28/03/2022

# MCEN4 – STM32 et Deep Learning

Compte Rendu


STMicroelectronics

Lucas GAUVAIN & Romuald CONTAMINE
IUT GEII ANGOULEME
ENSEIGNANT : M. GARCIER

# Table des matières

# Introduction

Le but de ce module complémentaire est d'approfondir les notions de programmation en langage C d'informatique embarquée vu en première année par la mise en œuvre d'un microcontrôleur arm couplé à un écran couleur TFT. A la suite de cela, nous introduirons des notions d'intelligence artificielle et nous aurons l'occasion de programmer et de comprendre le fonctionnement du Deep Learning.

Dans un premier temps nous allons donc être emmené à programmer un microcontrôleur STM32, pour ensuite prendre en main le principe de l'intelligence artificielle afin de faire prédire un nombre à notre programme.

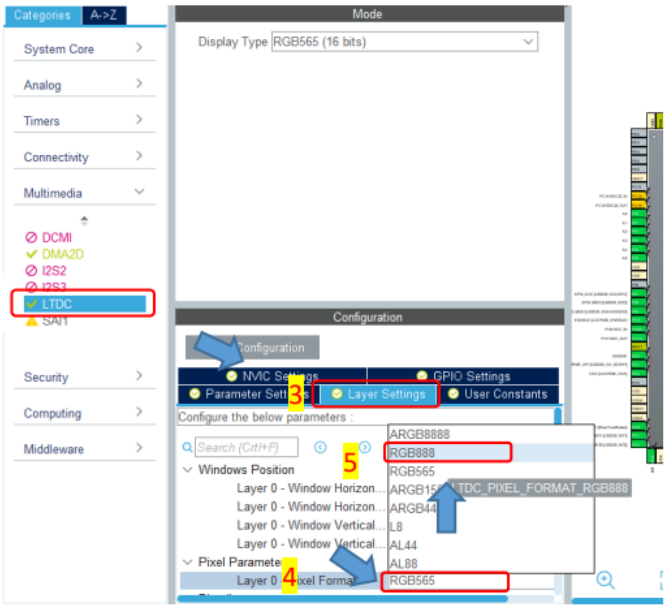
# 1. Programmation du STM32

## 1.1. Configuration du microcontrôleur

Pour commencer, nous devons commencer par récupérer les librairies sur l'ENT contenant les fichiers .c et .h nécessaire pour utiliser le microcontrôleur STM32.

Ensuite, lançons l'assistant de configuration des microcontrôleurs STM32 appelé 'STM32 Cube MX' pour paramétrer les différentes entrées/sorties en fonction du kit de développement utilisé et de nos besoins.

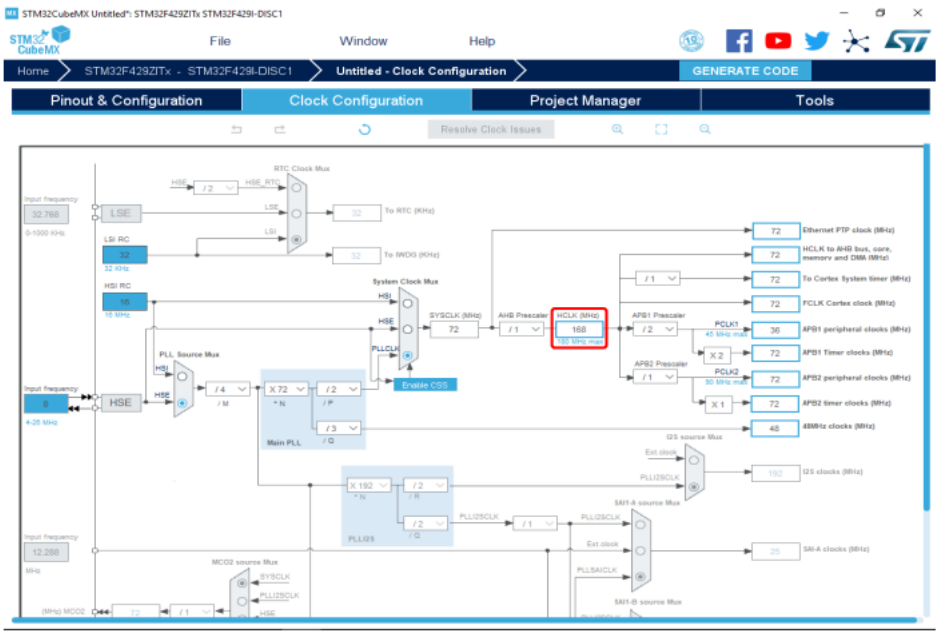
Une fois le bon STM32 sélectionné, changeons quelques paramètres de base de l'afficheur LCD ;




Nous allons modifier le codage des pixels sur l'écran en plaçant le paramètre RGB888 à la pace du RGB565, permettant d'avoir un codage des pixels rouge, vert et bleu sur 8 bits. Ce paramètre permet d'avoir un meilleur rendu mais nécessite un peu plus de temps pour charger chaque pixel.

Ensuite, désactivons le système d'expoitation en temps réels FreeRTOS ainsi que la librairies USB HOST. Cette librairie permet de gérer des equipements comme les claviers et les souris branchés en USB au microcontrôleur mais nous n'en avons pas besoin dans notre projet.

Changeons ensuite la cadence de l'horloge, en la mettant au maximum possible par le STM32 soit 168MHZ.




La dernière étape du processus de configuration de notre STM32 consiste à nommer le fichier dans le menu 'Project Manager'. Nous le nommerons MCEN_LCD_V1.

Une fois fait, nous cliquons sur GENERATE CODE, puis sur Open Project pour lancer automatiquement le nouveau projet dans l'IDE STM32 Cube IDE, qui est l'environnement de programamtion fournit par ST.

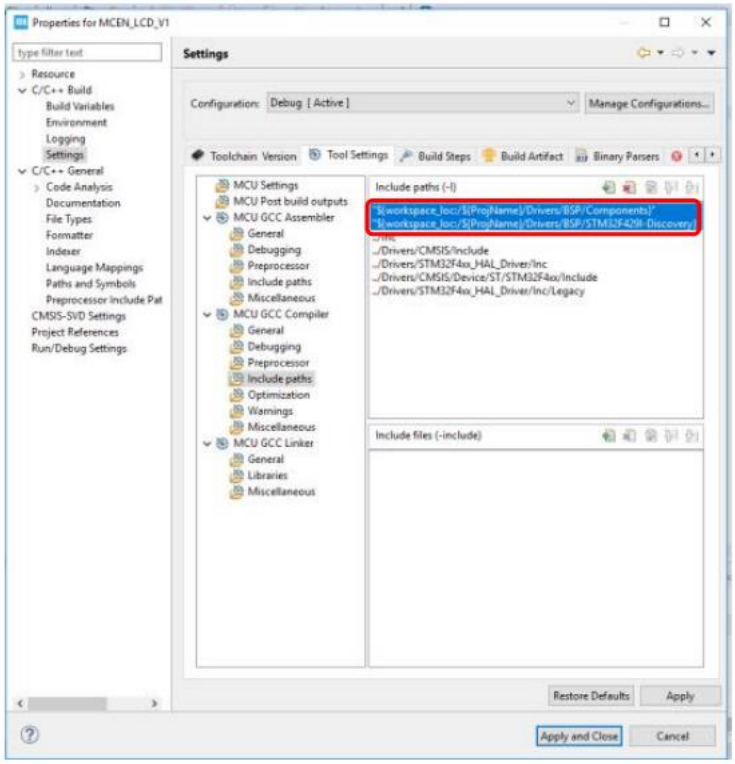
Afin que notre configuration soit complète, il ne faut pas oublier d'inclure les fichiers .c et .h téléchargés au début du projet. Pour cela nous devons :

- Copier Utilities dans le dossier racine de votre projet (MCEN_LCD_V1).
- Copier le dossier BSP dans le dossier Drivers du projet.
- Copier le fichier main.c dans le dossier SRC et écraser le fichier existant.
- Copier le fichier imgSTM32.h dans le dossier Inc.

