

Identification de chiffres avec un réseau de neurones convolutif

Projet réalisé par Cyril et Jonathan

Sommaire

1. Rappel du contexte
2. Création du modèle
3. Développement de l'interface utilisateur (via streamlit)
4. Déploiement de l'application
5. Démonstration

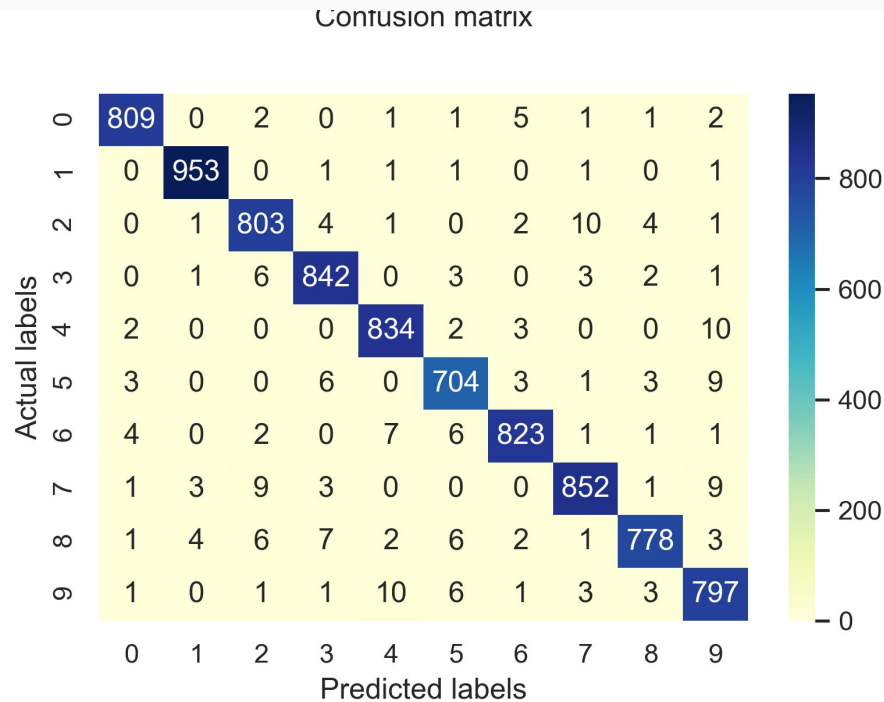
Rappel du contexte

- **Objectif du projet** : créer le frontend d'une application pour reconnaître les chiffres (entre 0 et 9)
- Deux jeux de données (train.csv et test.csv)
- Création d'un modèle en utilisant le deep learning (CNN)
- Utilisation de streamlit et d'Heroku pour la création et le déploiement de l'application
- Mise à disposition de l'application pour le client

