bibliothèques (NumPy, Pillow, Matplotlib, opencv-python…)
- Vaste communauté d’utilisateurs

Python 3.12

3

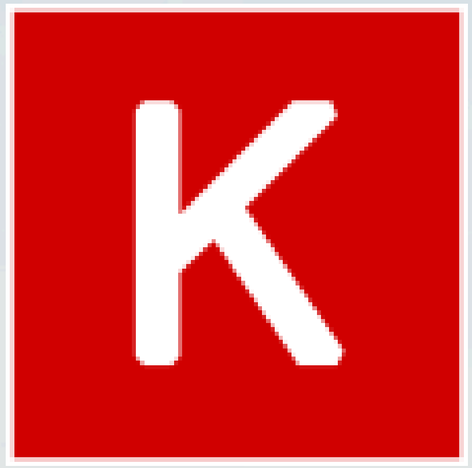
Pourquoi Tensorflow ?

Tensorflow 2.16

- Développée par Google
- Bibliothèque puissante et flexible
- Utilisateurs avancés voulant utiliser le plein potentiel de Tensorflow
- API détaillée
- Écrite en C++, CUDA, Python

3

Pourquoi Keras ?

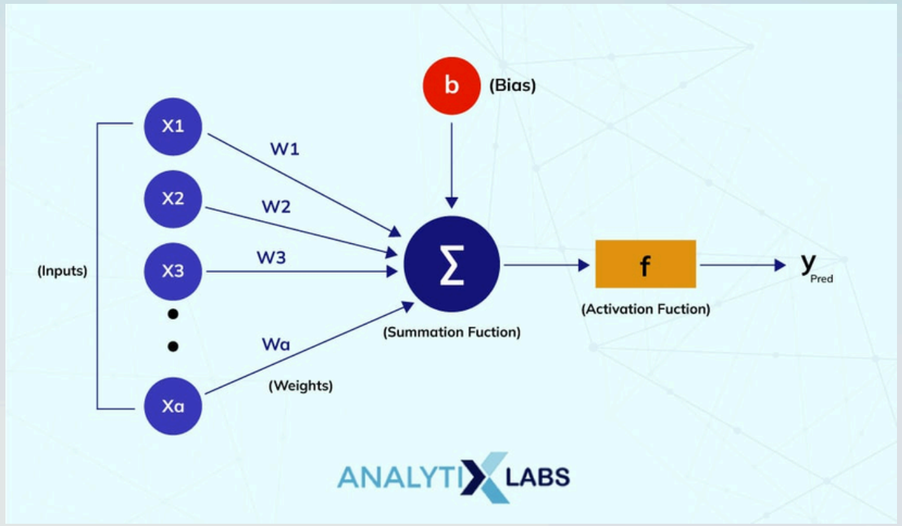


Keras 3.3.3

- API “Beginner Friendly”
- Prototypes rapides pour des projets rapides
- Ecrite en Python
- Intégré à Tensorflow

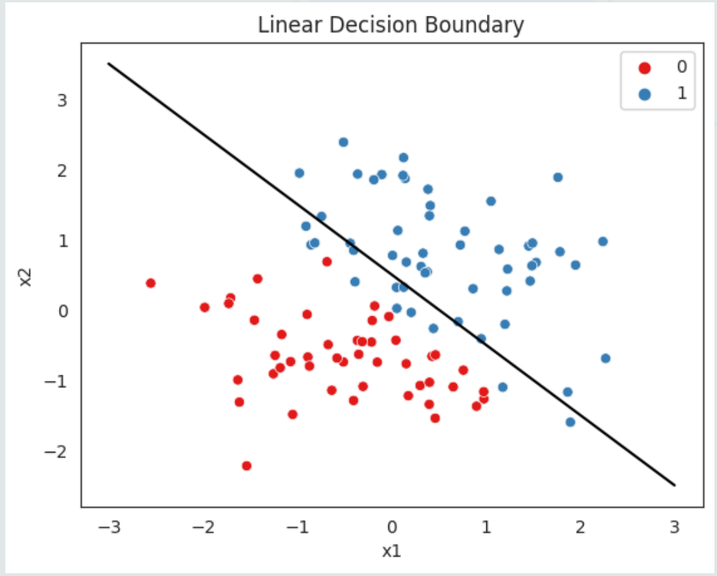
4

L’origine des réseaux de neurones



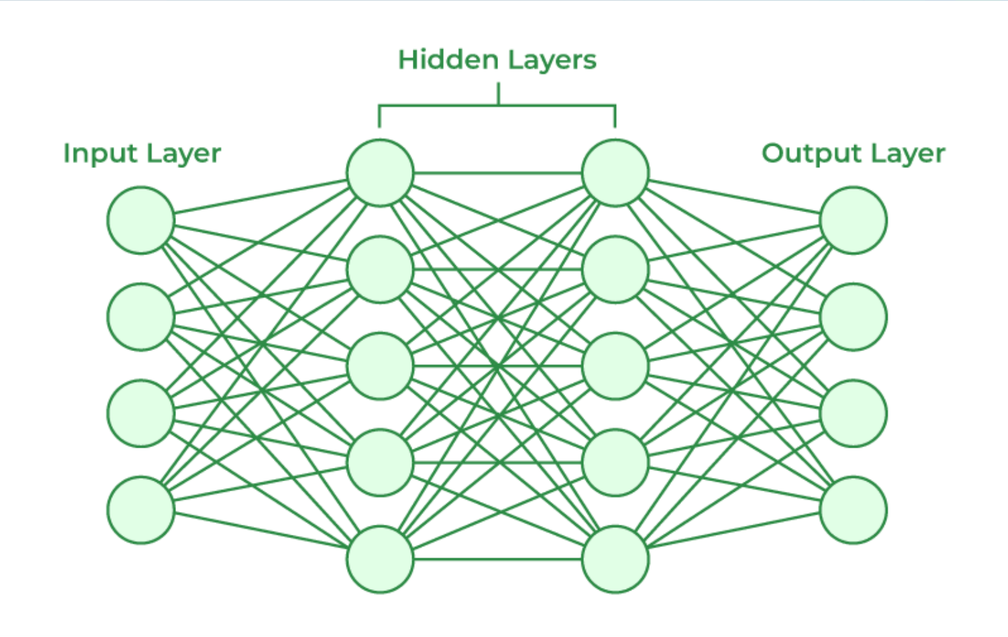
Perceptron

- Créé en 1958 par F. Rosenblatt
- Classification binaire représentable graphiquement



4

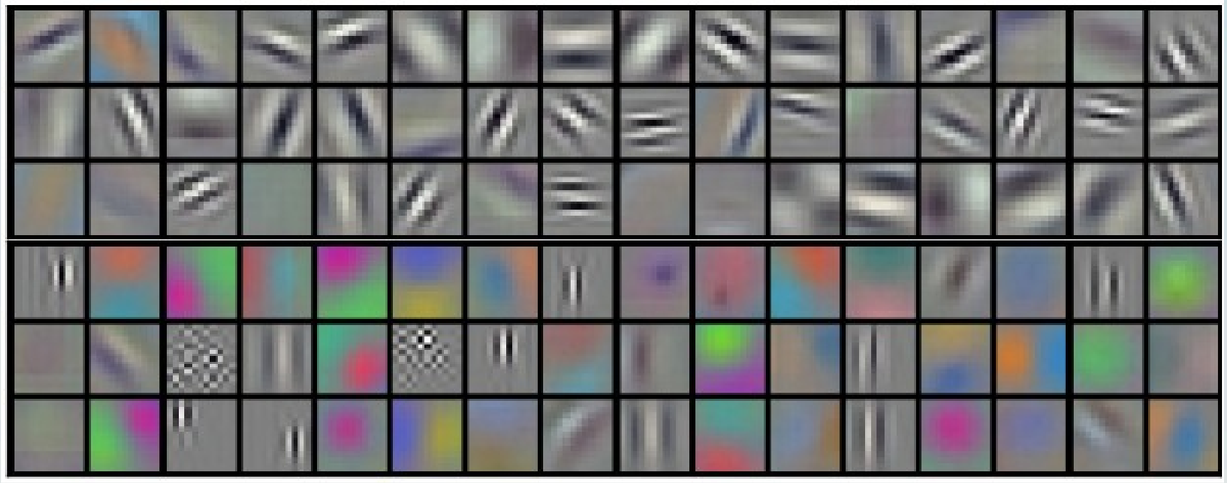
Réseaux de Neurones Convolutionnels



- Adapté à la vision par ordinateur par la convolution
- Classification multi-classes

4

Filtres et Features



- CNN « AlexNet »
- Filtres de (Alex) Krizhevsky
- Filtres devéloppés sur la base de données ImageNet

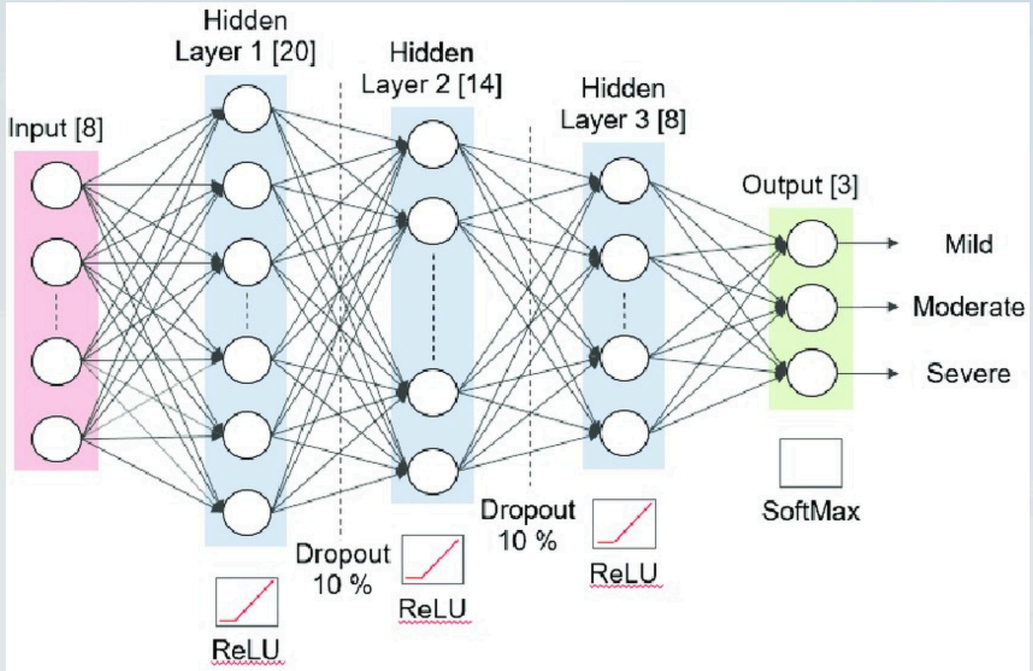
4

Fonctions d’activation

```
model.add(tf.keras.layers.Conv2D(32, (3, 3), activation='relu',
    input_shape=(240, 320, 3)))
model.add(tf.keras.layers.MaxPooling2D((2, 2)))
model.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu'))
model.add(tf.keras.layers.MaxPooling2D((2, 2)))

model.add(tf.keras.layers.Flatten())

model.add(tf.keras.layers.Dense(128, activation='relu'))
model.add(tf.keras.layers.Dense(64, activation='relu'))
model.add(tf.keras.layers.Dense(51, activation='softmax'))

model.compile(optimizer='adam',
    loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

4

Fonctions d’activation



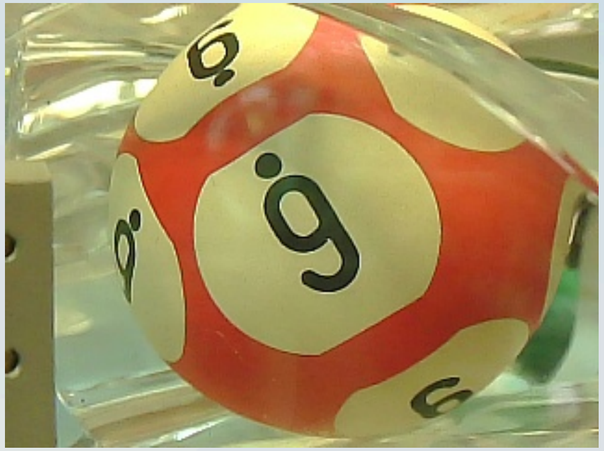
- RELU dans les couches d’entrée et les couches cachées
- Softmax en couche de sortie

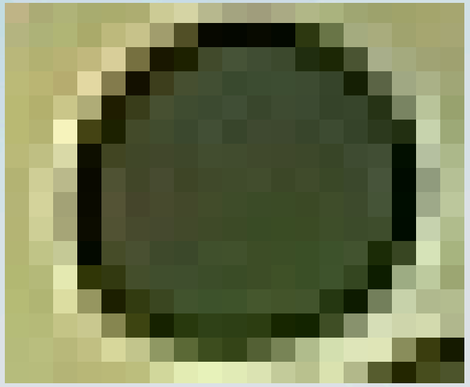
4

Features pour les numéros de LOTO

4

Features pour les numéros de LOTO
Cas particulier du 6 et du 9





4

Représentation des images