Dans l'IDE, nous allons définir le workspace dans lequel nous voulons travailler. Nous allons alors pointer sur le dossier MCEN4_STM32 sur le disque I.

Les dossiers de BSP copiés dans le projet ne sont pas à ce stade connus par l'IDE, il est alors nécessaire d'ajouter leurs chemins dans celui-ci afin que le compilateur en tienne compte.

Une fois leurs chemins ajoutés, nous pouvons voir que ceux-ci sont bien intégrés au projet.




## 1.2. Gestion de l'afficheur TFT

Dans le fichier main.c, nous pouvons voir le code suivant :

```
// Configuration de l'écran
BSP_LCD_Init();
BSP_LCD_LayerDefaultInit(LCD_FOREGROUND_LAYER, (LCD_FRAME_BUFFER ));
BSP_LCD_SelectLayer(LCD_FOREGROUND_LAYER);
BSP_LCD_Clear(LCD_COLOR_WHITE);
BSP_LCD_DisplayOn();
```

En appuyant sur la touche F3 après avoir cliquer sur LCD_COLOR_WHITE, nous obtenons une liste de couleur prédéfinis codées en ARGB8888.

Un codage ARGB permet de coder un pixel avec sur 32 bits avec des valeurs allant de 0 à 255 pour les couleurs rouge, verte et bleu (comme le codage RGB) mais en rajoutant une couche alpha qui spécifie l'opacité de chaque pixel. La valeur 0 sur alpha correspond à une image transparente et la valeur 255 correspond à une image entièrement opaque.

Les trois premières couleurs de la liste sont :

```
116 #define LCD_COLOR_BLUE          0xFF0000FF
117 #define LCD_COLOR_GREEN         0xFF00FF00
118 #define LCD_COLOR_RED           0xFFFF0000
```

Couche Alpha à 255 (opaque)

La première couleur est du bleu car le niveau de bleu est au maximum (255) et les autres couleurs sont à 0. C'est donc du bleu pur.

La deuxième couleur est du vert car le niveau de vert est au maximum (255) et les autres couleurs sont à 0. C'est donc du vert pur.

La troisième couleur est du rouge car le niveau de rouge est au maximum (255) et les autres couleurs sont à 0. C'est donc du rouge pur.

Modifions à présent le code pour remplacer le texte « MGA 2022 » par nos noms puis jouons avec la couleur pour afficher un nom en vert et l'autre en rouge. Modifions aussi la couleur du fond d'écran.

Couleur du fond d'écran (noir)

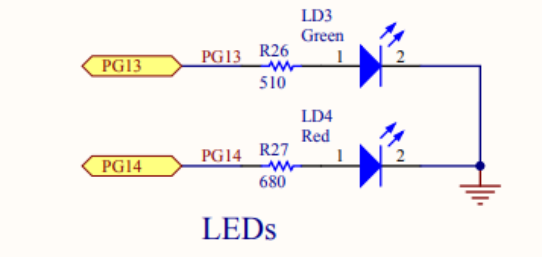
```
// Affichage de l'ecran d'accueil
HAL_LTDC_ProgramLineEvent(&hltdc, 0);
BSP_LCD_SetBackColor(LCD_COLOR_BLACK);
//BSP_LCD_SetTextColor(LCD_COLOR_LIGHTBLUE);
BSP_LCD_SetFont(&Font16);
BSP_LCD_DisplayStringAt(0,LINE(3), (uint8_t *)"Contamine", CENTER_MODE);
BSP_LCD_DisplayStringAt(0,LINE(2), (uint8_t *)"Gauvain", CENTER_MODE);
CopyBuffer((uint32_t *)imgSTM32.data, (uint32_t *)LCD_FRAME_BUFFER, (240-imgSTM32.width)/2 , 80,
```

Affichage de nos noms

Voici l'affichage correspond sur l'écran :

L'objectif maintenant est d'allumer une LED lorsque l'on touche l'écran. Pour commencer créons un nouveau projet qui s'appelle MCEN4_LCD_TouchPad.

On souhaite allumer les LED LD3 et LD4 présente sur la carte. Pour savoir où elles sont raccordées, il faut regarder la schématique de la carte fournit par ST. Le branchement des LED est montré ci-dessous :




On peut constater que la LED LD3 est verte et qu'elle est connectée au PORT G bit 13.

On peut constater que la LED LD4 est rouge et qu'elle est connectée au PORT G bit 14.

On veut que la LED s'allume lorsque l'on appuie sur l'écran et qu'elle soit éteinte le reste du temps. Nous allons alors rajouter cette boucle dans le code.

```
if (Touch_Current)
    {
    HAL_GPIO_WritePin(GPIOG, LD3_Pin, GPIO_PIN_SET);
    }
else
    {
    HAL_GPIO_WritePin(GPIOG, LD3_Pin, GPIO_PIN_RESET);
    }
Touch_Old=Touch_Current;
```

Nous devons à présent afficher une image sur l'écran, qui servira d'arrière-plan avant d'afficher les coordonnées de la souris.

On va alors créer une image sous le logiciel Inkscape, en commençant par définir la taille de 240x320px, correspondant à la taille de l'écran sur la carte. Sur l'image, nous inscrirons alors les textes « X = » et « Y = » afin d'anticiper l'affichage des coordonnées de la souris. Une fois l'image dessinée, nous devons l'exporter en format PNG et récupérer le fichier .h correspondant à l'aide de Lcd-Image-Converter disponible sur l'ENT.

Pour implanter l'image dans le microcontrôleur, nous devons copier-coller le fichier .h que nous venons de générer dans I:/MCEN4_STM32/MCEN4_LCD_TouchPad/Inc.

Les fichiers avec l'extension .h sont des fichiers d'en tête contenant les prototypes des fonctions, les types personnalisés, les classes, les structures…

Pour que notre fichier .h soit compatible avec le programme, il faut enlever la structure qui est en commentaire.

- Nous allons donc afficher notre image à la place de l'image précédente : Nous devons rajouter ces deux lignes de code pour charger l'image. (Nous avons enregistrer l'image sous le nom image.PNG).

```
#include "image.h"

CopyBuffer((uint32_t *)image.data, (uint32_t *)LCD_FRAME_BUFFER, (240-image.width)/2 , 0, image.width, image.height);
```

- Ensuite nous voulons afficher un message d'accueil pendant 2 secondes avant l'affichage de l'image. Les lignes de codes ci-dessous permettent d'afficher le message « Hello World » pendant deux secondes. Les délais sont exprimés en millisecondes donc 2 secondes correspond à un délai de 2000.

```
BSP_LCD_DisplayStringAt(0,LINE(3), (uint8_t *)"Hello World", CENTER_MODE);
HAL_Delay(2000);
```

- Maintenant, nous voulons afficher les coordonnées du doigts sur l'écran. Pour cela, nous devrons utiliser la fonction itoa(), qui permet de convertir un entier en chaine de caractère. Itoa() appartient à la bibliothèque iostream. Il est donc nécessaire d'appeler cette bibliothèque au début du code avec #include <iostream>.

Pour réaliser la conversion d'un entier vers un string, itoa() à besoin d'un buffer. En langage C, un buffer est une mémoire tampon sous forme d'un tableau de valeurs, dont chaque case de ce tableau va contenir un caractère. Dans notre cas, chaque case contiendra un chiffre de coordonnées. La dernière case du tableau est un 0 et permet de signifier la fin du tableau. La valeur maximale d'une coordonnée étant de 320, il faut donc que le buffer est une taille minimale de 4 cases pour fonctionner correctement.

