# Coin classification and amount Calculation

Created by:

- Mkawar Wael (DLSI_MM_TD2_TP2)
- Douma Hamdi (DLSI_MM_TD2_TP1)

# summary

# Introduction

- the project is about making an algorithm that can automatically identify coins and calculate their total value.
- The goal of this system is to simplify tasks for people.

# problematic

- Counting coins by hand can take a long time, especially with large amounts.
- mistakes in counting or adding up coin values can result in incorrect totals.
- identifying coins of different types can be difficult, particularly when they look alike.

# Dataset Creation and Preprocessing



- **creating dataset :**

  We collected images of various coins from different angles, lighting conditions, and backgrounds to ensure a robust dataset.

- **Preprocessing steps:**

  Before training our model, we resized all images to a standardized resolution, converted them to grayscale, and normalized pixel values.

- **Labelling coins with their corresponding values:**

  Each coin image in our dataset is labelled with its corresponding value (e.g., 100 millime, 50 millime).

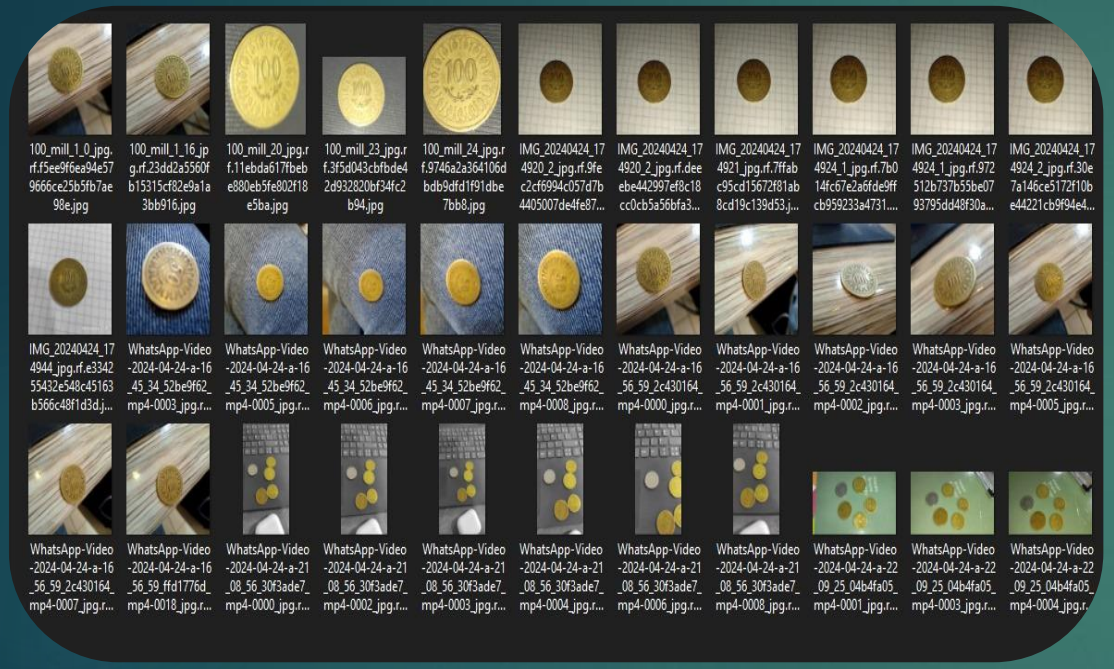
# Creating dataset

```
from roboflow import Roboflow
rf = Roboflow(api_key="flZP1Jus8QmKKqLJeCvZ")
project = rf.workspace("coins-pw0uj").project("coins-tunisia")
version = project.version(2)
dataset = version.download("yolov5")
model = project.version(1).model
```

[64] ✓ 8.3s

```
loading Roboflow workspace...
loading Roboflow project...
```

# Data used



# coin selecting and labelling

# Test preparing



```
#preparing test data

x_test=[]
y_test=[]

path_images_test_0="D:/dataset/test/100"
images_0 = os.listdir(path_images_test_0)

for img_name in images_0:
    path_img= os.path.join(path_images_test_0,img_name) #getting the path of an image

    image = load_img(path_img, target_size=size) #uploading the image
    image = img_to_array(image) #getting the array values of the image
    #print(image)
    image_grayscale = tf.image.rgb_to_grayscale(image) #converting the RGB image into grayscale
    image_grayscale = np.array(image_grayscale)
    #print(image_grayscale)
    x_test.append(image_grayscale)
    y_test.append(0)


path_images_test_0="D:/dataset/test/50"
images_0 = os.listdir(path_images_test_0)

for img_name in images_0:
    path_img= os.path.join(path_images_test_0,img_name) #getting the path of an image

    image = load_img(path_img, target_size=size) #uploading the image
    image = img_to_array(image) #getting the array values of the image
    #print(image)
    image_grayscale = tf.image.rgb_to_grayscale(image) #converting the RGB image into grayscale
    image_grayscale = np.array(image_grayscale)
    #print(image_grayscale)
    x_test.append(image_grayscale)
    y_test.append(1)

x_train=np.array(x_train)
y_train=np.array(y_train)

x_test=np.array(x_test)
y_test=np.array(y_test)
```

# Creating a CNN model



- **Algorithm Selection :** Chose CNN for best results.
- **Model Architecture :** CNN with tuned hyperparameters.
- **Training Process :** Prevented overfitting with dataset split, and used dropout and batch normalization.

# prediction



```
#prediction
y_pred=[]
for test in x_test:

  responce=model.predict(test,confidence=3,overlap=40).json()
  print([i["class"] for i in responce['predictions']])
  y_pred.append(100)

y_pred=np.array(y_pred)
print("prediction: ", y_pred)
print("ground truth: ", y_test)

#confusion matrix
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
print('acc=',accuracy_score(y_test,y_pred))
```

# result



```
100
100
200
200
prediction:  [100 100 200 200]
ground truth:  [100 100  50  50]
[[0 0 2]
 [0 2 0]
 [0 0 0]]
              precision    recall  f1-score   support

          50       0.00      0.00      0.00         2
         100       1.00      1.00      1.00         2
         200       0.00      0.00      0.00         0

    accuracy                           0.50         4
   macro avg       0.33      0.33      0.33         4
weighted avg       0.50      0.50      0.50         4

acc= 0.5
```

References:


- https://roboflow.com/
- https://opencv.org/
- https://tensorflow.org/
- https://keras.io/