```
array([[[255, 253, 255],
        [255, 254, 255],
        [255, 255, 253],
        ...,

        [[213, 255, 205],
         [210, 252, 202],
         [195, 237, 187],
         ...,
         [133, 168, 136],
         [133, 166, 135],
         [133, 164, 133]]], dtype=uint8)
```

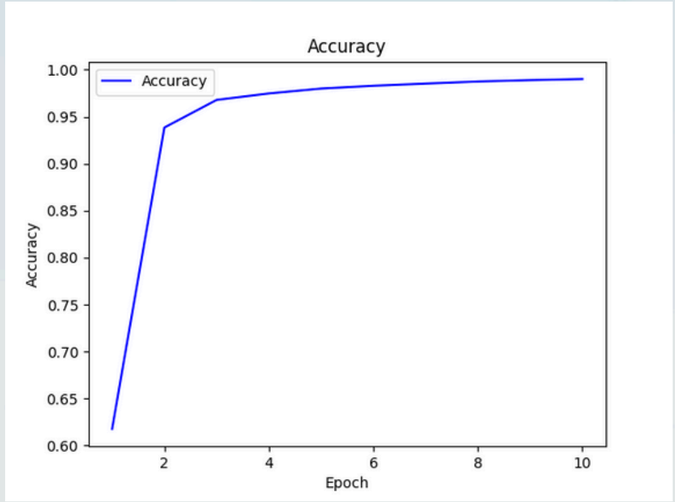
5

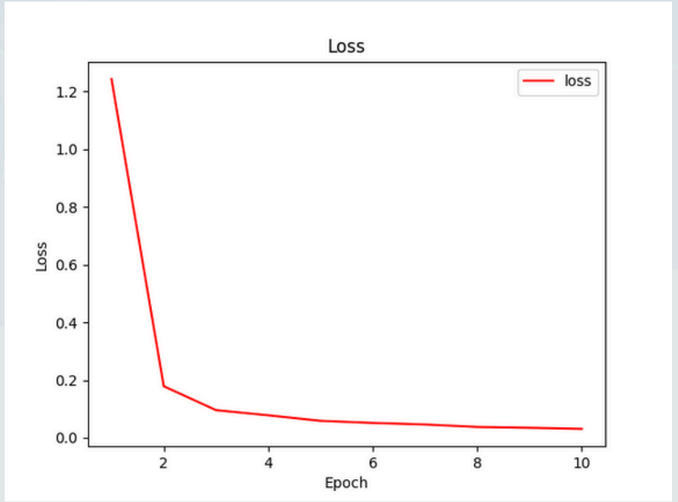
NOS TESTS

5

Exemple d’un entraînement

