Rapport : Apprentissage Profond

Reconnaissance de valeur de monnaie

Groupe G6 (L3-L4)
AMGHAR Souhail L1
ECH-CHEBLAOUY Yassine L3
JIA Qingwei L3

Département Sciences du Numérique - Deuxième année
2020-2021

Table des matières

Table des figures

1 Introduction

1.1 Description

Depuis la création de la monnaie unique européenne, il existe huit billets en euros répartis sur deux séries, dont les valeurs s'échelonnent de 5 à 500 euros dont huit pièces de monnaies (1 cent, 2 cent,5,10, 20,50, 1€, 2€). (ceux qui font les champs de notre classification de la DB par la suite).

De l'introduction de ces billets le 1er janvier 2002 au 1er mai 2013, une seule série a été en circulation, mais une deuxième série a commencé à remplacer celle-ci à partir du 2 mai 2013.

Contrairement aux pièces en euros, le design des billets est le même pour l'ensemble de la zone euro. Ils sont cependant imprimés et émis par les différents États membres. Quel que soit le pays émetteur, les billets sont cependant utilisables dans tous les pays dont l'euro est la monnaie. Ils sont conçus en pure fibre de coton, ce qui leur donne une meilleure résistance et ce qui permet de les reconnaitre facilement au toucher. Jusqu'à présent, la taille des billets s'échelonne d'un format de 120 sur 62 millimètres pour le billet de 5 euros à 160 sur 82 millimètres pour le billet de 500 euros. Ils ont tous une couleur différente, en particulier avec les billets de valeur proches, afin de pouvoir bien les distinguer. Ce qui fait effectivement une sorte de bruit (eng. noice) naturel dans nos données.

Clarification : Pour nos pièces de monnaie, on utilisera ceux de la France.

1.2 Méthodologie d'acquisition des données

L'acquisition de la base de données passe par plusieurs étapes :

- Recherche sur Google de notre topic de données.
- Collecte des liens de tous les images qui correspondent au topic recherché à partir d'un script en Javascript.

- Utilisation d'un script Python pour télécharger les images voulus à partir des liens collecté précedemment.
- Répartition des données (ci-dessous).

1.3 Méthodologie de répartition des données

La répartition se fait principalement en plaçant les données (images) sur une hiérarchie de dossiers spécifique. Ils sont d'abord divisés en 3 parties : données d'entrainement, de validation et de test. Après, chacune des parties est divisé en plusieurs parties représentant les champs de nos données : de 1c jusqu'à 10e.