Par exemple, si l'on veut convertir 123, le tableau sera le suivant :

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 0 |

Avant d'afficher la position du doigt en temps réel, nous allons afficher à la place des valeurs en dur pour vérifier que l'affichage et que la fonction itoa() fonctionne correctement.

Le code pour afficher les valeurs en dur est le suivant :

```
char buffer[4];
int coordx = 168;
int coordy = 246;
```

Coordonnées X et Y en dur.

```
BSP_LCD_DisplayStringAt(0,LINE(6), (uint8_t *)itoa(coordx, buffer, 10), RIGHT_MODE);
BSP_LCD_DisplayStringAt(0,LINE(14), (uint8_t *)itoa(coordy, buffer, 10), RIGHT_MODE);
```

10 permet de spécifier un affichage en base 10.

Pour afficher la position en temps réel du doigt sur l'écran, nous devons créer en plus des variables des coordonnées instantanée, deux variables contenant la position d'avant. Ces variables sont essentielles pour mettre a jour les nouvelles valeurs, sans elles nous n'aurions que la première position du doigt et l'actualisation ne se ferait donc pas.

```
int main(void)
{
    /* USER CODE BEGIN 1 */
    char buffer[4];
    int coordx;
    int coordy;
    int coordx_old =0;
    int coordy_old=0;
```

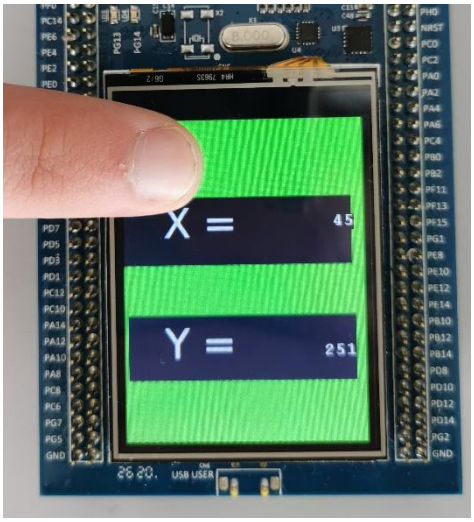
coordx_old et coordy_old contiennent respectivement les anciennes valeurs de la position du doigt.

```
/* USER CODE BEGIN 3 */
HAL_Delay(1);
BSP_TS_GetState(&TouchScreen);
Touch_Current=TouchScreen.TouchDetected;

if (Touch_Current!=Touch_Old)
{
    coordx=TouchScreen.X;
    coordy=TouchScreen.Y;

    if (coordx_old != coordx)
    {
        BSP_LCD_SetTextColor(LCD_COLOR_BLACK);
        BSP_LCD_DisplayStringAt(0,LINE(6), (uint8_t *)itoa(coordx_old, buffer, 10), RIGHT_MODE);
        BSP_LCD_SetTextColor(LCD_COLOR_WHITE);
        BSP_LCD_DisplayStringAt(0,LINE(6), (uint8_t *)itoa(coordx, buffer, 10), RIGHT_MODE);
    }

        coordx_old = coordx;

    if (coordy_old != coordy)
    {
        BSP_LCD_SetTextColor(LCD_COLOR_BLACK);
        BSP_LCD_DisplayStringAt(0,LINE(14), (uint8_t *)itoa(coordy_old, buffer, 10), RIGHT_MODE);
        BSP_LCD_SetTextColor(LCD_COLOR_WHITE);
        BSP_LCD_DisplayStringAt(0,LINE(14), (uint8_t *)itoa(coordy, buffer, 10), RIGHT_MODE);
    }
        coordy_old = coordy;

    if (Touch_Current)
        {
        HAL_GPIO_WritePin(GPIOG, LD3_Pin, GPIO_PIN_SET);
        }
    else
        {
        HAL_GPIO_WritePin(GPIOG, LD3_Pin, GPIO_PIN_RESET);
        }
    Touch_Old=Touch_Current;
}

}
```

Coordonnées en temps réels

Actualisation des coordonnées en X

Actualisation des coordonnées en Y

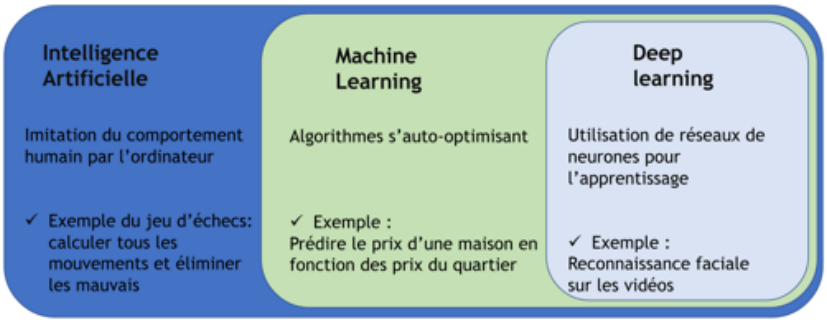



On constate que la position se rafraichit à chaque fois le doigt bouge sur l'écran donc le programme fonctionne correctement.

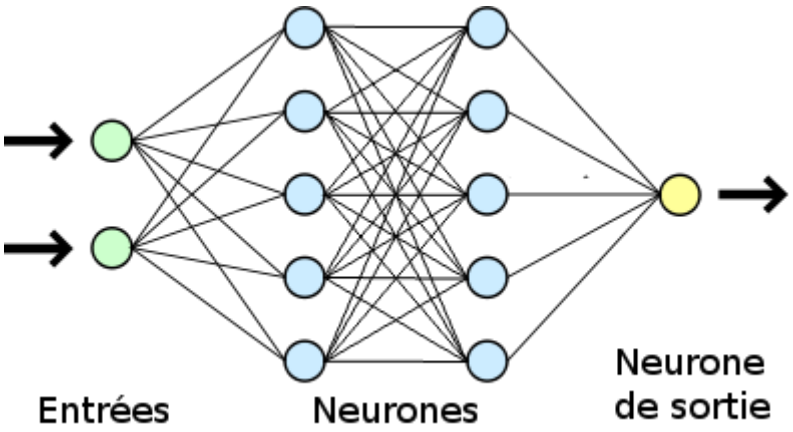
# 2. Intelligence Artificielle

## 2.1. Quelques recherches

L'intelligence artificielle permet à un ordinateur d'imiter le comportement humain. L'intelligence artificielle inclue un domaine appelé le Machine Learning (qui sont des algorithmes s'auto-optimisant). Ce Machine Learning inclue le Deep Learning, aussi appelé en français 'Apprentissage profond', qui utilise des réseaux de neurones pour l'apprentissage, principalement utilisé pour de la reconnaissance d'images ou de la reconnaissance faciale. On peut visualiser le Deep Learning de la manière suivante :




Dans le Deep Learning, la machine est capable d'apprendre seule. Ce modèle de fonctionnement se base sur celui du cerveau humain, par un réseau de neurones artificiels, qui se nomment des neurones formels. Les neurones formels possèdent plusieurs entrées (que l'on peut assimiler aux dendrites en biologie) et une seule sortie (que l'on peut assimiler à un cône). Un neurone formel est donc une équation. Ce réseau de neurones artificiels peut être constituer de plusieurs centaines de couches de neurones qui vont recevoir des informations interprétées par les couches précédentes. Le neurone apprendra en optimisant une fonction de perte, c'est-à-dire en cherchant les valeurs qui minimise la fonction. C'est en mathématique une minimisation de fonction. C'est ce que l'on appelle l'algorithme de descente de gradient. On peut visualiser un réseau de neurones de la manière suivante :



Le domaine de l'intelligence artificielle comporte du vocabulaire important :

Une prédiction en Deep Learning correspond à la sortie du réseau de neurones et donc à ce que l'algorithme pense. Plus les prédictions de l'algorithme sont juste pour celui-ci est fiable.

La classification est le processus d'identification de la catégorie ou de l'étiquette de classe de la nouvelle observation à laquelle elle appartient.