```
(254, 70, 70, 3)
Début génération
Fin génération
Nombre d'images total : 79376
x_train : (63500, 70, 70, 3)
x_test : (15876, 70, 70, 3)
Epoch 1/10
1985/1985 ━━━━━━━━━━━━━━━━━━━━ 71s 35ms/step - accuracy: 0.3552 - loss: 2.2523
Epoch 2/10
1985/1985 ━━━━━━━━━━━━━━━━━━━━ 69s 35ms/step - accuracy: 0.9301 - loss: 0.2041
Epoch 3/10
1985/1985 ━━━━━━━━━━━━━━━━━━━━ 72s 36ms/step - accuracy: 0.9667 - loss: 0.1003
Epoch 4/10
1985/1985 ━━━━━━━━━━━━━━━━━━━━ 75s 38ms/step - accuracy: 0.9747 - loss: 0.0773
Epoch 5/10
1985/1985 ━━━━━━━━━━━━━━━━━━━━ 73s 37ms/step - accuracy: 0.9797 - loss: 0.0578
Epoch 6/10
1985/1985 ━━━━━━━━━━━━━━━━━━━━ 75s 38ms/step - accuracy: 0.9829 - loss: 0.0526
Epoch 7/10
1985/1985 ━━━━━━━━━━━━━━━━━━━━ 74s 37ms/step - accuracy: 0.9863 - loss: 0.0440
Epoch 8/10
1985/1985 ━━━━━━━━━━━━━━━━━━━━ 73s 37ms/step - accuracy: 0.9891 - loss: 0.0324
Epoch 9/10