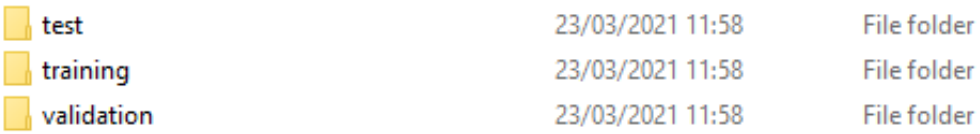

FIGURE 1 – Types de données

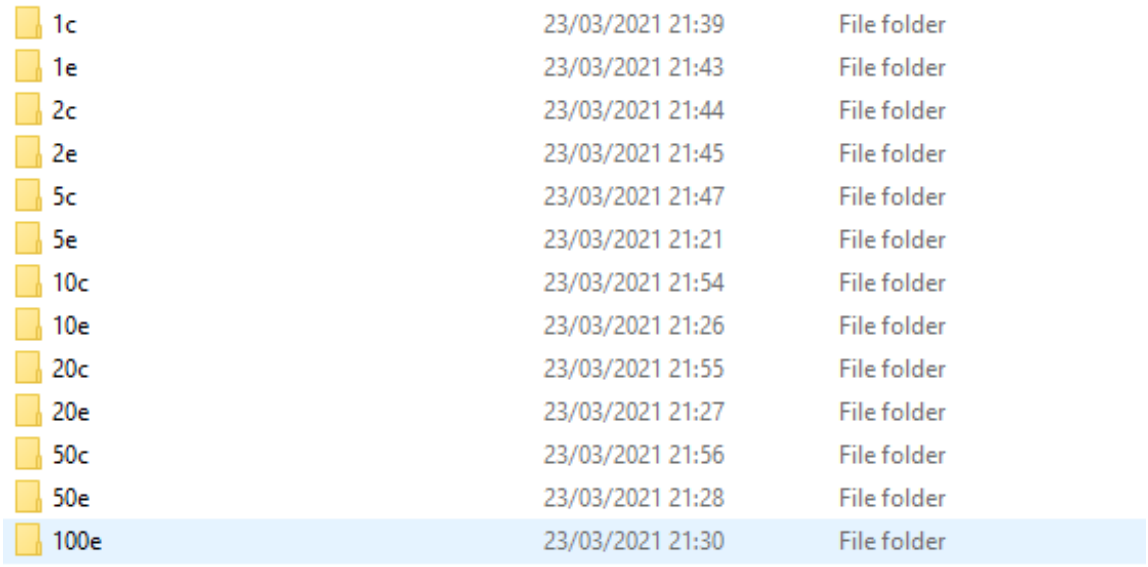

FIGURE 2 – Classes de données

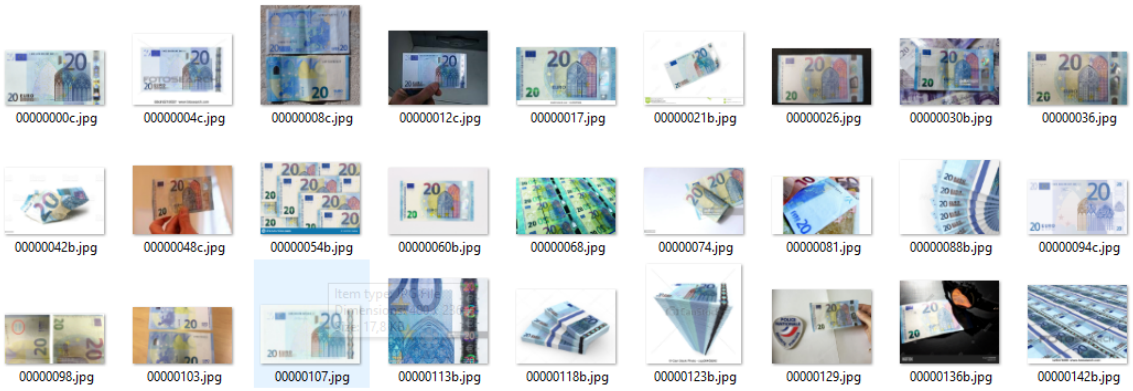

FIGURE 3 – Nomination des données

1.4 Pronostic

Nous estimons que le défi que nous devons relever avec notre projet d'apprentissage profond n'est pas simple. Le projet répond à un ap-

prentissage suprvisé de type classification multinomiale avec un grand problème de bruit sur la base de données. Surtout avec des catégories de billets qui se ressemblent (même couleurs, même taille, même font et même design), qui rend l'apprentissage plus challengeant et complexe.

2 Démarches du projet

2.1 Initiation

On a crée un dépot git sur lequel on stocke nos données et notre script Google colab dans lequel on écrit notre code. On commence au début par importer toutes les libraires dont on aurait besoin au cours de la réalisation de notre projet. Ensuite, on importe notre git repo pour charger les données sur le colab. Juste après, on utilise une fonction qui charge nos données, à partir du git chargé, et les divises en 3 types (entrainement, validation et tests) puis les classifient.

```
labels = ['1c', '2c', '5c', '10c', '20c', '50c', '1e', '2e', '5e', '10e', '20e', '50e', '100e']

x_train, y_train = load_data(path, labels, dataset='training', image_size=150)

x_val, y_val = load_data(path, labels, dataset='validation', image_size=150)

x_test, y_test = load_data(path, labels, dataset='test', image_size=150)
```

FIGURE 4 – Output du chargement de données

2.2 Version simple du modèle

On a commencé par implémenter un modèle simplistic inspiré des tps et qui va jouer le rôle d'un template pour les améliorations qu'on va introduire.
C'est un modèle convolutionnel de :

- 5 couche de convolutions 32 $\rightarrow$ 256 avec activation relu
- une couche flatten

- une couche dense 512 avec activation relu et une input_dim 128
- une couche de sortie dense 13 avec activation softmax.