La fonction de convolution est utilisée pour exprimer comment la forme d'une fonction est modifiée par l'autre.

Pour notre travail, nous aurons besoin d'utiliser la base MNIST.

La base de données MNIST pour Modified ou Mixed National Institute of Standards and Technology, est une base de données de chiffres écrits à la main. Elle regroupe 60000 images d'apprentissage et 10000 images de test. Ce sont des images en noir et blanc, normalisées de 28 pixels de côté.

## 2.2. Afficher un chiffre manuscrit de la base MNIST

Dans le dossier MCEN4_STM32, nous allons créer un dossier MNIST qui contiendra toute notre partie sur l'intelligence artificielle. Nous allons ensuite récupérer le fichier python mnist_plot.py sur l'ENT. Ouvrons le dans Visual Studio Code et lançons-le à l'aide d'Anaconda Prompt.

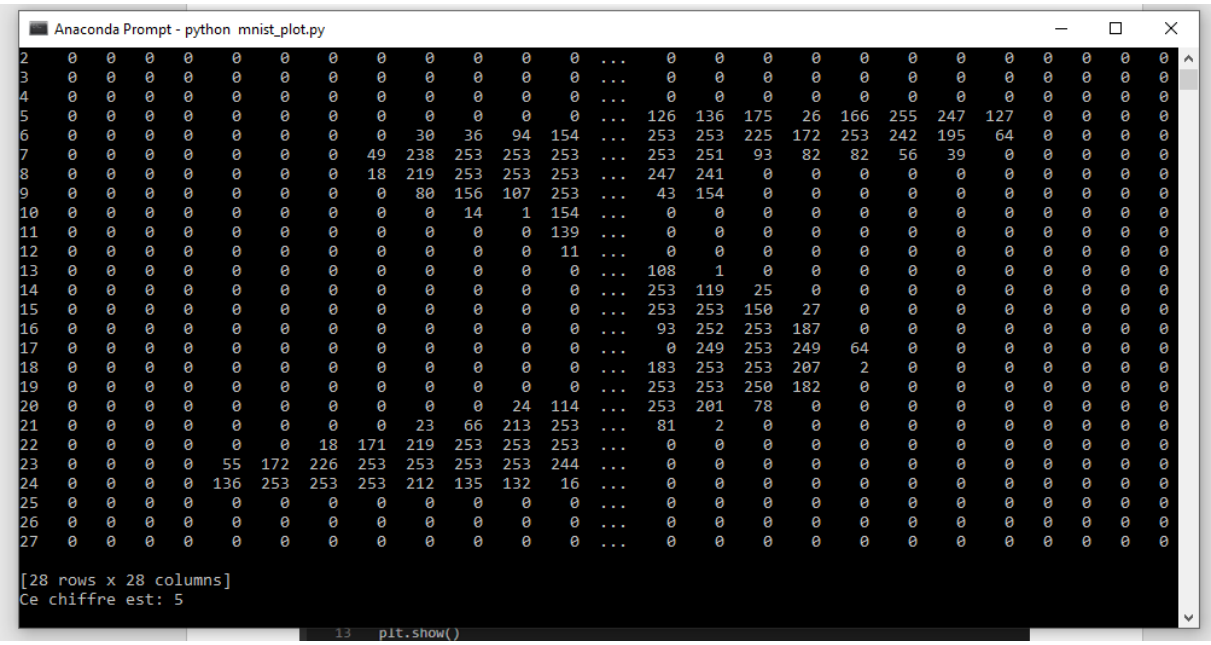
Le code est le suivant :

mnist_plot.py

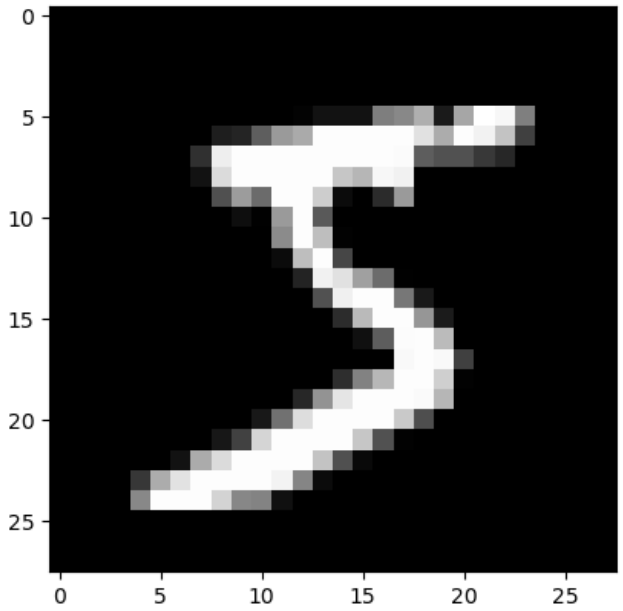
I: > MCEN4_STM32 > MNIST > mnist_plot.py

```python
# Plot mnist instances
from keras.datasets import mnist
from pandas import DataFrame
import matplotlib.pyplot as plt
# load (downloaded if needed) the MNIST dataset
(X_train, y_train), (X_test, y_test) = mnist.load_data()
# plot 1 image as gray scale
plt.subplot(111)
plt.imshow(X_train[0], cmap=plt.get_cmap('gray'))
print(DataFrame(X_train[0]))
print("Ce chiffre est:",y_train[0])
# show the plot
plt.show()
```

Dans la console, nous pouvons visualiser ce tableau après l'exécution.




Une autre fenêtre s'ouvre avec l'image ci-dessous :




Ce code permet d'afficher une image de la bibliothèque MNIST. En l'occurrence, l'image est ici celle d'un chiffre 5.

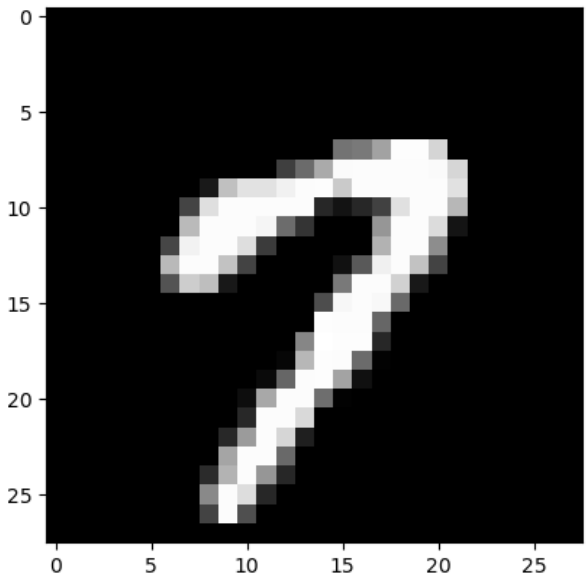
Dans le code, X_train indique l'image a utilisé dans la bibliothèque MNIST. On indique la position de l'image entre les crochets. Ici, nous prenons la première image qui se trouve à la position 0.

Y_train est la prédiction faite par le programme à travers le réseau de neurones.

Maintenant, pour afficher le chiffre 7, il faut trouver la position d'une image contenant le chiffre 7 dans la bibliothèque MNIST. Il faut donc changer la position du pointeur entre les crochets de X_train. Après plusieurs essais, les emplacements 15 et 42 contiennent le chiffre 7.

```
1   # Plot mnist instances
2   from keras.datasets import mnist
3   from pandas import DataFrame
4   import matplotlib.pyplot as plt
5   # load (downloaded if needed) the MNIST dataset
6   (X_train, y_train), (X_test, y_test) = mnist.load_data()
7   # plot 1 image as gray scale
8   plt.subplot(111)
9   plt.imshow(X_train[15], cmap=plt.get_cmap('gray'))
10  print(DataFrame(X_train[15]))
11  print("Ce chiffre est:",y_train[15])
12  # show the plot
13  plt.show()
```

En modifiant le code en pointant les vers la 8ème image de la bibliothèque (position 7), on obtient l'image suivante :



## 2.3. Création d'une fenêtre de dessin en python

Dans cette partie on cherche à créer une petite fenêtre dans laquelle une zone blanche servira de « feuille virtuelle » pour que l'on puisse y inscrire à la souris un chiffre.