1985/1985 ━━━━━━━━━━━━━━━━━━━━ 72s 36ms/step - accuracy: 0.9894 - loss: 0.0324
Epoch 10/10
1985/1985 ━━━━━━━━━━━━━━━━━━━━ 71s 36ms/step - accuracy: 0.9914 - loss: 0.0275
```

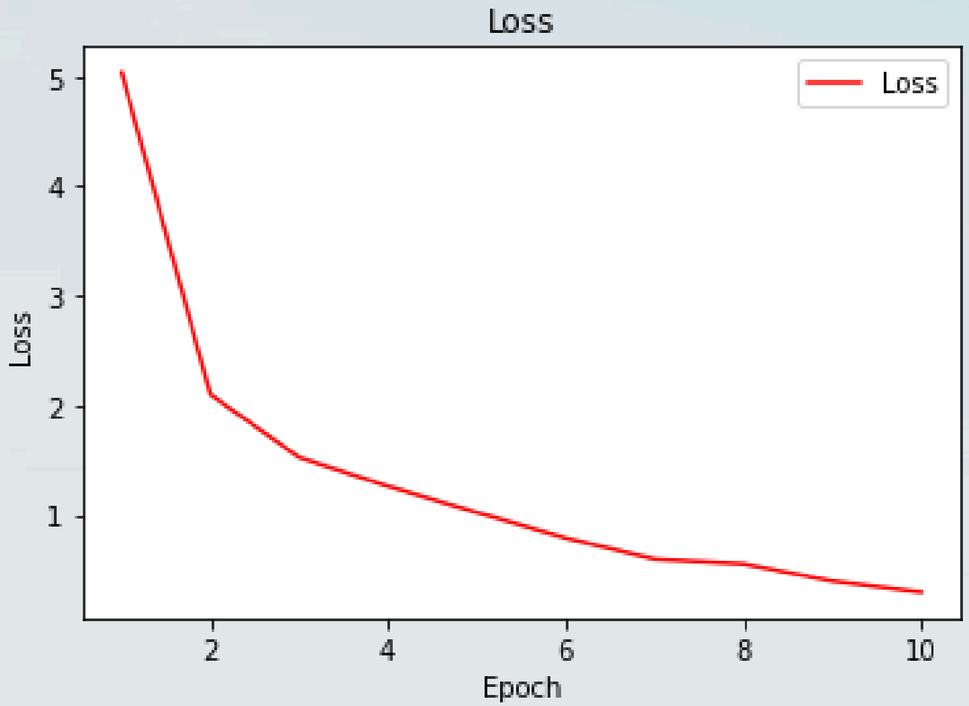


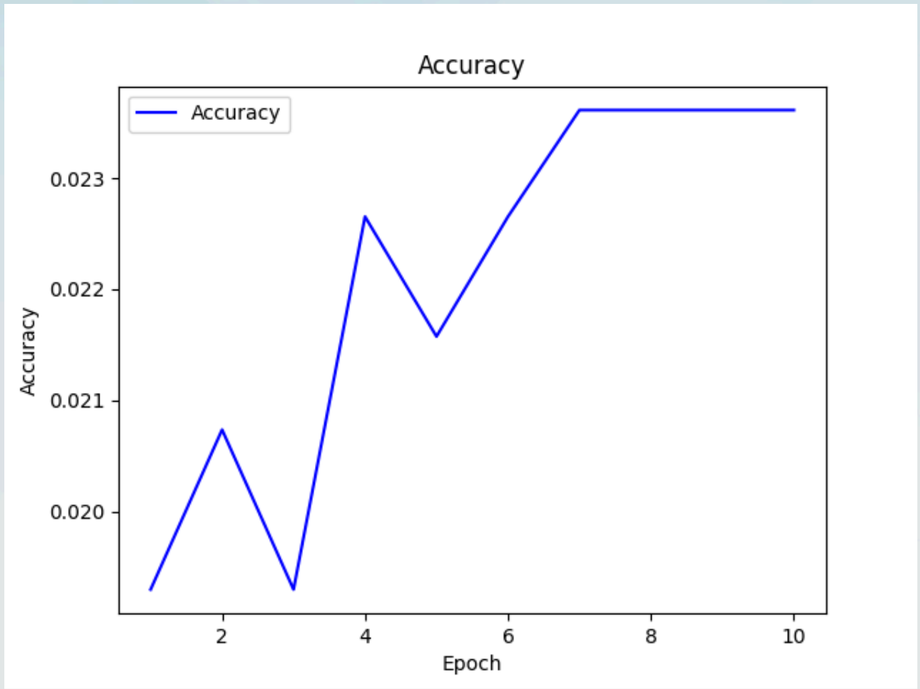


5

Images brutes

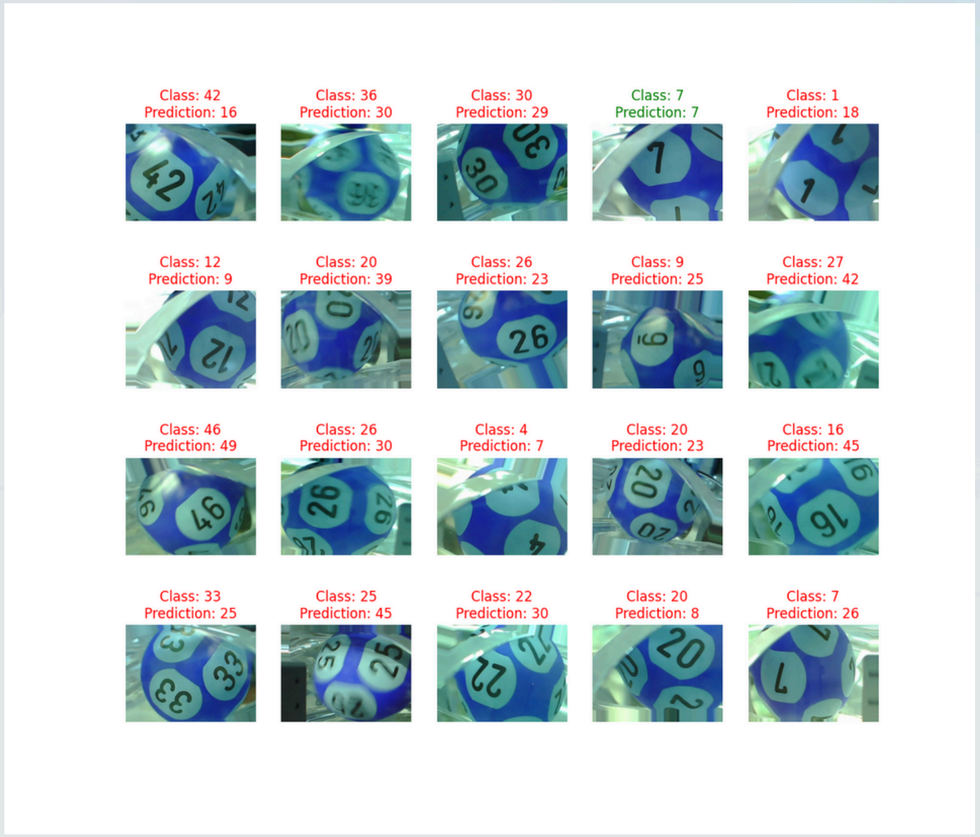
Images en couleur

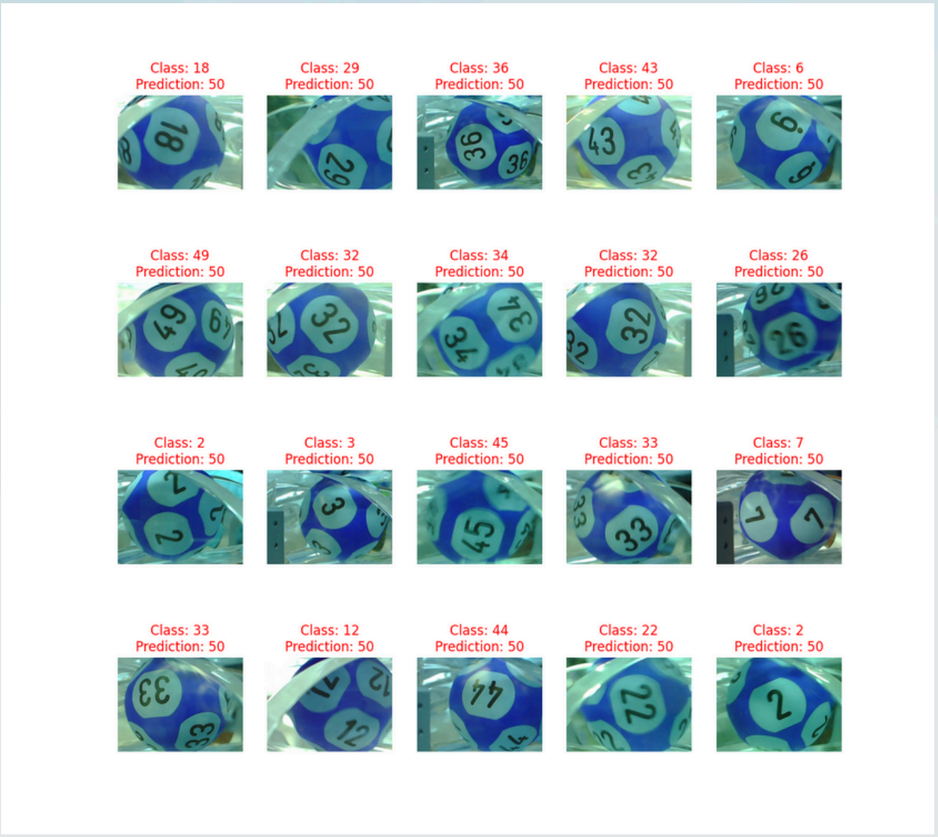




5

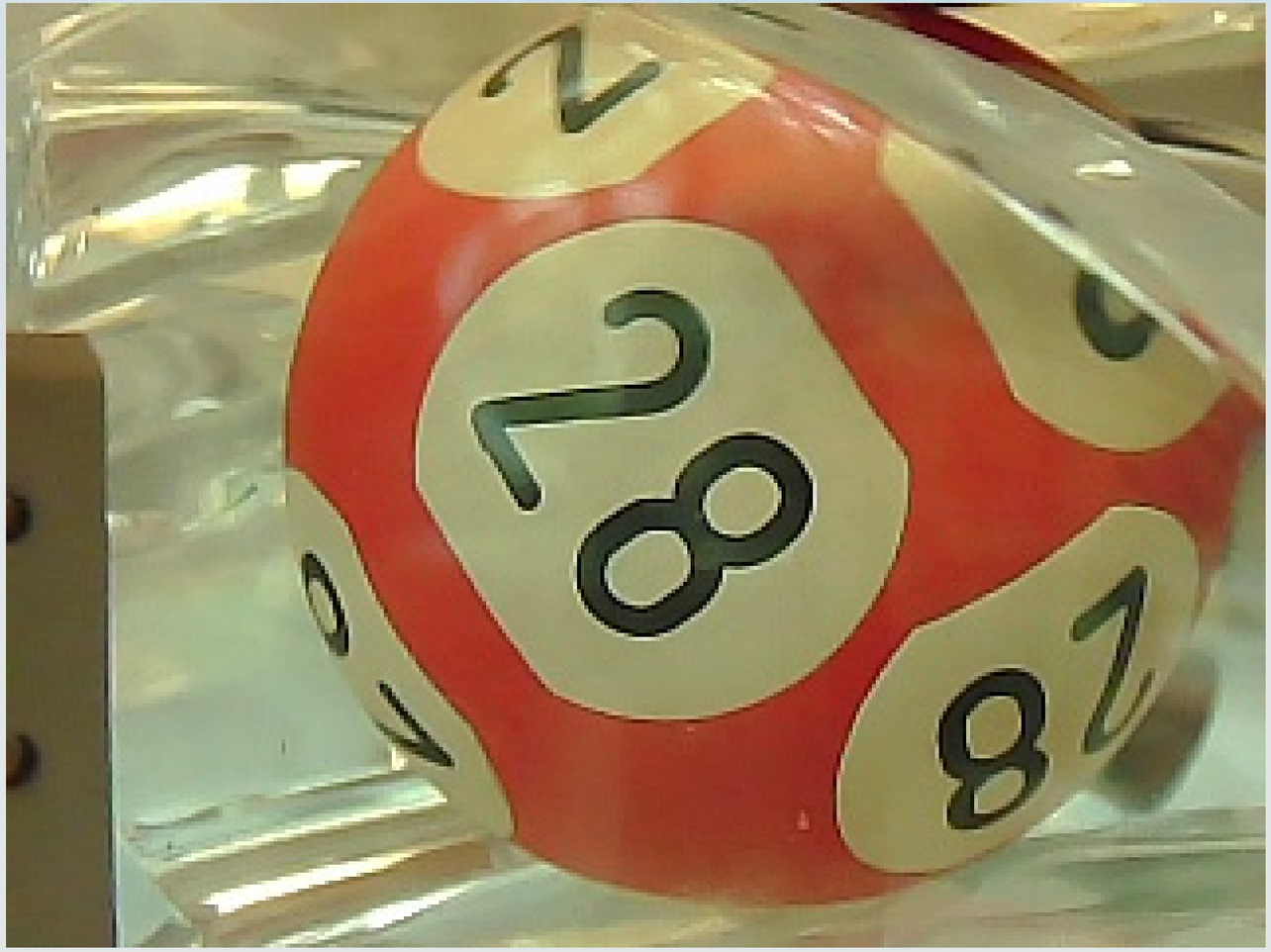
Résultats visuels





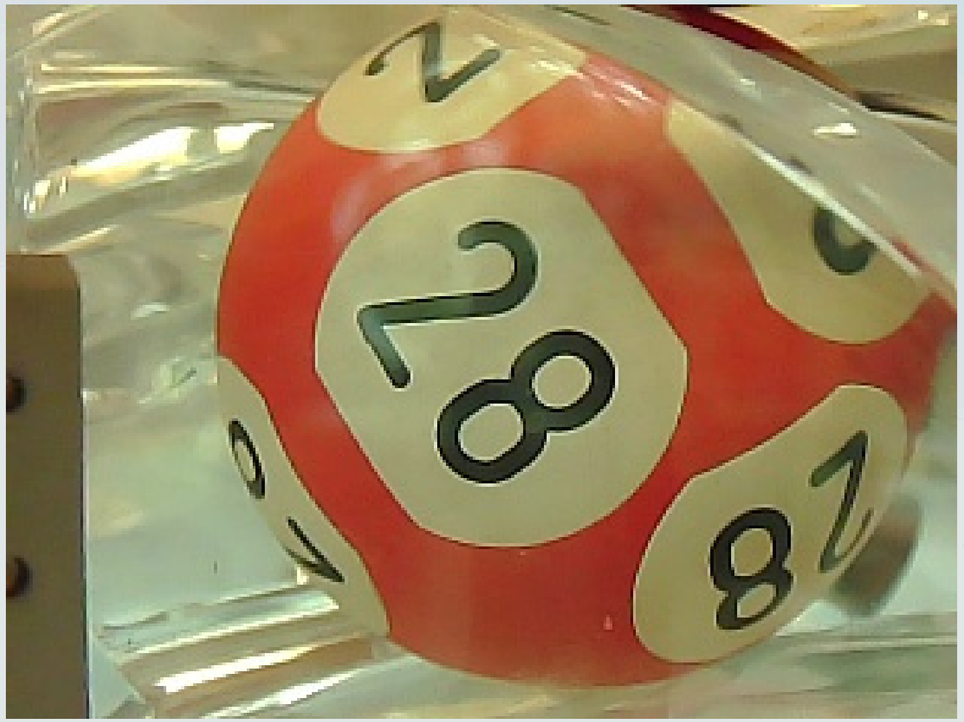
5

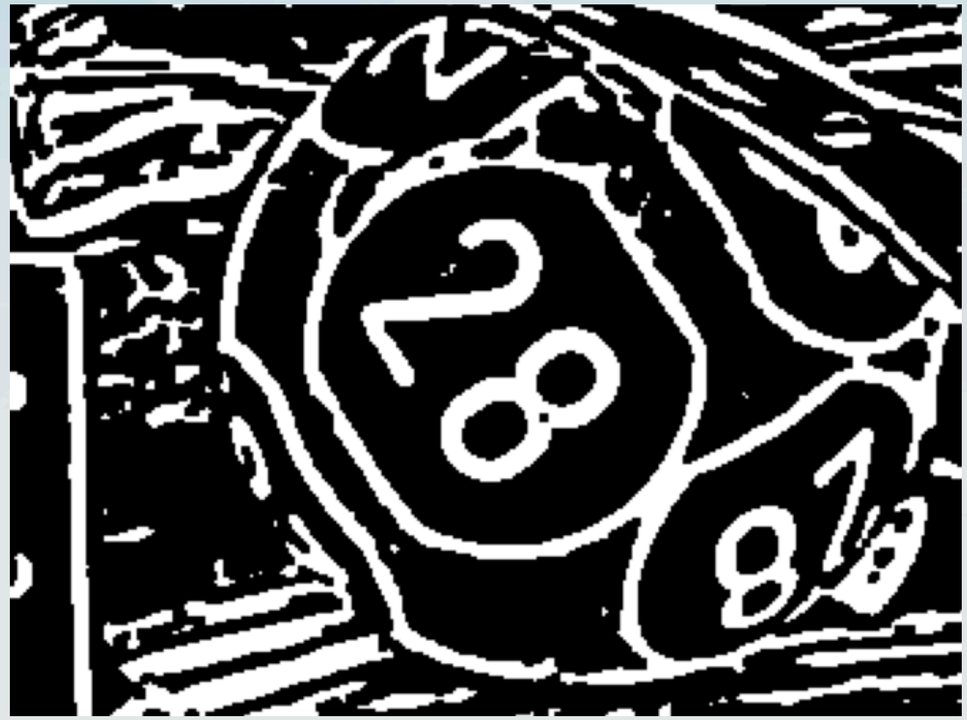
Problème des images (240x320) couleurs



5

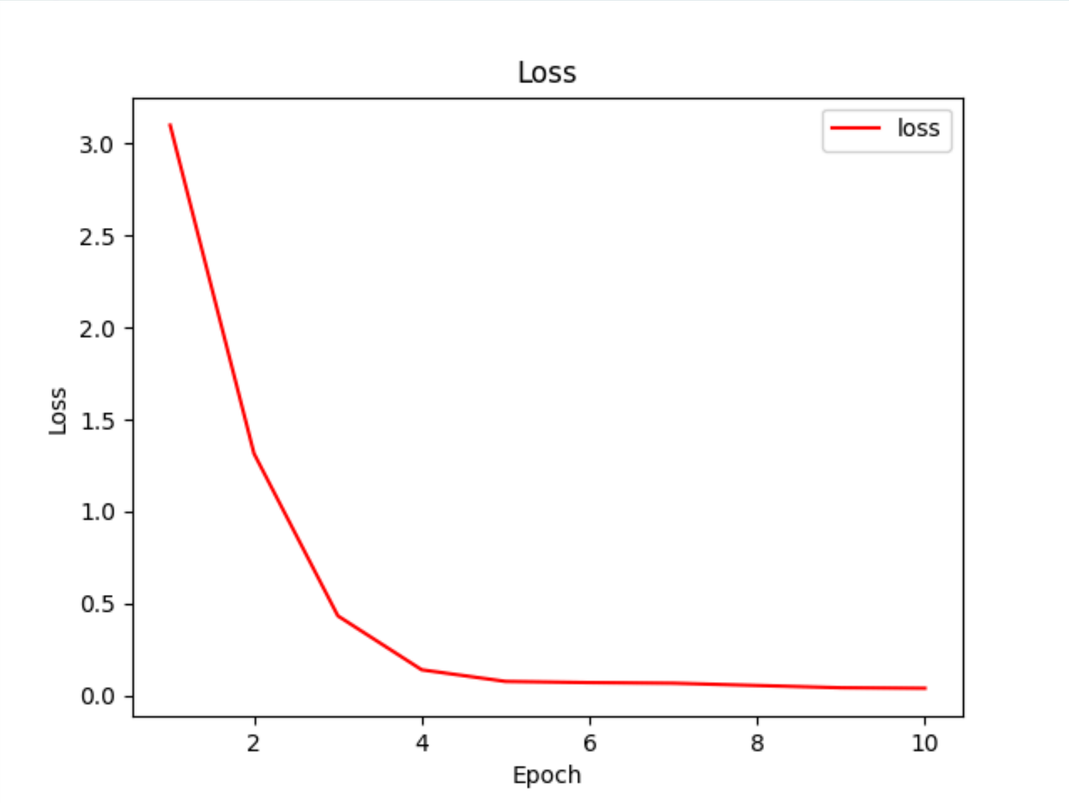
Prétraitement

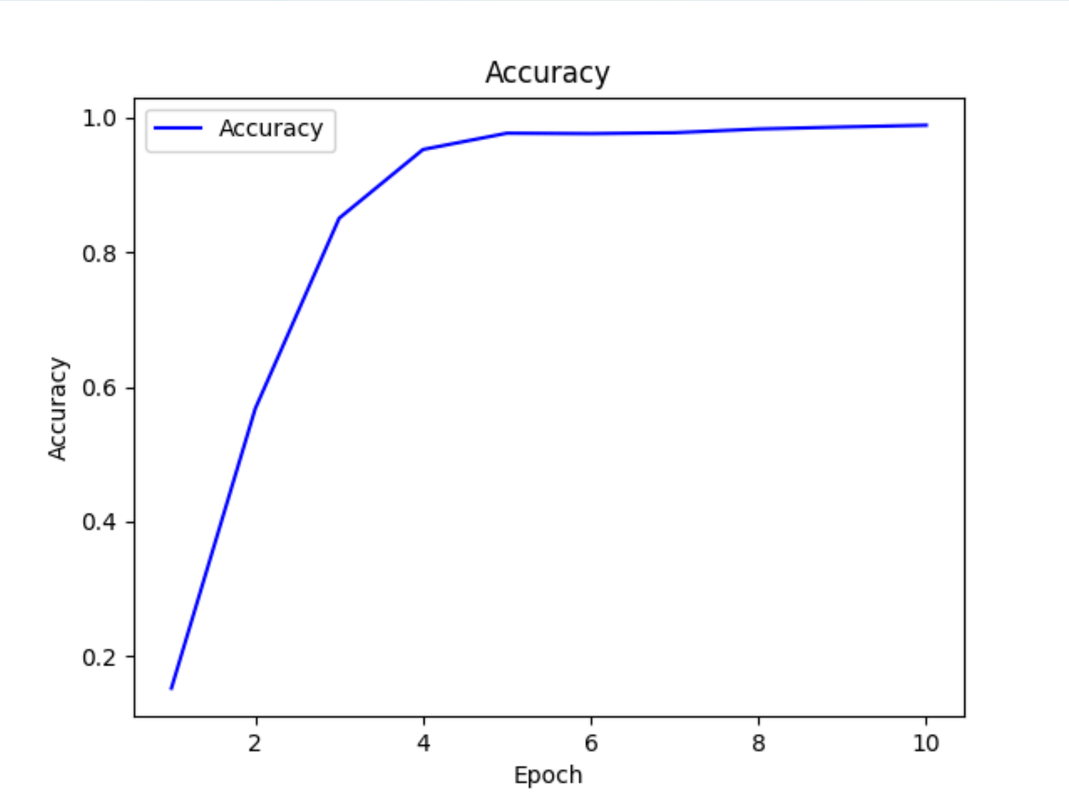




5

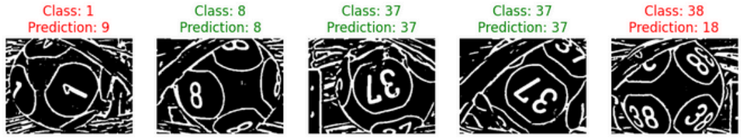
Images pré-traitées

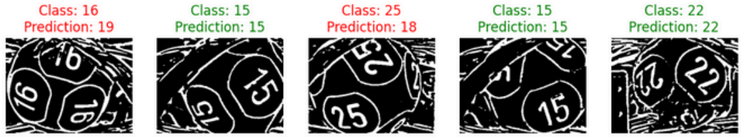


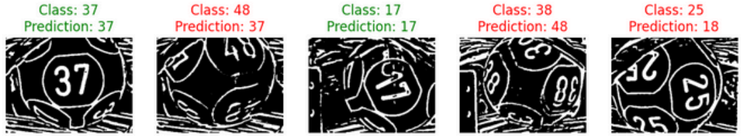


5

Résultats visuels

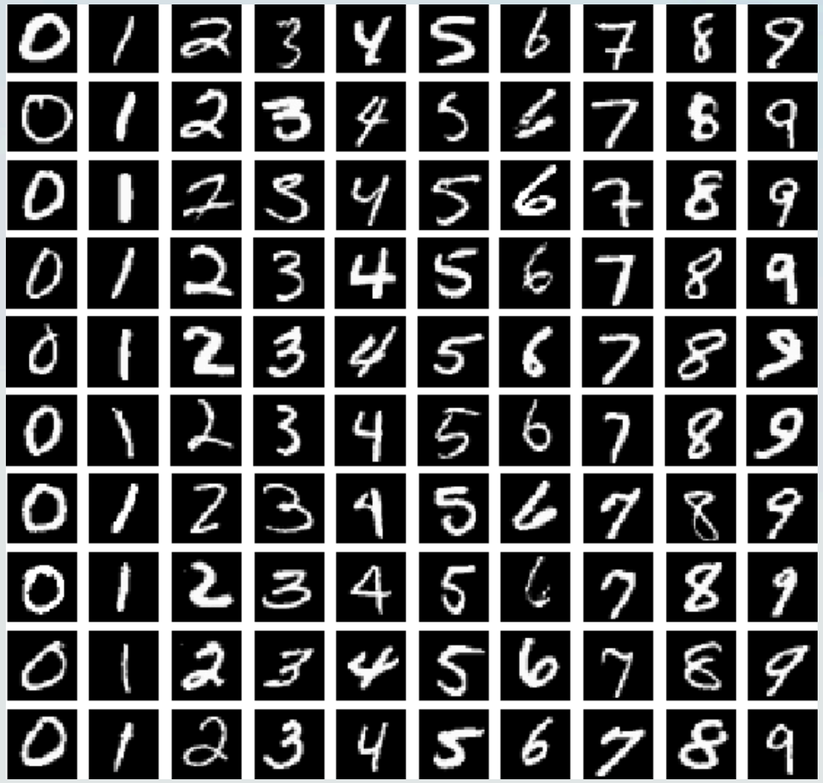






5

MNIST



- Base de données de chiffres écrits à la main