On compile notre réseau avec une fonction de coût "categorical crossentropy" et un optimiseur Adam avec taux d'entrainement 3e-4 On observe que le modèle donne des résultats positivement inattendus :

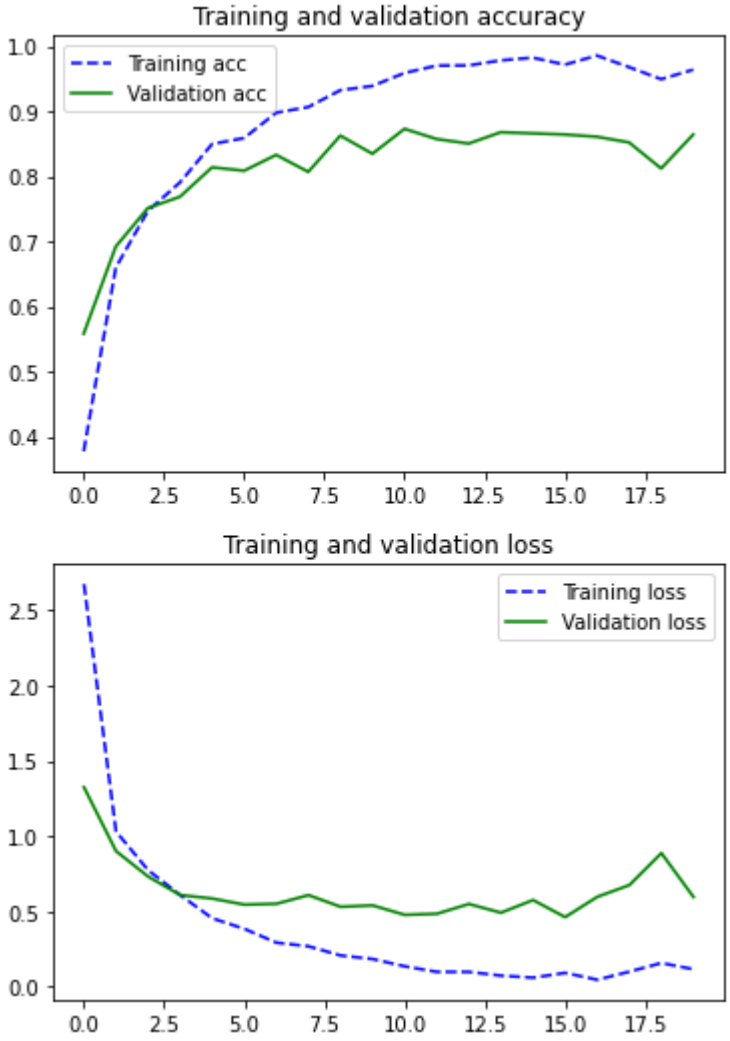

FIGURE 5 – Résultat de la version initiale

2.3 Améliorations sur la version simple

On observe que dans la version initiale qu'on a obtenu de l'overfitting. Donc on y introduit quelques améliorations :

- ajout de la régularisation
- utilisation de données augmentées.

Plus concrètement, on ajoute aux couches conv et denses un kernel_regularizer L2 de 0.01, et on fourni au modèle un générateur de données augmentées :

```
train_datagen = ImageDataGenerator(
    rotation_range=40,
    rescale=1./255,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,)
```

FIGURE 6 – Générateur de données augmentées

Les résultats ne sont pas assez satisfaisants :