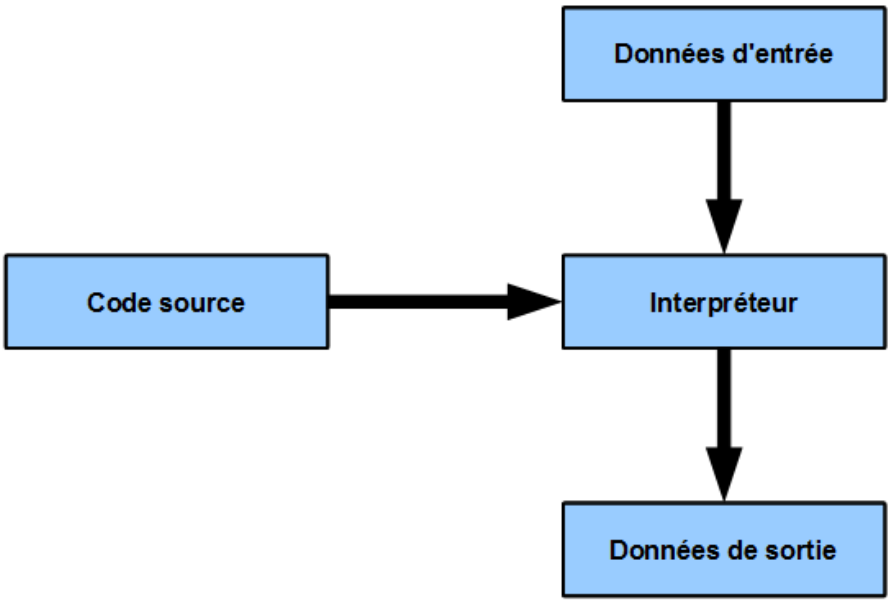
Il nous est demander de faire :

- Une zone blanche de dessin créée grâce à un widget Canvas de 200px par 200px
- Une zone de texte indiquant les coordonnées de la souris créée grâce à un widget Label
- Les coordonnées de la souris seront affichées et mis à jour en permanence, c'est-à-dire quelque-soit l'état des boutons de la souris.
- Le dessin se fera lorsque le clic gauche de la souris est enfoncé.
- Le clic droit de la souris permettra d'enregistrer automatiquement le dessin du Canvas dans une image de 28px par 28px au format png

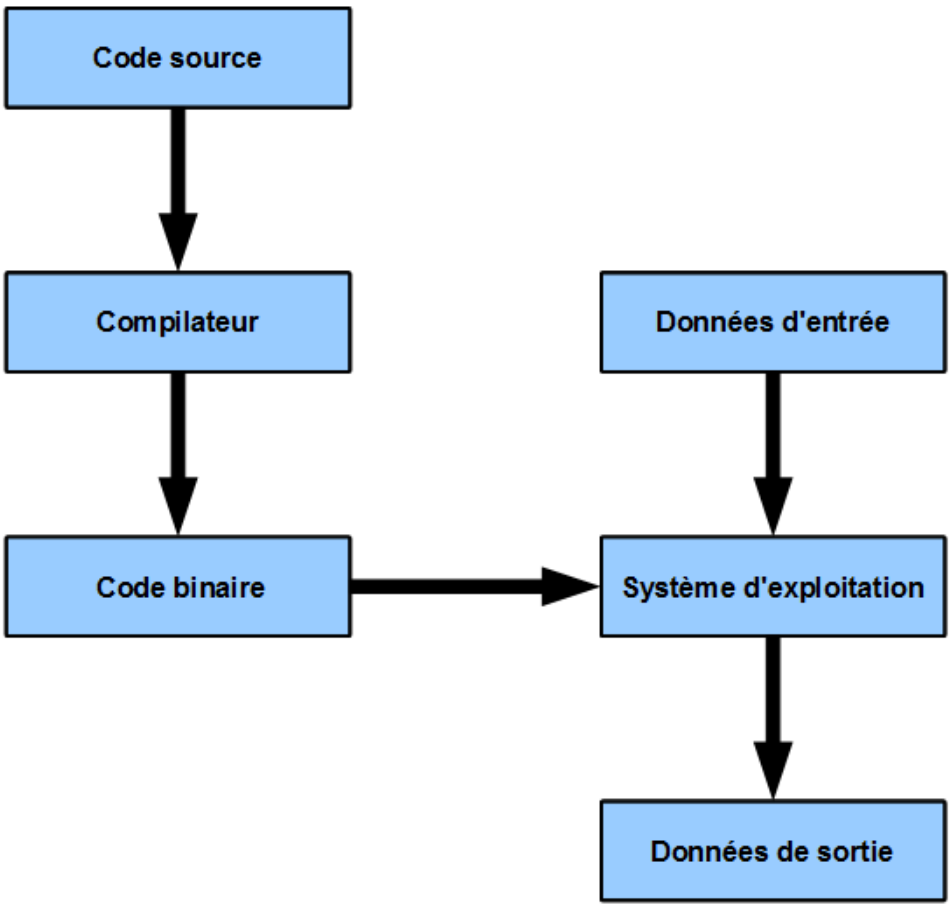
Pour nous aider, nous avons à notre disposition un fichier python draw_etu.py que nous devons modifier pour atteindre le résultat attendu.

Python est un langage interprété, c'est-à-dire que le code source est interprété, par un logiciel qu'on appelle interpréteur. Celui-ci va utiliser le code source et les données d'entrée pour calculer les données de sortie. L'interprétation du code source est un processus « pas à pas ». L'interpréteur va exécuter les lignes du code une par une, en décidant à chaque étape ce qu'il va faire ensuite.

On peut visualiser un langage interprété avec le schéma suivant :



A la différence d'un langage compilés, le code source est tout d'abord compilé, par un logiciel qu'on appelle compilateur en un code binaire qu'un humain ne peut pas lire mais qui est très facile à lire pour un ordinateur. C'est alors directement le système d'exploitation qui va utiliser le code binaire et les données d'entrée pour calculer les données de sortie. On peut visualiser les langages compilés avec le schéma suivant :




Dans un langage interprété, le même code source pourra marcher directement sur tout ordinateur. Avec un langage compilé, il faudra (en général) tout recompiler à chaque fois ce qui pose parfois des soucis.

Dans un langage compilé, le programme est directement exécuté sur l'ordinateur, donc il sera en général plus rapide que le même programme dans un langage interprété.

Tkinter est un module intégré à Python pour développer des applications graphiques. Ce module se base sur la bibliothèque graphique Tcl/Tk.

Pour créer un logiciel graphique, nous devons ajouter dans une fenêtre des éléments graphiques que l'on nomme widget. Ces widgets peuvent aussi bien être une liste déroulante que du texte.

Pour lier un évènement à un widget, nous devons utiliser la méthode bind.