- Très utilisée dans le Machine Learning
- Tests avec notre réseau de neurones satisfaisant à 100% sur cette base de données
- Idée de travailler avec des images où l’on ne voit que les numéros

5

Images rognées

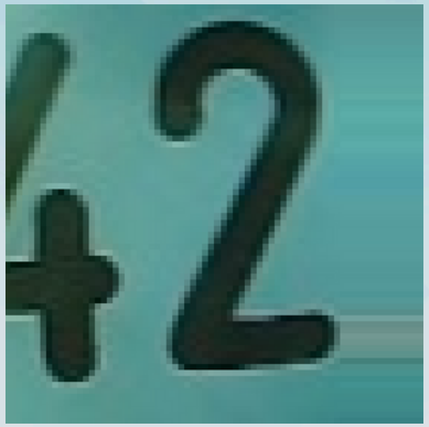
Problème :
Images rognées à la main par nous, peu d'images pour l'entraînement

ImageDataGenerator, augmentation artificielle de notre dataset

5

Images rognées

Image initiale



Décalage horizontal

Décalage vertical

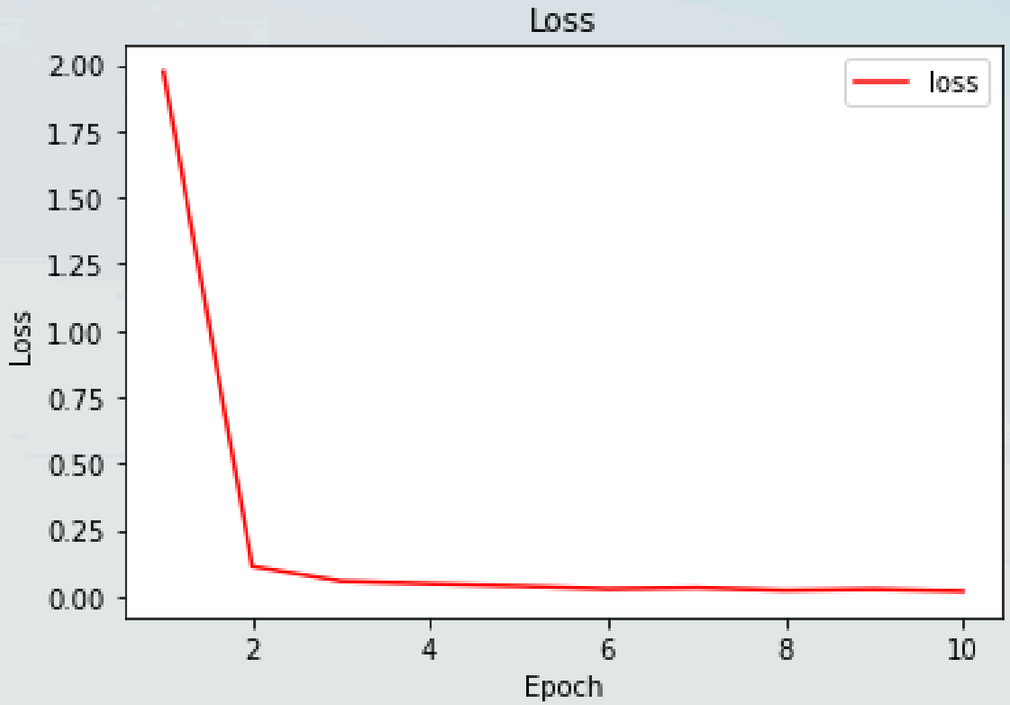
Zoom

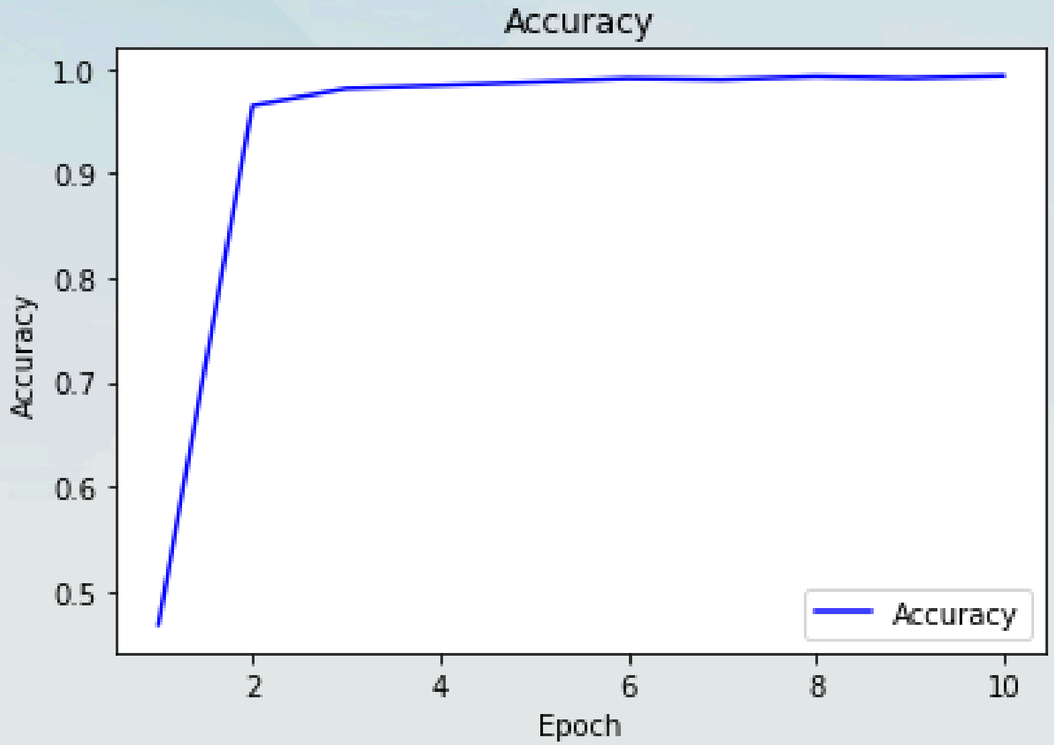
Rotation

5

Images rognées

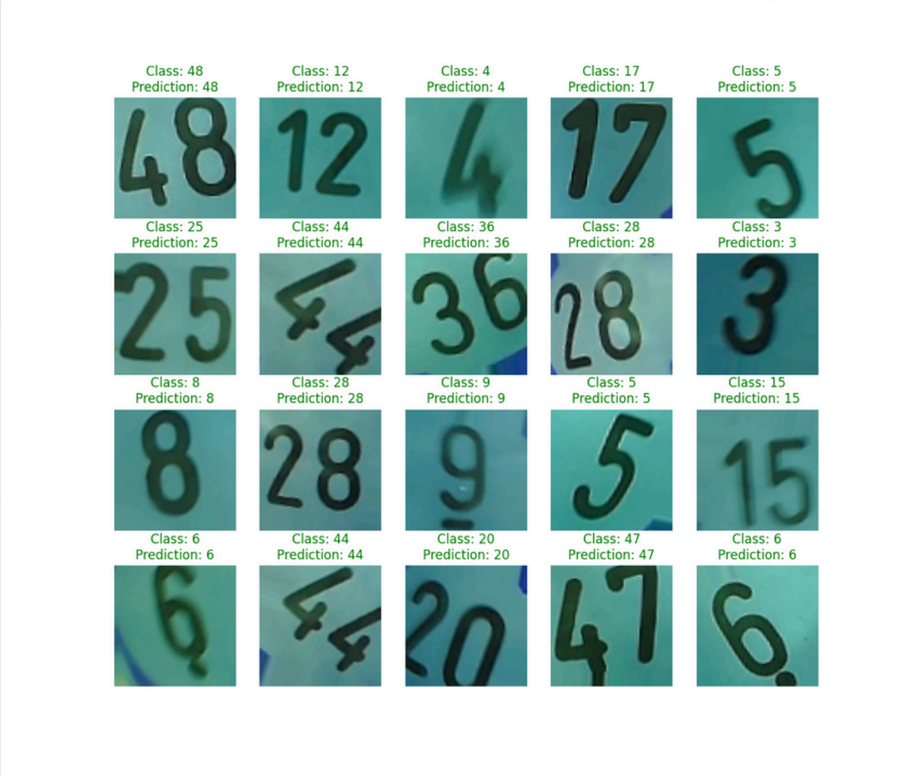
Images en couleur

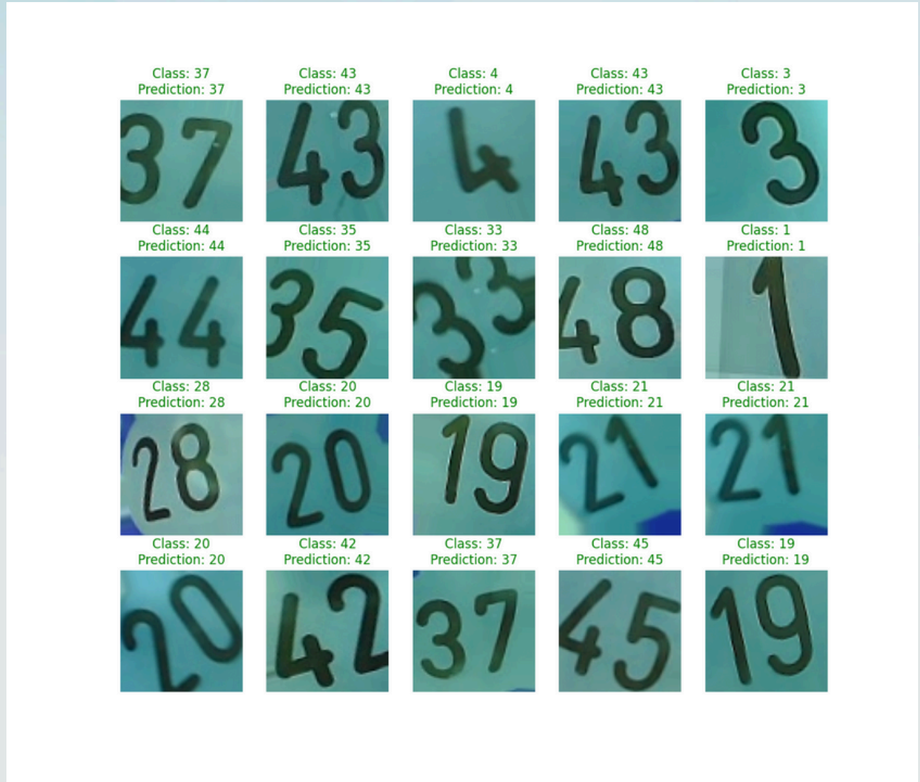




5

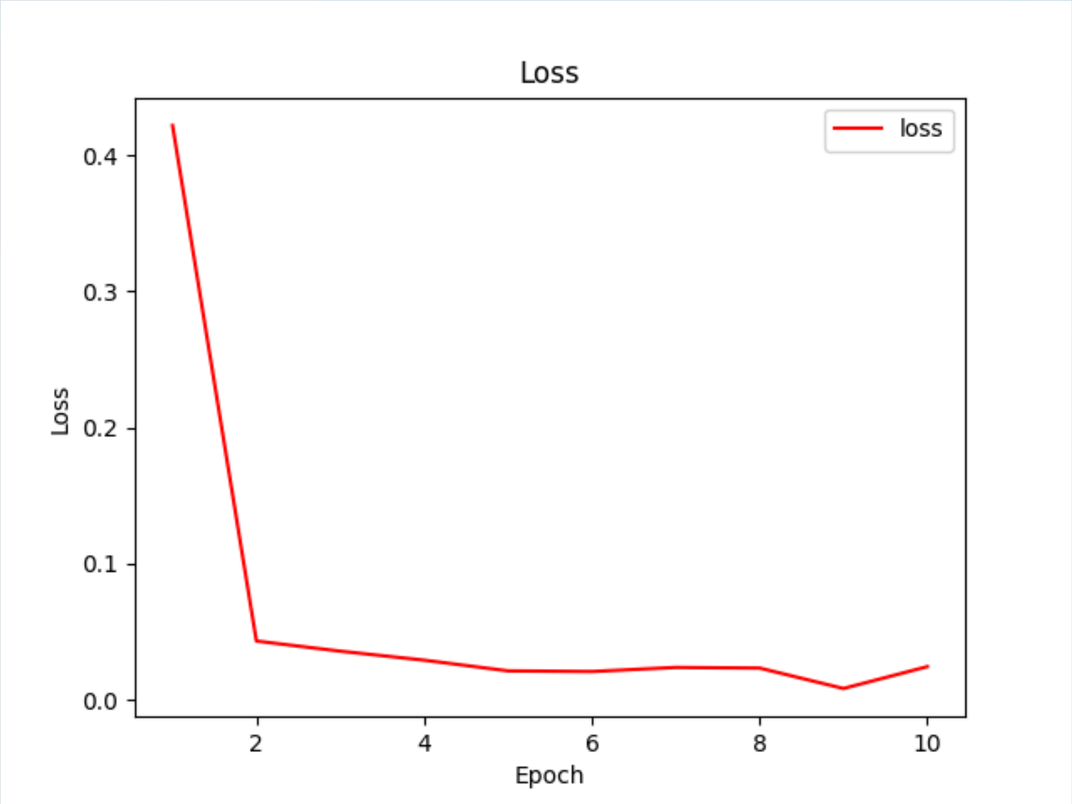
Résultats visuels

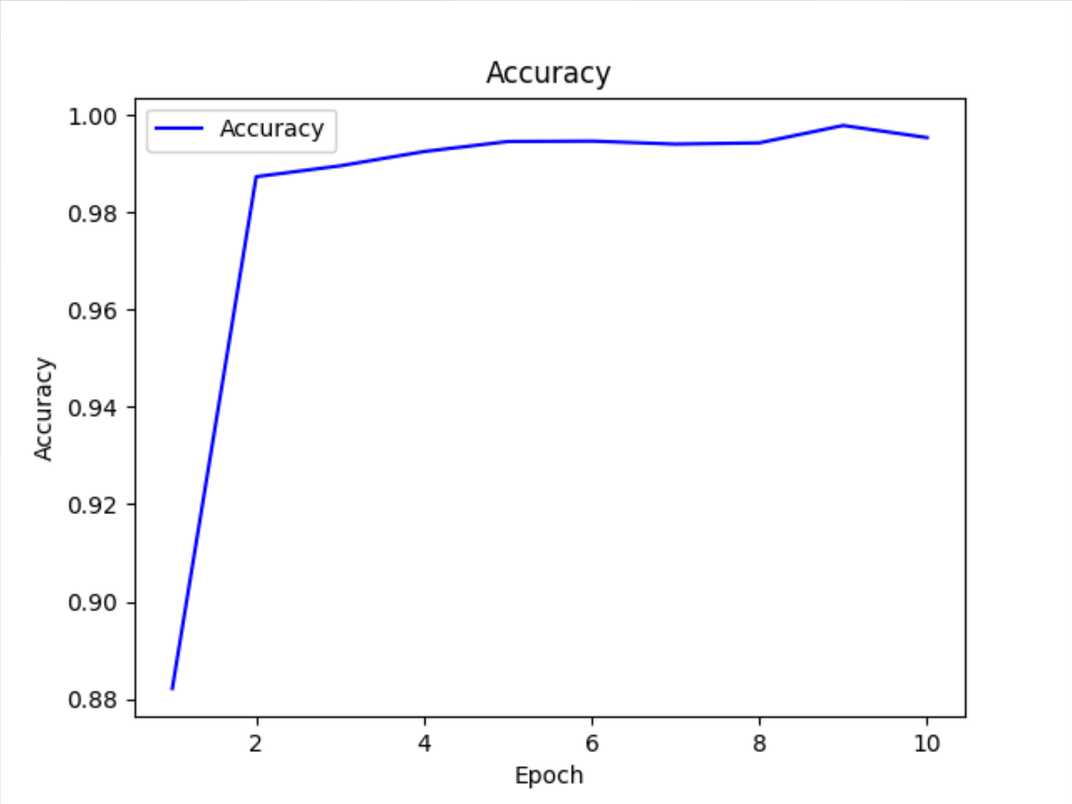




5

Images pré-traitées



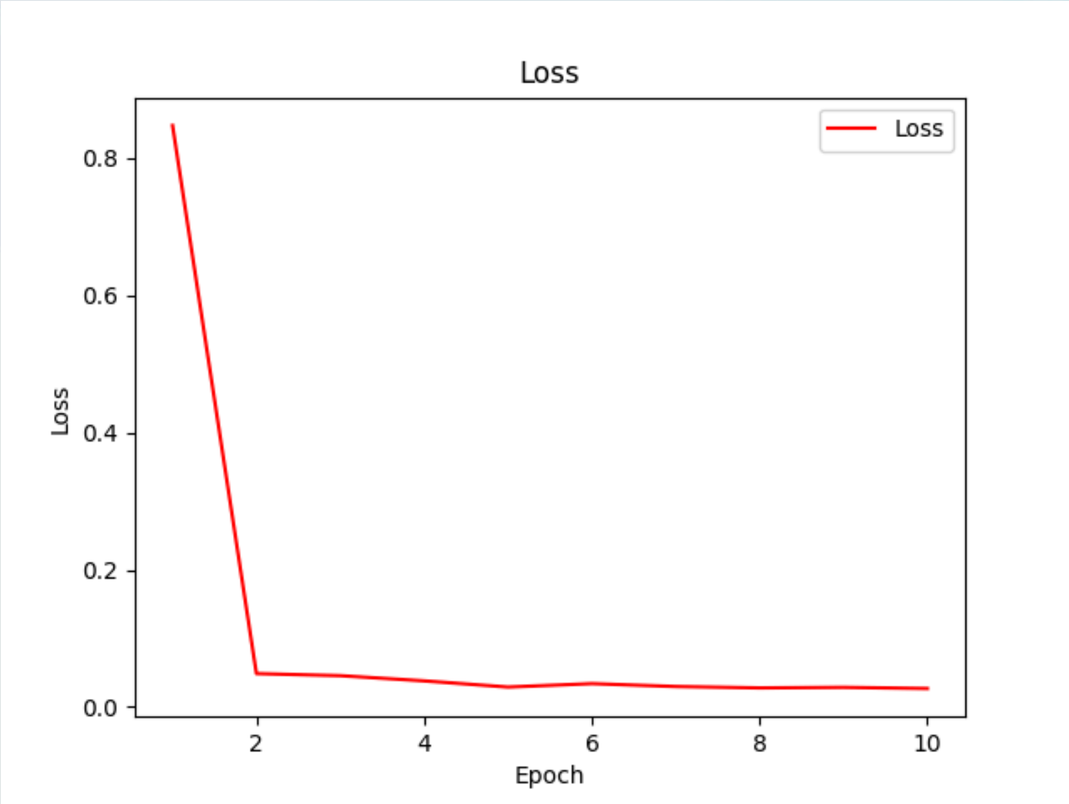