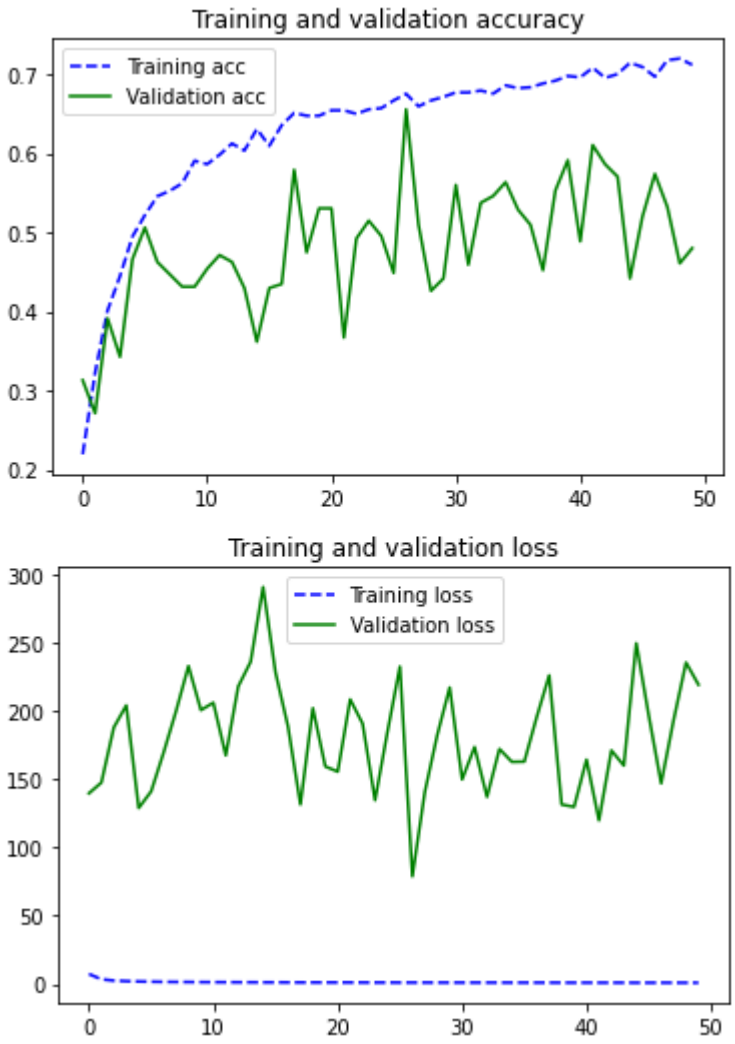

FIGURE 7 – Résultat de la version "améliorée"

2.4 Introduction du transfer learning :

2.4.1 VGG-16 verouillé :

On essaye d'améliorer notre modèle en introduisant du transfer learning par le biais de VGG-16. Plus concètement, on remplace dans notre modèle précedent les couches de convolutions par les couches du réseau VGG-16 verouillé.
On ne remaque pas une amélioration dans notre réseau :

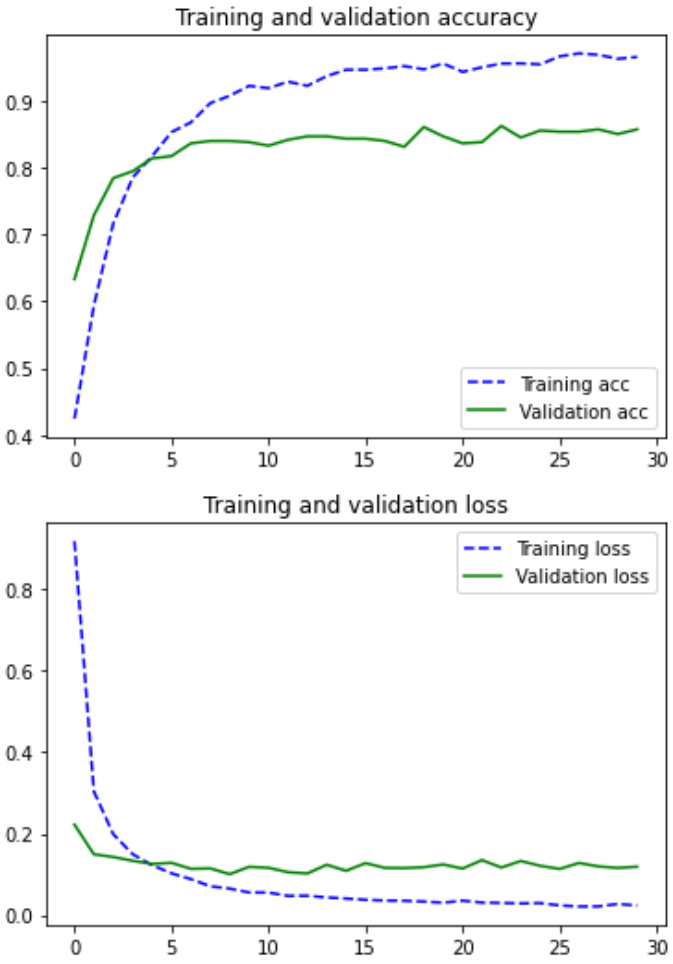

FIGURE 8 – Résultat de la version VGG-16 verouillé

2.4.2 VGG-16 déverouillé

On essaye d'améliorer notre accuracy en débloquant notre réseau VGG, et on utilisant un faible taux d'entrainement 1e-5 (faible pour éviter un overfitting). Et de même, on a pas observé une grande amélioration dans les résultat :