Le code ci-dessous permet de réaliser la fenêtre demandée pour afficher les coordonnées de la souris, dessiner un chiffre et enregistrer l'image en format PNG.

```
3
4   from tkinter import *
5   from PIL import Image, ImageOps                              #ImageOps for grayscale
6
7   #Variables globales
8   old_x, old_y = 0, 0
9
10  #This function displays the mouse coordinates in Label widget
11  def MouseCoord(event):
12      global cadre
13      global old_x, old_y                                          #Allows to modify a global variable
14      X = event.x
15      Y = event.y
16      coord.set("x= "+ str(X)+ " ; y= "+ str(Y))
17
18      old_x = event.x
19      old_y = event.y
20
21
22  def Dessin(event):
23      global old_x, old_y
24      x = event.x
25      y = event.y
26      cadre.create_line(old_x, old_y ,x, y, width= 10)
27
28      old_x = event.x
29      old_y = event.y
30
31  def Save_as_png(event):
32      global cadre         #Allows to modify a global variable
33      # MGA: bellows are listed the different steps to store your canvas drawing into a 28px x 28 px grayscale png picture
34      # 1 - save postscipt image
35      imgps_save = cadre.postscript(file='image.ps')
36
37      # 2 - use PIL to convert to PNG
38      image = Image.open('image.ps')
39
40      # 3 - use ImageOps to invert black and white
41      inverted_image = ImageOps.invert(image)
42
43      # 4 - use PIL to convert in grayscale
44      image_grayscale = ImageOps.grayscale(inverted_image)
45
46      # 5 - use PIL to reduce picture to 28px x 28 px
47      image_reduce = image_grayscale.resize((28,28), Image.ANTIALIAS)
48
49      # 6 - use PIL to save picture as png
50      image_reduce.save('image_finale.png')
51
52
53  fenetre1=Tk()    #Swinen's book p76
54
55
56
57  cadre=Canvas(fenetre1, width=200, height=200, bg='#FFFFFF')     #p89
58  cadre.bind("<Motion>", MouseCoord)
59  cadre.bind("<B1-Motion>", Dessin)
60  cadre.bind("<Button-3>", Save_as_png)                           # call the save function on left click
61  cadre.pack()
62
63
64  # create here your Label widget
65  coord = StringVar()
66  label = Label(fenetre1, textvariable = coord , bd = 20)
67  coord.set("x= "+ X+ " ; y= "+ Y)
68  label.pack()
69
70  fenetre1.mainloop()
```

La fonction MouseCoord(event) permet de rafraichir les coordonnées de la souris en temps réels. Pour cela, on relie cette fonction avec la commande <Motion>.

La fonction dessin permet de dessiner avec la souris et elle est relié au clic gauche enfoncé avec la commande <B1-Motion>.

La fonction Save_as_png permet d'enregistrer une l'image du canvas en png et elle est reliée au clic droit de la souris avec la commande <Button-3>.

On obtient ainsi la fenêtre suivante ou l'on peut dessiner ce que l'on veut.


On constate aussi que l'enregistrement de l'image en png se fait aussi dans le dossier. L'image à une taille de 28x28px.

| Nom | Modifié le | Type | Taille |
|---|---|---|---|
| .vs | 28/03/2022 11:22 | Dossier de fichiers | |
| draw_and_interface | 28/03/2022 11:22 | Python source file | 4 Ko |
| draw_etu | 14/03/2022 15:24 | Python source file | 3 Ko |
| image.ps | 28/03/2022 11:28 | Fichier PS | 25 Ko |
| image_finale | 25/03/2022 12:14 | Image PNG | 1 Ko |
| mnist_plot | 11/03/2022 15:22 | Python source file | 1 Ko |
| mnist_rnn_inference | 25/03/2022 11:42 | Python source file | 2 Ko |
| mnist_rnn_train | 14/03/2022 16:01 | Python source file | 2 Ko |
| model_RNN_handwritting.h5 | 14/03/2022 17:17 | Fichier H5 | 7 326 Ko |

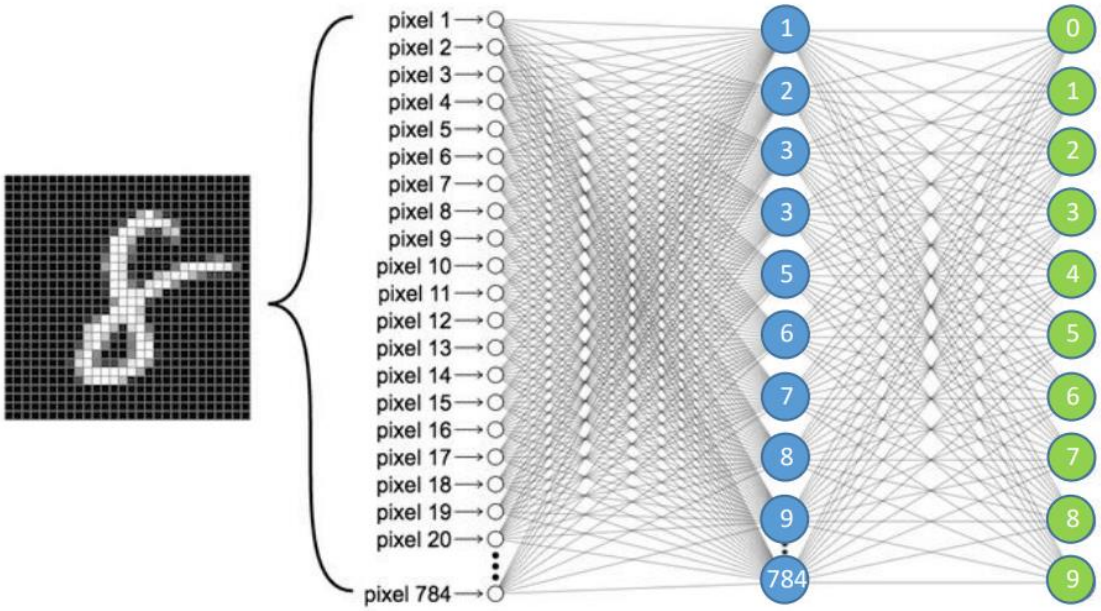
## 2.4. Entrainer un réseau neuronal simple

Nous avons déjà vu que la base MNIST contient des images de chiffres compris entre 0 et 9 codés en niveau de gris sur 8 bits de 28px par 28 px.

La construction d'un réseau de neurone le plus simple possible sera formé de 2 couches (layers) :

- La couche d'entrées contiendra autant d'entrées que de pixels soit 28x28=784.
- La couche de sortie contiendra autant de sortie que de classes soit 10 (une pour chaque chiffre).

En voici la configuration :




On peut voir sur l'image ci-dessus que chaque neurone d'une couche est connecté à l'ensemble des neurones de l'autre couche.

On dit que les couches sont « fully connected » ou « Dense ». Comme les couches sont associée les unes après les autres, on parle de modèle séquentiel.

Téléchargeons sur l'ENT le fichier python mnist_rnn_train.py :

```python
from keras.datasets import mnist
from keras.models import load_model
from pandas import DataFrame
from keras.models import Sequential
from keras.layers import Dense
from keras.utils import np_utils

import numpy
import matplotlib.pyplot as plt

#Recuperation de la base d'image MNIST et affichage de l'une d'elle
# load (downloaded if needed) the MNIST dataset
(X_train, y_train), (X_test, y_test) = mnist.load_data()


# reshape to be [samples][width][height][channels]
X_train = X_train.reshape(X_train.shape[0], 784).astype('float32')
X_test = X_test.reshape(X_test.shape[0], 784).astype('float32')
# normalize inputs from 0-255 to 0-1
X_train = X_train / 255
X_test = X_test / 255
# one hot encode outputs
y_train = np_utils.to_categorical(y_train)
y_test = np_utils.to_categorical(y_test)


#Creation du reseau de neurone simple
RNN=Sequential()
RNN.add(Dense(784, input_dim=784, activation="relu", kernel_initializer="normal"))
RNN.add(Dense(10, activation='softmax', kernel_initializer="normal"))
RNN.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
print("_"*140+'\n')
print("_"*140+'\n')
RNN.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=1, batch_size=200, verbose=1)

RNN.save("model_RNN_handwritting.h5")
Fiabilite=RNN.evaluate(X_test,y_test, verbose=0)
print("\nFiabilite lors de la phase d'évaluation:     " + str(round(100*Fiabilite[1],2)) + "%")
print("\nLe réseau entraîné est enregistré dans :model_RNN_handwritting.h5")
print("_"*140+'\n')
```