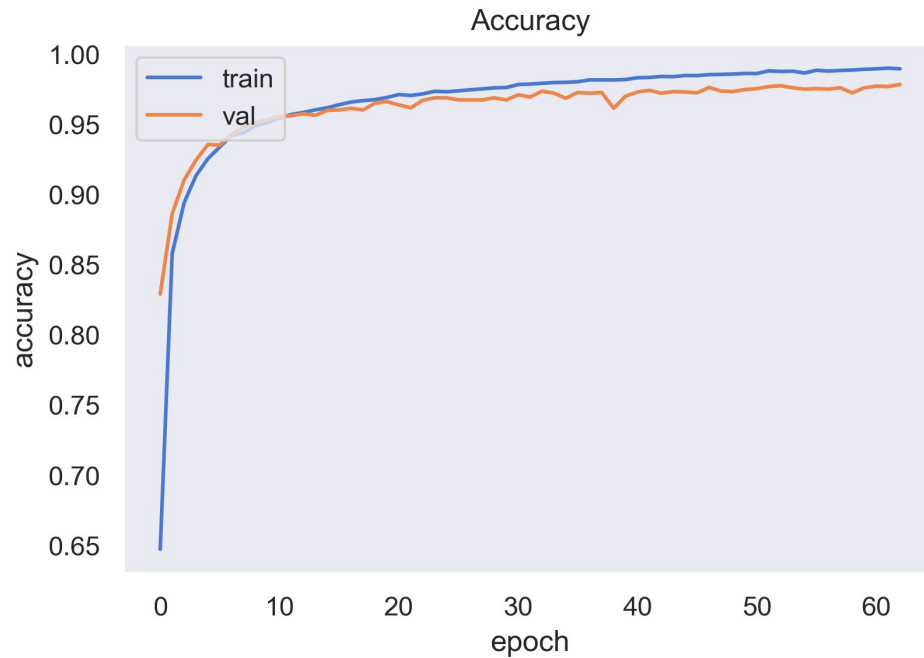
Création du modèle

- Plusieurs tests de modèle, **modèle retenu** :
- 3 couches convolutionnelles :
 - 1ère couche - 8 filtres (kernel [3,3])
 - 2ème couche - 16 filtres (kernel [3,3])
 - 3ème couche - 32 filtres (kernel[3,3])
- 2 opérations d'applatissements :
 - 1ère couche - 256 flattening
 - 2ème couche - 512 flattening
- **Dropout : 0,1 ; Learning rate : 0,0001 ; Epochs : 100** (early stopping à partir de 10) ; **Batch size = 64 ; Validation split = 0.1**

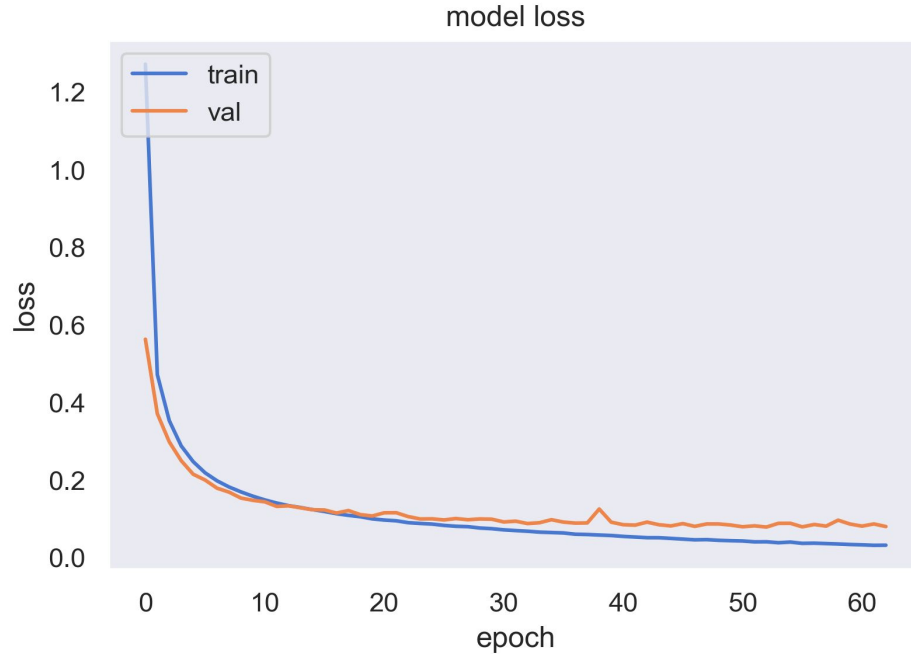
Statistiques du modèle



Statistiques du modèle



Statistiques du modèle



Développement de l'interface utilisateur

- Utilisation de streamlit pour créer le frontend de l'application
- Un seul fichier contenant l'application (app.py)
- Refactorisation pour rendre le code plus lisible et moins lourd
- Deux parties principales dans le script, avec une partie pour l'outil aléatoire et une autre pour l'outil de dessin.