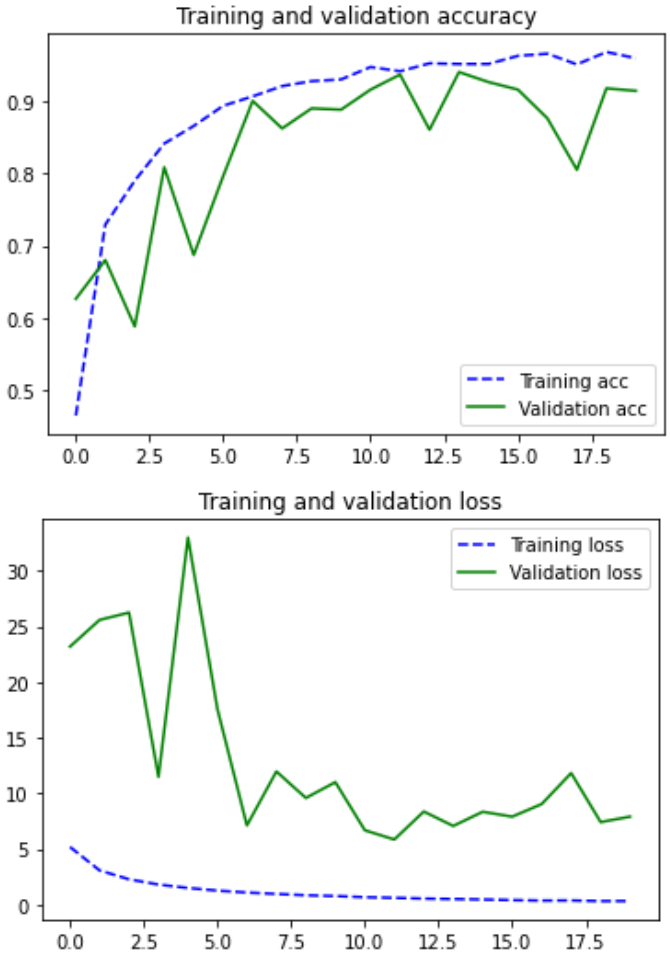

FIGURE 9 – Résultat de la version VGG-16 déverouillé

2.5 Conclusion et version finale

On a sorti avec la conclusion que notre problème est simple, et en essayant de compliquer notre réseau, on obtient des résultats moins pertinent que ceux obtenu avec un réseau plus simple. Et donc on va prendre la version simple améliorée comme notre modèle principale, et on va lui introduire des améliorations pour réduire l'overfitting :

- on élimine la couche de convolution à 256 noeuds
- on remplace la couche flatten par la couche GlobalAveragePooling
- on ajoute un bias_regulizer L2 de 0.01 à la dernière couche de convolution.

```
model_reg = models.Sequential()

#Convolution
model_reg.add(Conv2D(32, 3, activation='relu', input_shape=(150, 150, 3)))
model_reg.add(MaxPooling2D())
model_reg.add(Conv2D(64, 3, activation='relu'))
model_reg.add(MaxPooling2D())
model_reg.add(Conv2D(96, 3, activation='relu'))
model_reg.add(MaxPooling2D())
model_reg.add(Conv2D(128, 3, activation='relu', kernel_regularizer=regularizers.l2(0.01), bias_regularizer=regularizers.l2(0.01)))
model_reg.add(MaxPooling2D())

#Flatten
model_reg.add(GlobalAveragePooling2D())

# Fully connected
model_reg.add(Dense(512, activation='relu', input_dim=128, kernel_regularizer=regularizers.l2(0.01)))
model_reg.add(Dense(13, activation='softmax'))

model_reg.compile(loss='categorical_crossentropy',
              optimizer=optimizers.Adam(lr=3e-4),
              metrics=['acc'])
```