Au l'exécution de ce code, nous obtenons le résultat suivant :

```
(base) I:\MCEN4_STM32\MNIST>python mnist_rnn_train.py
Using TensorFlow backend.
Downloading data from https://s3.amazonaws.com/img-datasets/mnist.npz
11493376/11490434 [==============================] - 4s 0us/step
2022-03-14 15:47:20.505022: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2
_________________________________________________________________


_________________________________________________________________
Train on 60000 samples, validate on 10000 samples
Epoch 1/1
60000/60000 [==============================] - 3s 56us/step - loss: 0.2806 - accuracy: 0.9197 - val_loss: 0.1366 - val_accuracy: 0.9608

Fiabilite lors de la phase d'évaluation:     96.08%

Le réseau entraîné est enregistré dans :model_RNN_handwritting.h5
_________________________________________________________________


(base) I:\MCEN4_STM32\MNIST>
```

Un epoch en IA est un nombre de cycle que le programme complète dans son processus de formation. Plus le nombre d'epoch est élevé, plus la précision est forte. Pour améliorer la précision du programme, on peut alors augmenter le nombre d'epoch.

```
36
37    RNN.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=3, batch_size=200, verbose=1)
38
```

```
(base) I:\MCEN4_STM32\MNIST>python mnist_rnn_train.py
Using TensorFlow backend.
2022-03-14 16:01:53.330166: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2
_________________________________________________________________


_________________________________________________________________
Train on 60000 samples, validate on 10000 samples
Epoch 1/3
60000/60000 [==============================] - 3s 56us/step - loss: 0.2851 - accuracy: 0.9178 - val_loss: 0.1440 - val_accuracy: 0.9581
Epoch 2/3
60000/60000 [==============================] - 3s 53us/step - loss: 0.1129 - accuracy: 0.9680 - val_loss: 0.0959 - val_accuracy: 0.9716
Epoch 3/3
60000/60000 [==============================] - 3s 54us/step - loss: 0.0724 - accuracy: 0.9790 - val_loss: 0.0827 - val_accuracy: 0.9739

Fiabilite lors de la phase d'évaluation:     97.39%

Le réseau entraîné est enregistré dans :model_RNN_handwritting.h5
```

On constate qu'en augmentant le nombre d'epoch à 3, la fiabilité a augmenté et est passée de 96.08% à 97.39%.

## 2.5. Réseau neuronal simple : Inférence

En intelligence artificielle, l'inférence est l'opération de déduction à partir d'informations implicites. L'inférence permet de créer des liens entre les informations afin d'en tirer une conclusion.

Récupérons le code python mnist_rnn_inference.py et lançons-le avec Anaconda. On obtient un affichage comme celui-ci :

Le second tableau permet de tester chaque valeur de 0 à 9. On peut voir que le programme est sûr à 98% qu'il s'agit du chiffre 4. Ce tableau est du a la ligne de code suivante :

```
41
42    print(DataFrame(numpy.vstack([resultat,prediction])))   #vstack concatene les lignes
43
```

Nous devons maintenant ne plus nous servir des images de la base MNIST, mais de les remplacer par une seule image que nous avons dessinée. Nous allons utiliser l'image dessiné en début de projet sur le STM32.

Nous allons alors créer un nouveau fichier python en se basant sur l'ancien code pour effectuer ces modifications :

Pour la première étape, nous allons transformer l'image déjà enregistrer en un tableau de niveau de gris. Le code est le suivant :

```
 7   #Recupération de l'image 28x28 créée précedemment
 8   im = Image.open('image_finale.png')
 9   imgArray = numpy.asarray(im)
10   print("_"*140+'\n')
11   print("_"*140+'\n')
12   print(DataFrame(imgArray))
13   print("\n\n")
```

Nous avions enregistrer un 9 et le tableau correspondant est le suivant :

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 4 | 2 | 1 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 4 | 103 | 172 | 195 | 54 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 14 | 179 | 255 | 255 | 255 | 188 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 1 | 23 | 200 | 255 | 98 | 123 | 252 | 205 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 2 | 175 | 255 | 68 | 0 | 85 | 255 | 144 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 22 | 252 | 137 | 0 | 4 | 85 | 255 | 108 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 19 | 241 | 189 | 32 | 0 | 117 | 255 | 171 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 1 | 77 | 255 | 198 | 14 | 187 | 255 | 249 | 38 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 98 | 255 | 252 | 255 | 249 | 255 | 49 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 72 | 138 | 131 | 224 | 237 | 15 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 39 | 255 | 114 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 7 | 4 | 112 | 255 | 30 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 207 | 235 | 14 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 1 | 1 | 0 | 0 | 244 | 153 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 17 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 240 | 146 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 95 | 131 | 0 | 0 | 4 | 3 | 241 | 147 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 89 | 255 | 188 | 38 | 0 | 0 | 240 | 146 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 114 | 255 | 255 | 154 | 72 | 245 | 151 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 57 | 167 | 255 | 255 | 255 | 85 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 22 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 32 | 48 | 50 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 23 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 24 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 25 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 26 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 27 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

En fait le réseau neuronal que nous avons entraîné n'est pas construit pour recevoir une image en 2 dimensions (28x28) mais en une seule ligne de 784 éléments. Nous devons alors effectuer cette modification :

Pour convertir le tableau en liste, nous devons utiliser la fonction reshape(). Nous mettrons les paramètre 1 ligne et 784 colonnes pour transformer le tableau.

```
14
15   #Transformation de la matrice en une liste
16   imgArray = imgArray.reshape(1,784)
17   print(imgArray)
18   print("\n\n")
19
```

On obtient alors la liste ci-dessous :

```
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   4   2
    1   2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   2   0   4
  103 172 195  54   0   2   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   1   0  14 179 255 255 255 188   0   2   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   2   1  23
  200 255  98 123 252 205   0   1   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   3   2 175 255  68   0  85 255 144   0   3
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   2   0
   22 252 137   0   4  85 255 108   0   4   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   1   1  19 241 189  32   0 117 255 171
    0   3   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   2   1  77 255 198  14 187 255 249  38   0   2   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   3   0  98 255 252 255
  249 255  49   0   2   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   4   0  72 138 131 224 237  15   1   1   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0  39 255 114   0   4   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   7   4 112 255  30   0   2   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    1   1   1 207 235  14   1   1   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   2   1   1   0   0 244 153   0   4   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   3   0   0 240 146   0   4   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   2   0  95 131   0   0   4   3 241 147   0
    4   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   3
    0  89 255 188  38   0   0 240 146   0   4   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   2   0 114 255 255 154  72 245
  151   0   4   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   2   0  57 167 255 255 255  85   0   3   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   3   0   0  32
   48  50   3   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   3   2   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   2   2   2   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]]
```

Pour finaliser ce programme, il faut fournir cette liste au réseau neuronal pour qu'il fasse l'inférence de l'image. Le code est le suivant :

```
#Chargement du reseau de neurone simple
RNN=load_model('model_RNN_handwritting.h5')
resultat= RNN.predict(imgArray,batch_size=None, verbose=1, steps=None, callbacks=None, max_queue_size=10, workers=1, use_multiprocessing=False)

# normalize inputs from 0-255 to 0-1
imgArray = imgArray / 255

#Prediction
prediction=numpy.zeros(10)  #une matrice ligne de 10 zéros
prediction[numpy.argmax(resultat)]=1                #argmax renvoi l'indice du + grand element du tableau

print(DataFrame(numpy.vstack([resultat, prediction])))   #vstack concatene les lignes

#Affichage du résultat
print("\nPour ce réseau ce chiffre est :",numpy.argmax(resultat))       # indice tableau de la valeur max
print("_"*140+'\n')
```

En exécutant ce code, nous obtenons le résultat suivant, avec comme image un chiffre 3 :