Fonctions

```
def load_data():  
    data_test = pd.read_csv(uploaded_file)  
    st.header('csv uploaded!')  
    return data_test
```

Charger le csv pour la partie
des prédictions sur chiffre
aléatoire

```
def viz_num(img):  
    #Reshape the 768 values to a 28x28 image  
    image = img.reshape([28,28])  
    fig = plt.figure(figsize=(1, 1))  
    plt.imshow(image, cmap=plt.get_cmap('gray'))  
    plt.axis(["off"])  
    return fig
```

Afficher l'image d'un chiffre
de notre csv

```
def predict_random_picture(csv):  
    rand_img = data.sample()  
    img_test = rand_img.values.reshape(rand_img.shape[0], 28, 28, 1)  
    pred = model.predict(img_test)  
    pred = np.argmax(pred, axis=1)  
    st.write('Prediction: ', int(pred))  
    st.pyplot(viz_num(img_test))
```

Prédire le chiffre d'une
image et l'afficher

```
def predict_drawn_image(number):  
    image = Image.fromarray((number_drawn[:, :, 0]).astype(np.uint8))  
    image = image.resize((28, 28))  
    image = image.convert('L')  
    image = (tf.keras.utils.img_to_array(image)/255)  
    image = image.reshape(1,28,28,1)  
    x_2 = tf.convert_to_tensor(image)  
    pred = model.predict(x_2)  
    pred = np.argmax(pred, axis=1)  
    st.write('Predicted number : ' + str(pred))
```

**Prédire le chiffre que l'on
dessine avec notre
composant *st.canvas***

Randomizing tool

```
### Header ###
st.title('Digit Prediction App')
st.header('Which prediction tool to use?')

### Randomizing Tool View ###
if st.selectbox('Tools', ['Randomizing Tool', 'Drawing Tool']) == 'Randomizing Tool':
    ### Upload test dataset ###
    uploaded_file = st.file_uploader("Choose a file")
    if uploaded_file is not None:
        data = load_data()

        if st.button("Predict one picture"):
            predict_random_picture(data)
```

Chargement des
données

Affichage de la prédiction et
du chiffre

Drawing tool

Définition des
caractéristiques
du composant
st.canvas

```
### Drawing Tool View ###
else:
    st.write('Draw a number and let the app makes a prediction')

    # Specify canvas parameters in application
    stroke_width = st.sidebar.slider("Stroke width: ", 1, 25, 8)
    stroke_color = st.sidebar.color_picker("Stroke color hex: ", "#fff")
    bg_color = st.sidebar.color_picker("Background color hex: ", "#000")

    # Create a canvas component
    canvas_result = st_canvas(
        fill_color="rgba(255, 165, 0, 0.3)", # Fixed fill color with some opacity
        stroke_width=stroke_width,
        stroke_color=stroke_color,
        background_color=bg_color,
        height=150,
        width=150,
        key="canvas",
    )

    # Resize drawn image and predict it
    if canvas_result.image_data is not None:
        number_drawn = canvas_result.image_data
        if st.button('Predict your drawn number'):
            predict_drawn_image(number_drawn)
```






Affichage de la
prédiction et du
chiffre

Déploiement de l'application

- Déploiement sur streamlit dans un premier temps
- Déploiement sur Heroku dans un deuxième temps
- Premier déploiement sur Heroku : problème sur la partie *Randomizing tool*
- Deuxième déploiement sur Heroku : code refactorisé et fonctionnel (pas encore testé avec ajout des boutons pour calculer le pourcentage de prédictions correctes)

Structure du dossier pour le déploiement

Fichiers du déploiement

 app.py	09/12/2021 19:55	Fichier source Pyt...	3 Ko
 first_model.h5	03/12/2021 10:53	Fichier H5	1 834 Ko
 Procfile	08/12/2021 16:01	Fichier	1 Ko
 requirements.txt	08/12/2021 16:23	Document texte	1 Ko
 setup.sh	08/12/2021 16:00	Fichier source SH	1 Ko

Ajout du fichier
requirements.txt

Ajout du fichier
Procfile

Structure du dossier pour le déploiement

requirements.txt

```
requirements.txt
1  matplotlib==3.3.4
2  numpy==1.21.4
3  pandas==1.2.4
4  Pillow==8.4.0
5  streamlit==1.2.0
6  streamlit_drawable_canvas==0.8.0
7  tensorflow-cpu==2.7.0
```

Ajout du fichier
requirements.txt

Procfile

```
Procfile
1  web: sh setup.sh && streamlit run app.py
```

Ajout du fichier
Procfile

Démonstration

Version Streamlit:

<https://share.streamlit.io/cycylagneau/cnn-project/main/app.py>

Version Heroku:

<https://digitcnnproject.herokuapp.com/>