FIGURE 10 – Code de la version finale du modèle

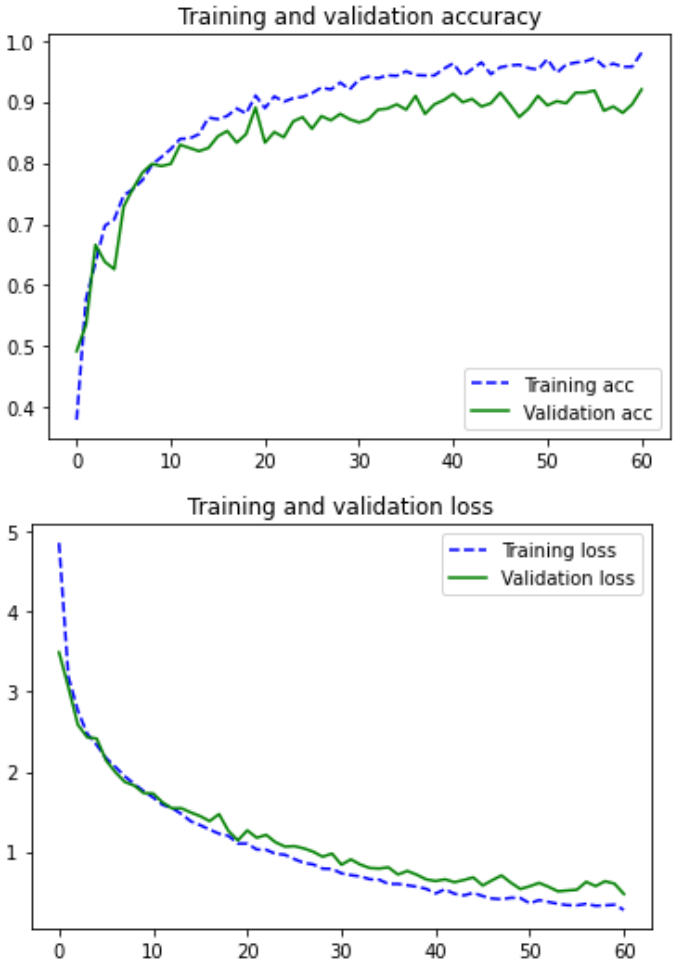

FIGURE 11 – Courbe résultat de la version finale

3 Analyse du résultat

3.1 Analyse métrique

3.1.1 Analyse globale :

On a obtenu comme précisions et coûts globales :

	Coût	**Accuracy**
Entrainement	0.27	0.98
Validation	0.46	0.92
Test	0.58	0.89

On observe que les résultats ci-dessus sont relativement biens, mais pas assez suffisament. Ainsi, on analyse plus profondément pour voir ce qui manque dans notre modèle ou base de données.

3.1.2 Analyse par classes :

Pour effectuer une analyse par classe, on fait appel à la fonction classification report, pour laquelle on donne comme paramètres nos données prédits et nos données de test (avec des changements de dimensions pour respecter les inputs de la fonction).

	precision	recall	f1-score	support
1c	0.83	1.00	0.91	15
2c	1.00	0.33	0.50	6
5c	0.99	0.99	0.99	84
10c	0.83	0.96	0.89	78
20c	0.91	0.87	0.89	71
50c	0.90	0.84	0.87	31
1e	1.00	0.83	0.91	59
2e	0.79	1.00	0.88	11
5e	0.96	0.76	0.85	34
10e	0.84	0.84	0.84	31
20e	0.91	0.94	0.92	52
50e	0.94	0.90	0.92	50
100e	0.77	0.91	0.84	45
accuracy			0.90	567
macro avg	0.90	0.86	0.86	567
weighted avg	0.91	0.90	0.90	567

FIGURE 12 – Analyse métrique par classes

On peut facilement voir qu'il y a une difference sur les précisions/recalls entre certains classes (surtout celle de 2c). Ceci est sûrement dû au fait qu'il n'y a pas assez de données dans cette classe.

3.2 Matrice de confusion

On va faire appel à la matrice de confusion pour s'assurer de notre hypothèse.