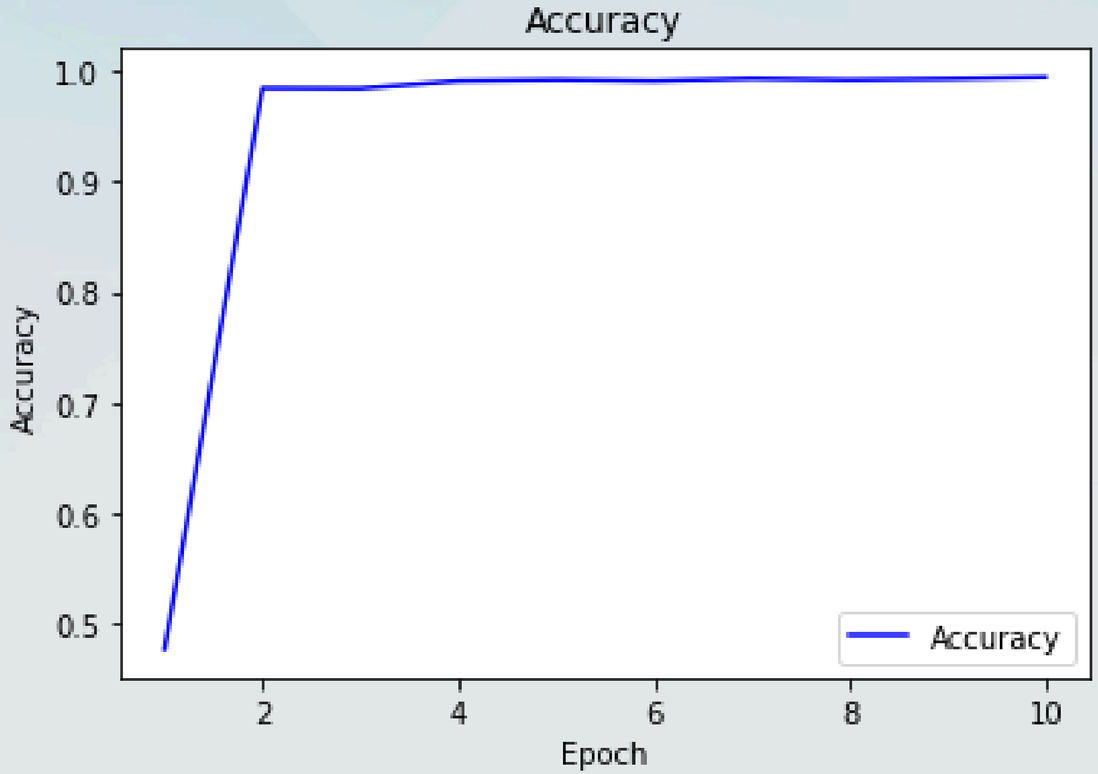
5

Résultats visuels

5

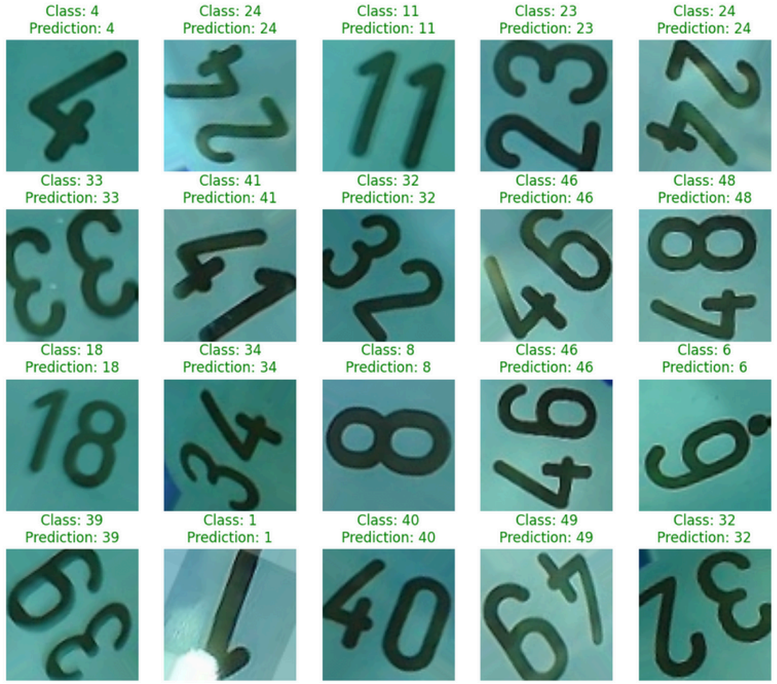
Images en couleur, et rotation à 360°

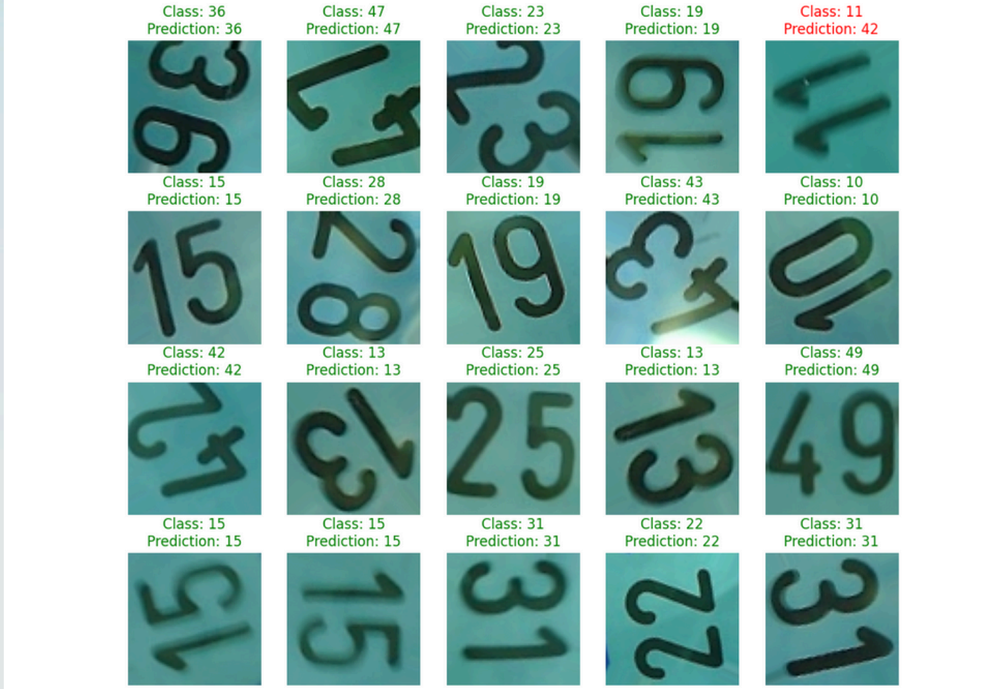




5

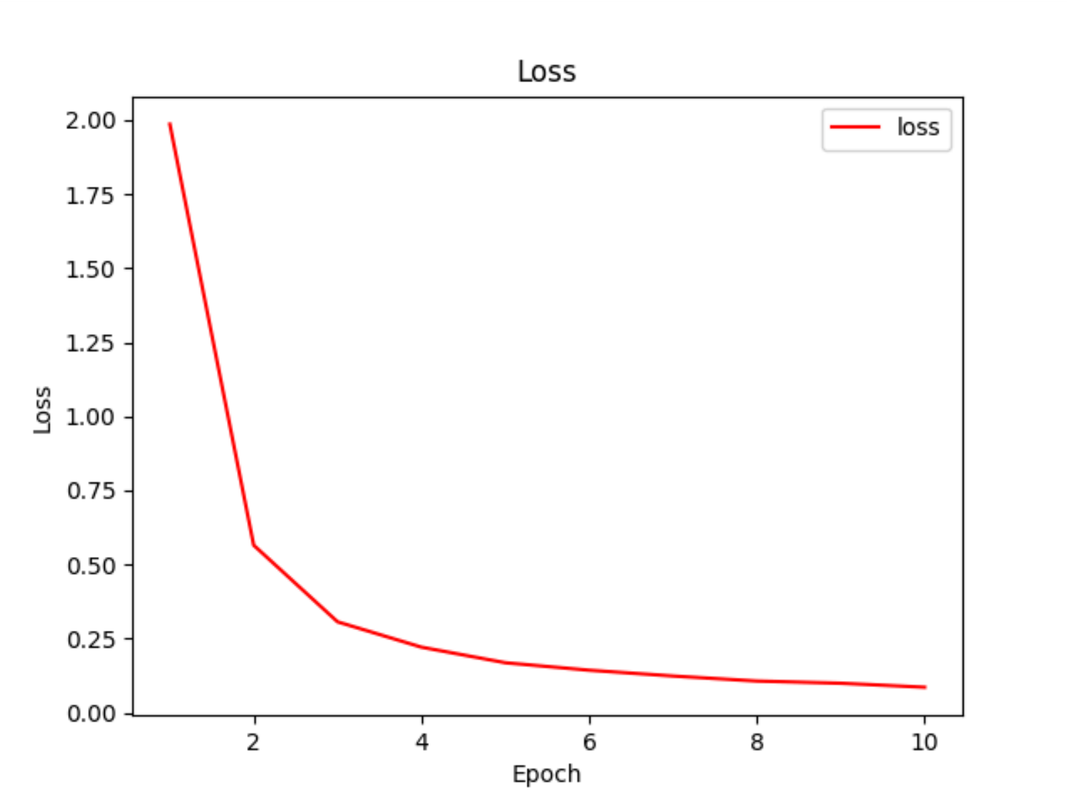
Résultats visuels

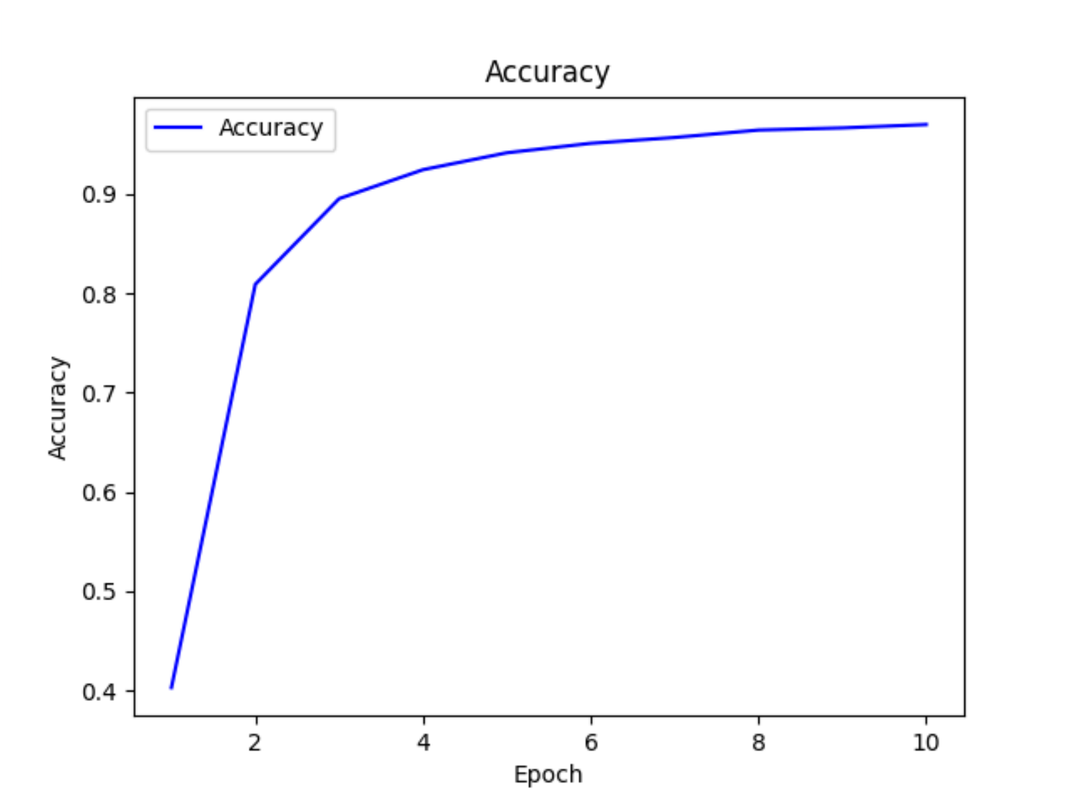




5

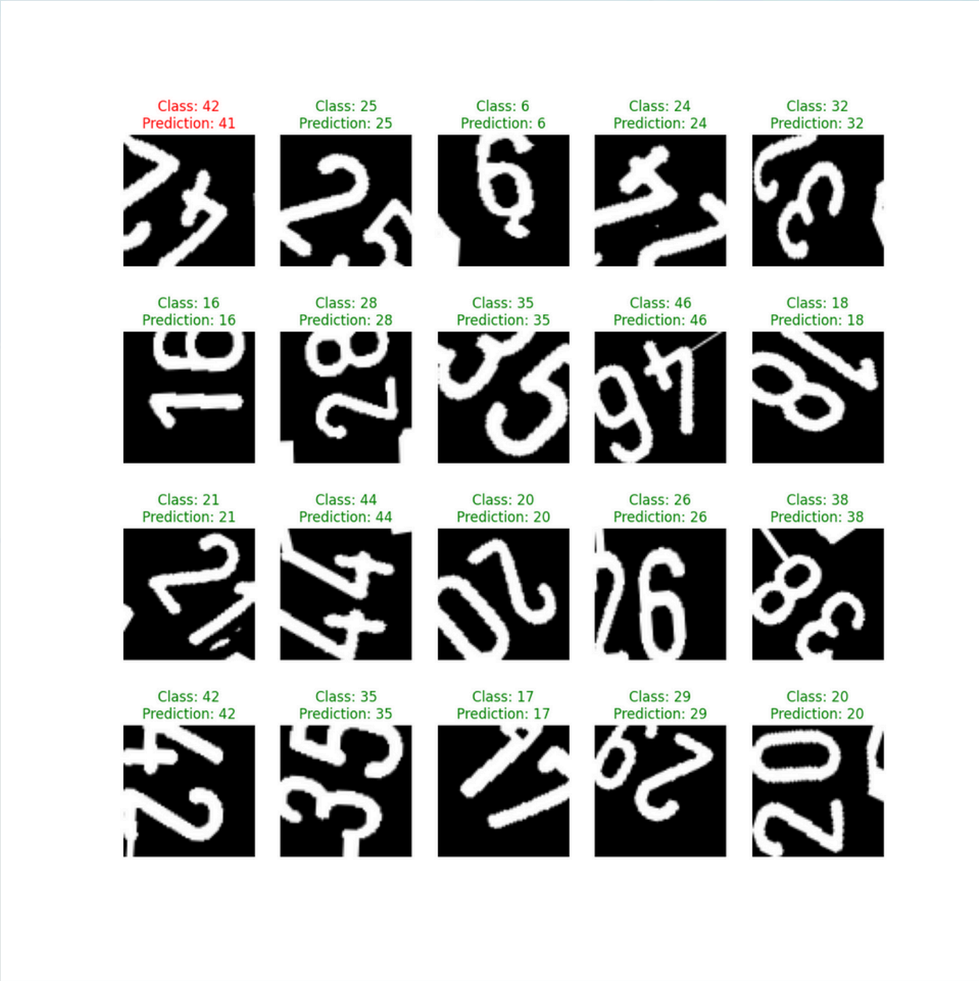
Images en couleur, et rotation à 360° + décalage

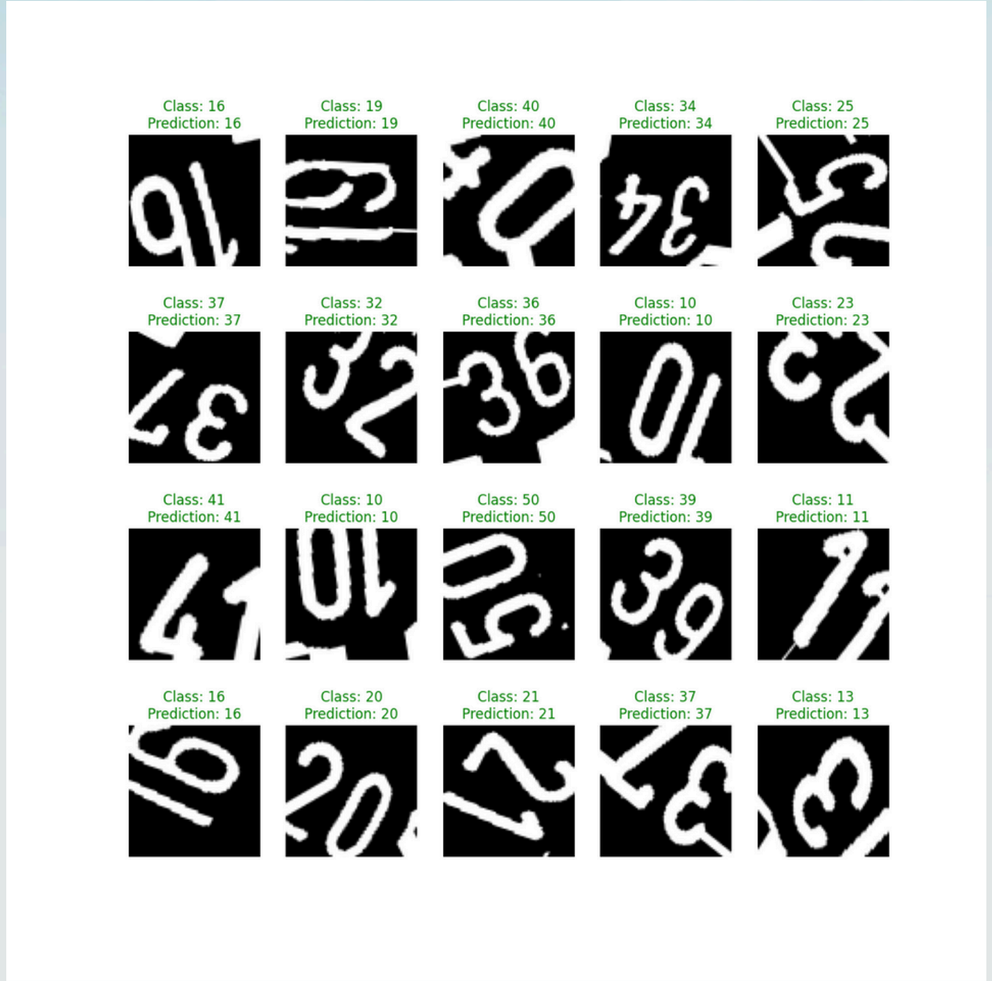




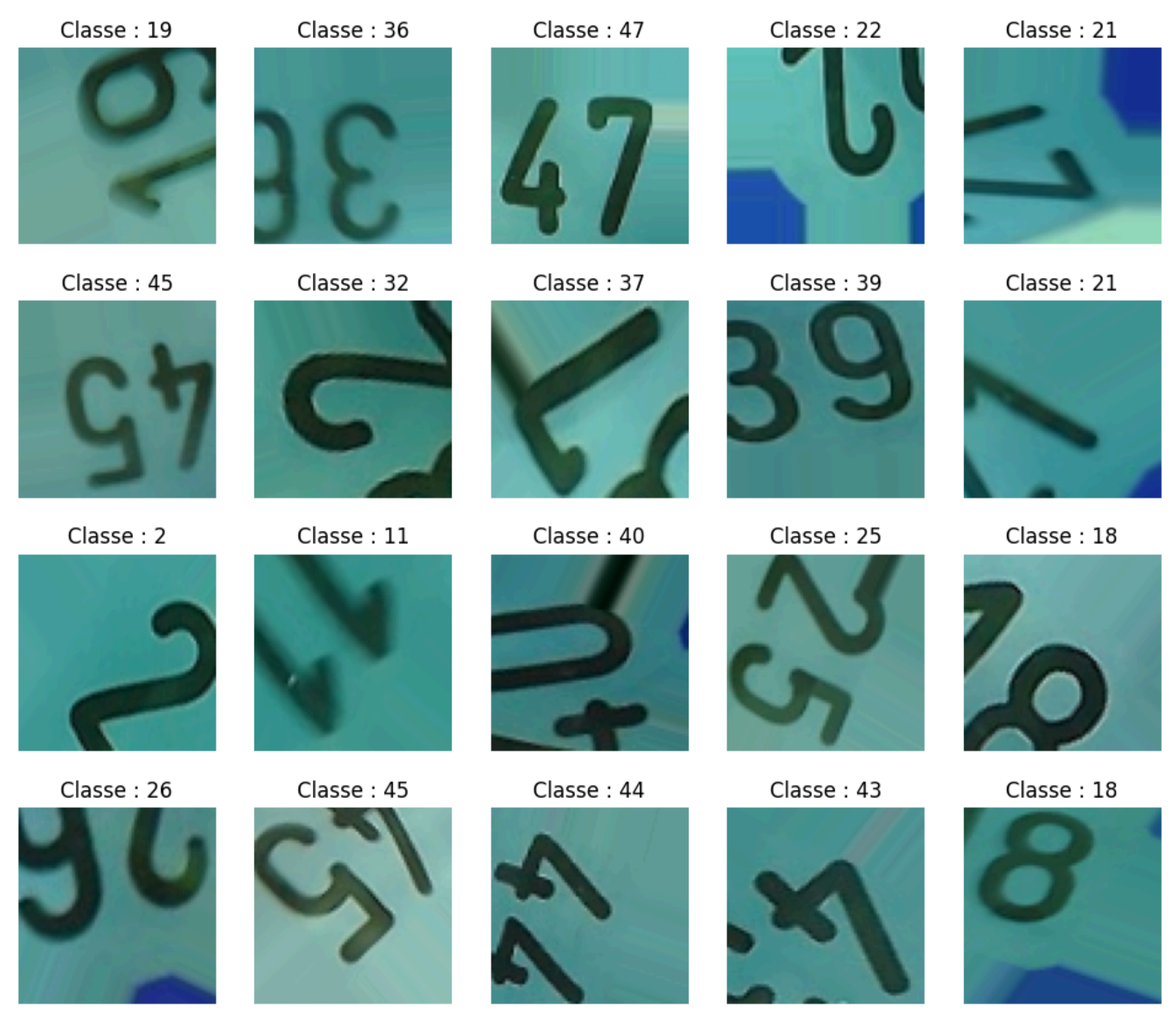
5

Résultats visuels



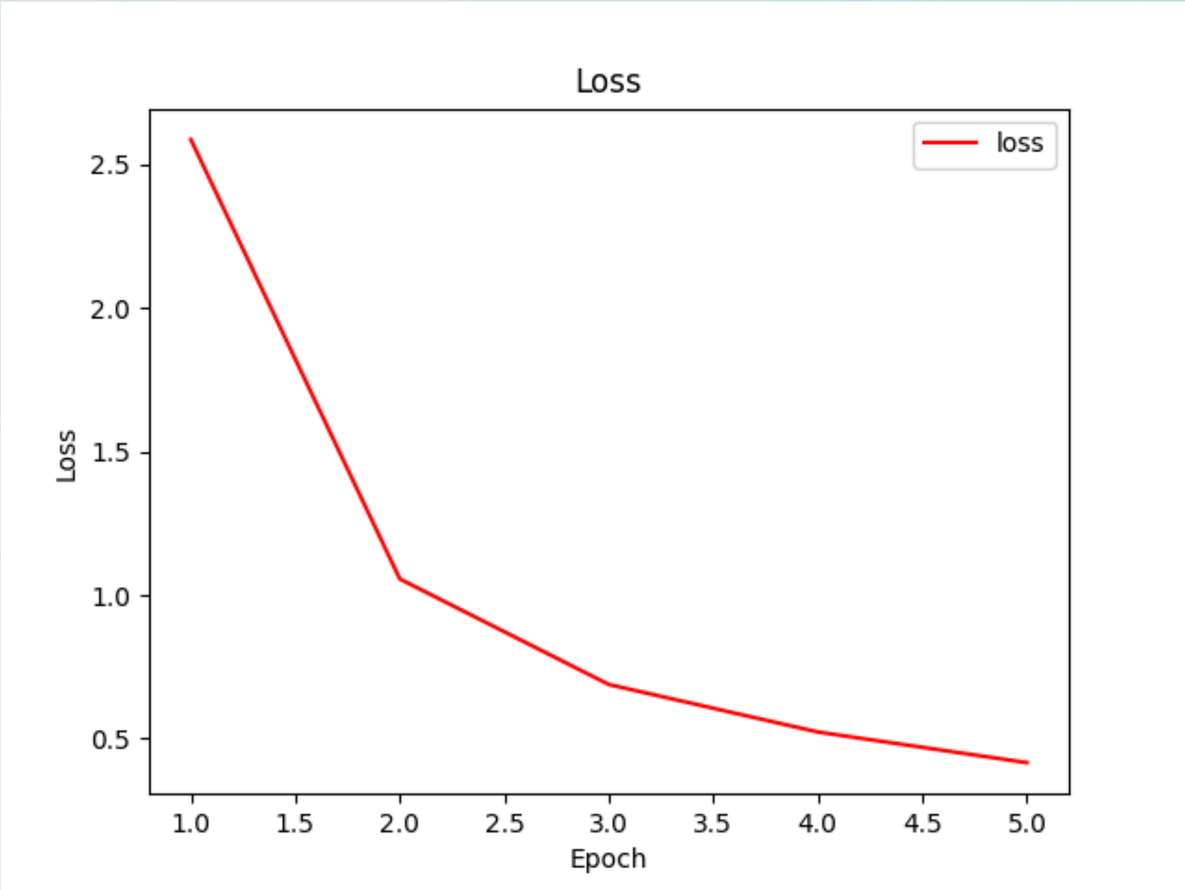


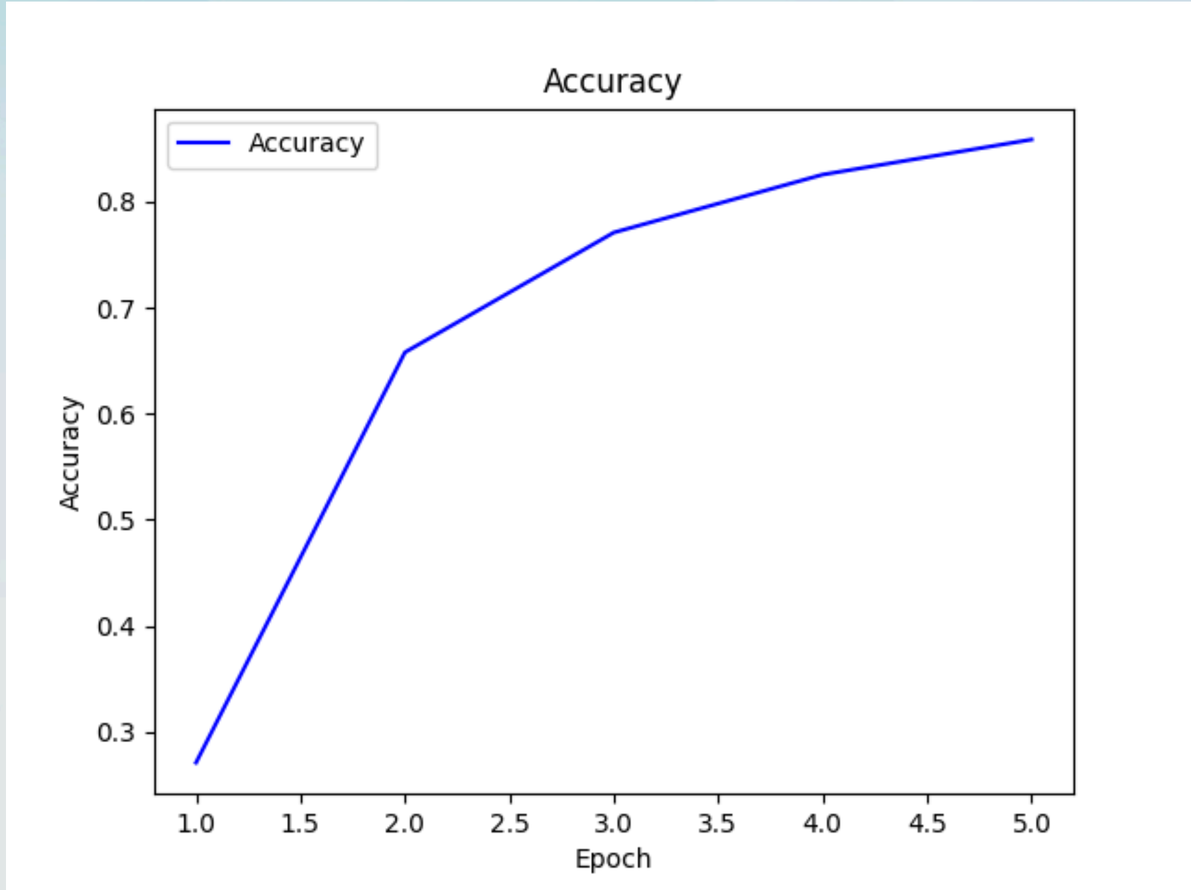
Cas extrême



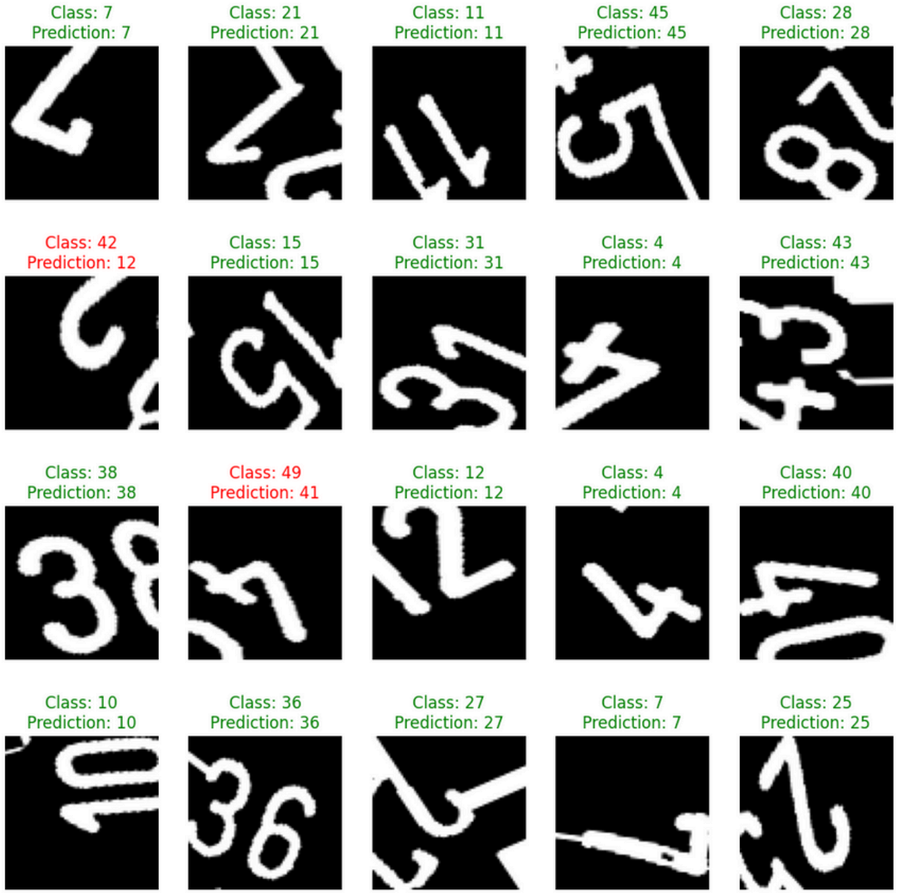
- Augmentation artificielle et modifications trop importantes
- Apprentissage plus difficile

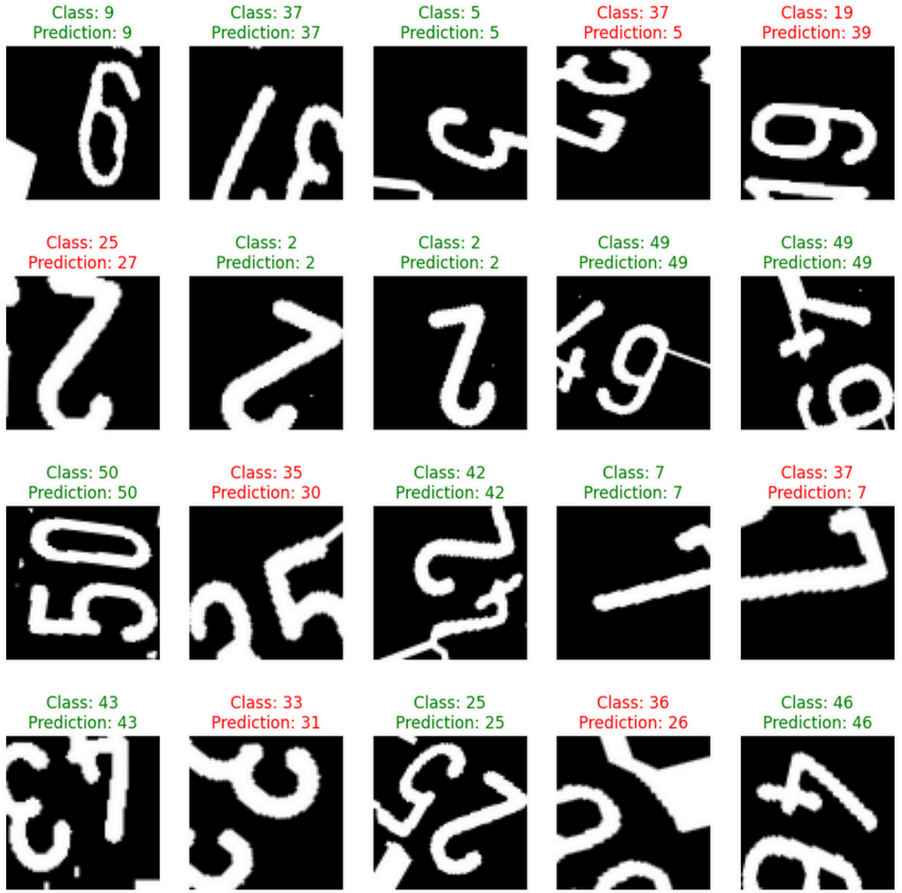
Cas extrême


Loss
loss
2.5
2.0
1.5
1.0
0.5
Loss
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Epoch


Accuracy
Accuracy
0.8
0.7
0.6
0.5
0.4