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 2 | 0 | 2 | 4 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 3 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 2 | 1 | 64 | 192 | 240 | 201 | 146 | 50 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 43 | 163 | 150 | 196 | 255 | 255 | 185 | 89 | 40 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 25 | 101 | 218 | 255 | 255 | 117 | 1 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 3 | 4 | 2 | 0 | 0 | 0 | 44 | 199 | 213 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 6 | 3 | 0 | 213 | 176 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 2 | 0 | 0 | 0 | 144 | 255 | 56 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 16 | 35 | 122 | 255 | 166 | 2 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 40 | 172 | 243 | 255 | 255 | 171 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 13 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 14 | 209 | 255 | 255 | 220 | 132 | 213 | 255 | 157 | 65 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 14 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 12 | 184 | 162 | 92 | 9 | 0 | 19 | 116 | 251 | 255 | 77 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 24 | 230 | 175 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 4 | 4 | 1 | 0 | 1 | 7 | 0 | 218 | 171 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 17 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 0 | 0 | 221 | 179 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 4 | 4 | 4 | 2 | 0 | 0 | 53 | 163 | 255 | 90 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 21 | 177 | 255 | 255 | 144 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 20 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 108 | 174 | 171 | 173 | 202 | 250 | 232 | 127 | 28 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 21 | 0 | 0 | 0 | 0 | 0 | 2 | 1 | 149 | 221 | 218 | 220 | 202 | 107 | 19 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 22 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 23 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 2 | 3 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 24 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 25 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 26 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 27 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

```
2022-03-25 11:38:45.744598: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2
1/1 [==============================] - 0s 0us/step
    0    1    2    3    4    5    6    7    8    9
0  0.0  0.0  0.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0
1  0.0  0.0  0.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0

Pour ce réseau ce chiffre est : 3
```

On constate que le programme fonctionne parfaitement et on peut constater sur le second tableau qu'il est sûr à 100% que le chiffre est un 3.

La dernière partie de ce sujet est de mélanger l'interface graphique permettant de dessiner, avec l'intelligence artificielle pour prédire un chiffre qui est dessiné.

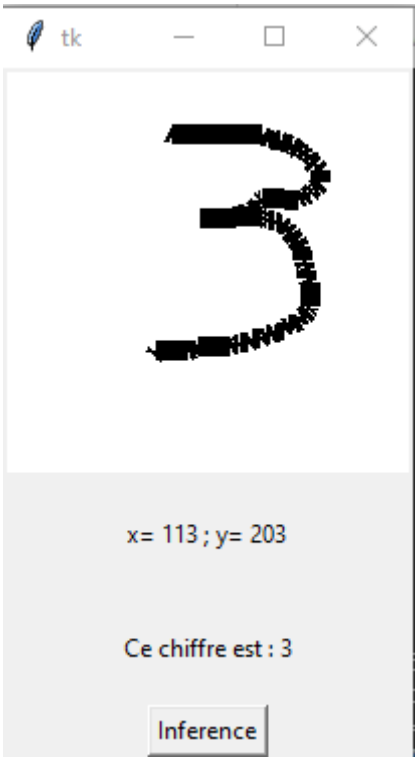
Nous allons alors créer un nouveau fichier python permettant de faire l'inférence d'un dessin. Nous devons alors garder le principe de la première interface en rajoutant un bouton 'Inférence' permettant d'enregistrer l'image dessiné en png, de lancer l'inférence du dessin et d'afficher dans une zone de texte le résultat obtenu.

```python
from tkinter import *
from PIL import Image, ImageOps                                 #ImageOps for grayscale
from keras.models import load_model
from pandas import DataFrame

import numpy

#Variables globales
old_x, old_y = 0, 0

#This function displays the mouse coordinates in Label widget
def MouseCoord(event):
    global cadre
    global old_x, old_y                                                 #Allows to modify a global variable
    X = event.x
    Y = event.y
    coord.set("x= "+ str(X)+ " ; y= "+ str(Y))

    old_x = event.x
    old_y = event.y


def Dessin(event):
    global old_x, old_y
    x = event.x
    y = event.y
    cadre.create_line(old_x, old_y ,x, y, width= 10)

    old_x = event.x
    old_y = event.y

def Save_as_png():
    global cadre        #Allows to modify a global variable
    # MGA: bellows are listed the different steps to store your canvas drawing into a 28px x 28 px grayscale png picture
    # 1 - save postscipt image
    imgps_save = cadre.postscript(file='image.ps')

    # 2 - use PIL to convert to PNG
    image = Image.open('image.ps')
```

```python
    # 3 - use ImageOps to invert black and white
    inverted_image = ImageOps.invert(image)

    # 4 - use PIL to convert in grayscale
    image_grayscale = ImageOps.grayscale(inverted_image)

    # 5 - use PIL to reduce picture to 28px x 28 px
    image_reduce = image_grayscale.resize((28,28), Image.ANTIALIAS)

    # 6 - use PIL to save picture as png
    image_reduce.save('image_finale.png')
    Reconnaissance()

def Reconnaissance():

    global affichagenombre,resultat
    #Recupération de l'image 28x28 créée précedemment
    im = Image.open('image_finale.png')
    imgArray = numpy.asarray(im)
    print("_"*140+'\n')
    print("_"*140+'\n')
    print(DataFrame(imgArray))
    print("\n\n")

    #Transformation de la matrice en une liste
    imgArray = imgArray.reshape(1,784)
    print(imgArray)
    print("\n\n")

    #Chargement du reseau de neurone simple
    RNN=load_model('model_RNN_handwritting.h5')
    resultat= RNN.predict(imgArray,batch_size=None, verbose=1, steps=None, callbacks=None, max_queue_size=10, workers=1, use_multiprocessing=False)

    # normalize inputs from 0-255 to 0-1
    imgArray = imgArray / 255
```

```
    #Prediction
    prediction=numpy.zeros(10)  #une matrice ligne de 10 zéros
    prediction[numpy.argmax(resultat)]=1                    #argmax renvoi l'indice du + grand element du tableau

    print(DataFrame(numpy.vstack([resultat, prediction])))   #vstack concatene les lignes

    #Affichage du résultat
    print("\nPour ce réseau ce chiffre est :",numpy.argmax(resultat))       # indice tableau de la valeur max
    print("_"*140+'\n')
    affichagenombre.set("Ce chiffre est : "+str(numpy.argmax(resultat)))


fenetre1=Tk()    #Swinen's book p76

cadre=Canvas(fenetre1, width=200, height=200, bg='#FFFFFF')     #p89
cadre.bind("<Motion>", MouseCoord)
cadre.bind("<B1-Motion>", Dessin)
#cadre.bind("<Button-3>", Save_as_png)                           # call the save function on left click
cadre.pack()

#bouton inférence
bouton = Button(text='Inference', command = Save_as_png)
bouton.pack(side='bottom')

# create here your Label widget
coord = StringVar()
labelcoord = Label(fenetre1, textvariable = coord , bd = 20)
coord.set("x= "+ X+ " ; y= "+ Y)
labelcoord.pack()

affichagenombre = StringVar()
labelaffichage = Label(fenetre1, textvariable = affichagenombre , bd = 20)

labelaffichage.pack()

fenetre1.mainloop()
```

En lançant ce code sur anaconda, nous obtenons la fenêtre ci-dessous. En dessinant un chiffre, on constate que le programme trouve le bon chiffre.



# Conclusion

Pour conclure, ce projet de MCEN4 nous a permis d'améliorer nos connaissances en systèmes embarquées et nos notions de C++. Nous avons appris à utiliser un microcontrôleur STM32 ainsi que de gérer un écran TFT relié à celui-ci. La deuxième partie du projet sur l'intelligence artificielle nous à permis de mieux comprendre comment se phénomène fonctionne. Nous avons découvert le langage python qui est très utilisé dans ce domaine.

Par manque de temps, nous n'avons pas pu mélanger la partie microcontrôleur et intelligence artificielle, qui pourrait avoir comme but de dessiner un chiffre avec son doigt sur l'écran pour que le programme puisse ensuite déterminer ce chiffre.