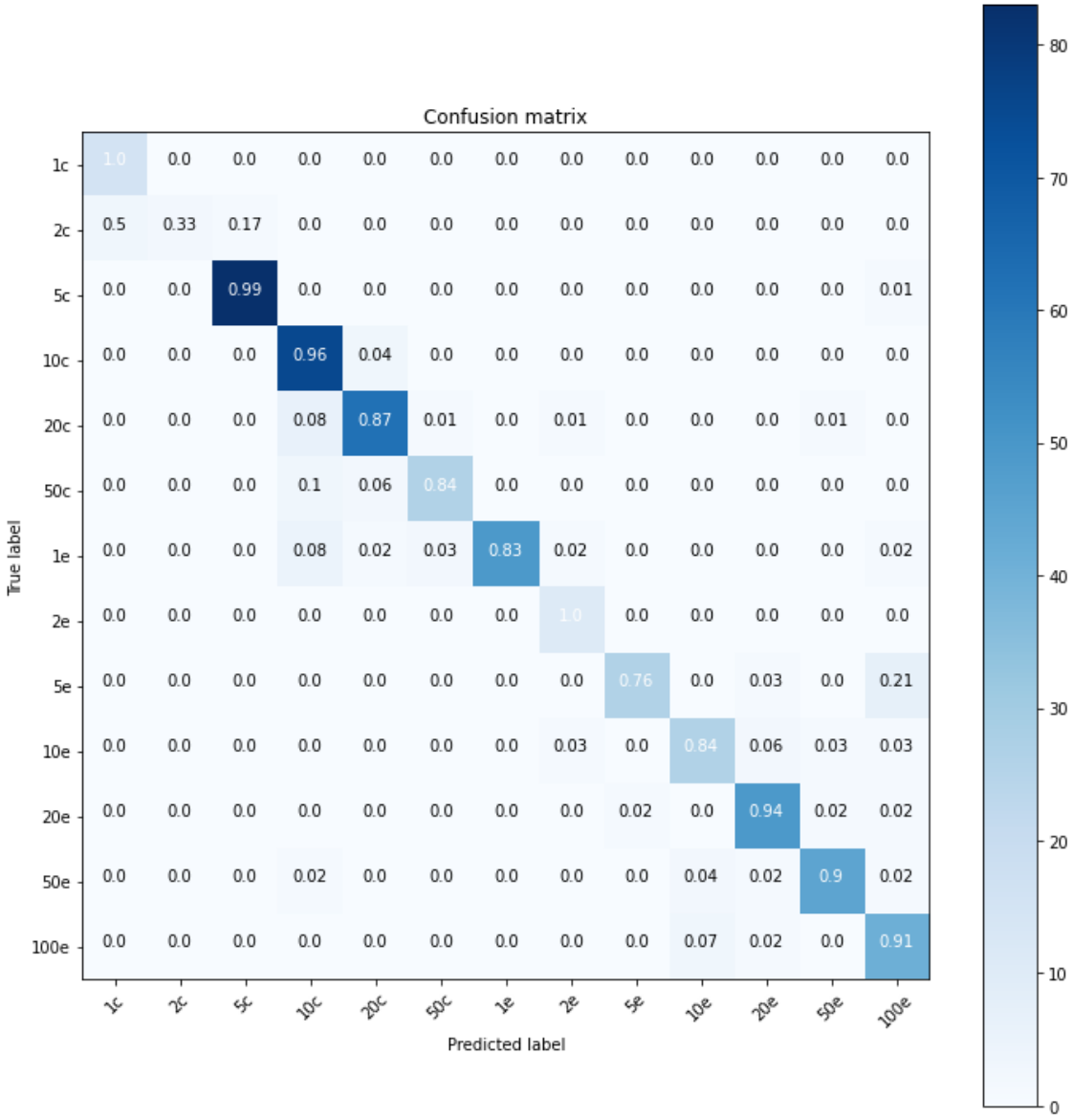

FIGURE 13 – Matrice de confusion normalisée

En analysant la matrice, on peut sortir avec deux conclusions principales :
On s'est premièrement, assuré de notre hypothèse. La classe 2c est confus avec les classes 1c et 5c surtout qu'ils se ressemblent.

Aussi, on observe qu'il y a une confusion entre la classe 5e et 100e, ceci peut être dû à la ressemblance en couleur entre les deux types de billets.

3.3 Analyse qualitative

FIGURE 14 – Comparaison entre le billet de 5e et celui de 100e

Encore une fois, nos doutes étaient réelles. On a une ressemblance entre les couleurs entre les images des deux billets, et ça peut être la cause de la confusion entre les deux billets.

4 Conclusion

Pour conclure, on juge que les résultats était satisfaisants pour un premier grand projet en apprentissage profond. On avait prévu que notre problème était bien plus complexe qu'il l'est en réalité, ce qui nous a poussé à perdre beaucoup de temps en essayant de créer un modèle complexe correcte (en utilisant l'augmentation de données et le transfer learning) alors que se concentrer sur un modèle simple et l'optimiser aurait fait l'affaire. Un autre problème qu'on a rencontré c'est le redimensionnement de nos inputs pour qu'ils soient compatibles avec les fonctions d'analyse (classification report et matrice de confusion). Pour améliorer les résultats de notre projet, on peut prévoir d'avoir une base de données plus équilibrées (surtout pour la classe 2c), et on peut user de l'augmentation de données pour mettre en relief la différence sur les niveaux de

couleurs afin d'essayer de résoudre le problème de confusion entre les classes 5e et 100e.