0.3
Accuracy
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Epoch

Cas extrême


Class: 7
Prediction: 7
Class: 21
Prediction: 21
Class: 11
Prediction: 11
Class: 45
Prediction: 45
Class: 28
Prediction: 28
Class: 42
Prediction: 12
Class: 15
Prediction: 15
Class: 31
Prediction: 31
Class: 4
Prediction: 4
Class: 43
Prediction: 43
Class: 38
Prediction: 38
Class: 49
Prediction: 41
Class: 12
Prediction: 12
Class: 4
Prediction: 4
Class: 40
Prediction: 40
Class: 10
Prediction: 10
Class: 36
Prediction: 36
Class: 27
Prediction: 27
Class: 7
Prediction: 7
Class: 25
Prediction: 25


Class: 9
Prediction: 9
Class: 37
Prediction: 37
Class: 5
Prediction: 5
Class: 37
Prediction: 5
Class: 19
Prediction: 39
Class: 25
Prediction: 27
Class: 2
Prediction: 2
Class: 2
Prediction: 2
Class: 49
Prediction: 49
Class: 49
Prediction: 49
Class: 50
Prediction: 50
Class: 35
Prediction: 30
Class: 42
Prediction: 42
Class: 7
Prediction: 7
Class: 37
Prediction: 7
Class: 43
Prediction: 43
Class: 33
Prediction: 31
Class: 25
Prediction: 25
Class: 36
Prediction: 26
Class: 46
Prediction: 46

5

DÉMONSTRATION

5

Limites

- Trop d’entrainements --> overfitting
- Satisfaction avec les images rognées
- Mais images brutes ? --> Amélioration : ajout d’une fonction de rognage pour les images brutes récupérées

CONCLUSION

- Approfondissement de nos connaissances sur les réseaux de neurones
- Pertinence du pré-traitement des images
- Limites et améliorations

SOURCES

- https://en.wikipedia.org/wiki/Optical_character_recognition
- https://www.tensorflow.org/?hl=en
- https://www.tensorflow.org/datasets/catalog/mnist
- https://www.researchgate.net/figure/Network-structure-of-artificial-neural-network-using-rectified-linear-unit-ReLU_fig4_362851634
- https://www.geeksforgeeks.org/artificial-neural-networks-and-its-applications/

Merci !