Identifier des chiffres

Problématique :

Déployer une application qui reconnaît des chiffres entre 0 et 9.

→ Utilisation d'un CNN pour prédire une image de chiffre donnée en entrée

→ Déploiement de l'application avec Heroku

Préparation des données

- 42000 images de chiffres de dimensions 28x28 au format flatten
- Target catégorisée : 10 catégories (0 à 9)
- Dataset balancé

- Données mises à l'échelle
- Données mise sous forme de tensor pour l'entrée dans le CNN

Modèle :

```
model = Sequential()
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu', input_shape=(28, 28, 1)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(32, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(20, activation='relu'))
model.add(Dense(df['label'].nunique(), activation="softmax"))
```

- Input : (28,28,1)

- 2 couches de convolution : Conv2D
  - 64 & 32 neurones
  - kernel (3,3)
  - activation : relu
  - pas de padding

- 2 couches de pooling : MaxPooling2D
  - taille (2,2)

- 2 couche dense
  - 20 & 10 neurones
  - activation : softmax

Modèle :

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 64)	640
max_pooling2d (MaxPooling2D)	(None, 13, 13, 64)	0
conv2d_1 (Conv2D)	(None, 11, 11, 32)	18464
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 32)	0
flatten (Flatten)	(None, 800)	0
dense (Dense)	(None, 20)	16020
dense_1 (Dense)	(None, 10)	210

Total params: 35,334
Trainable params: 35,334
Non-trainable params: 0

Modèle :

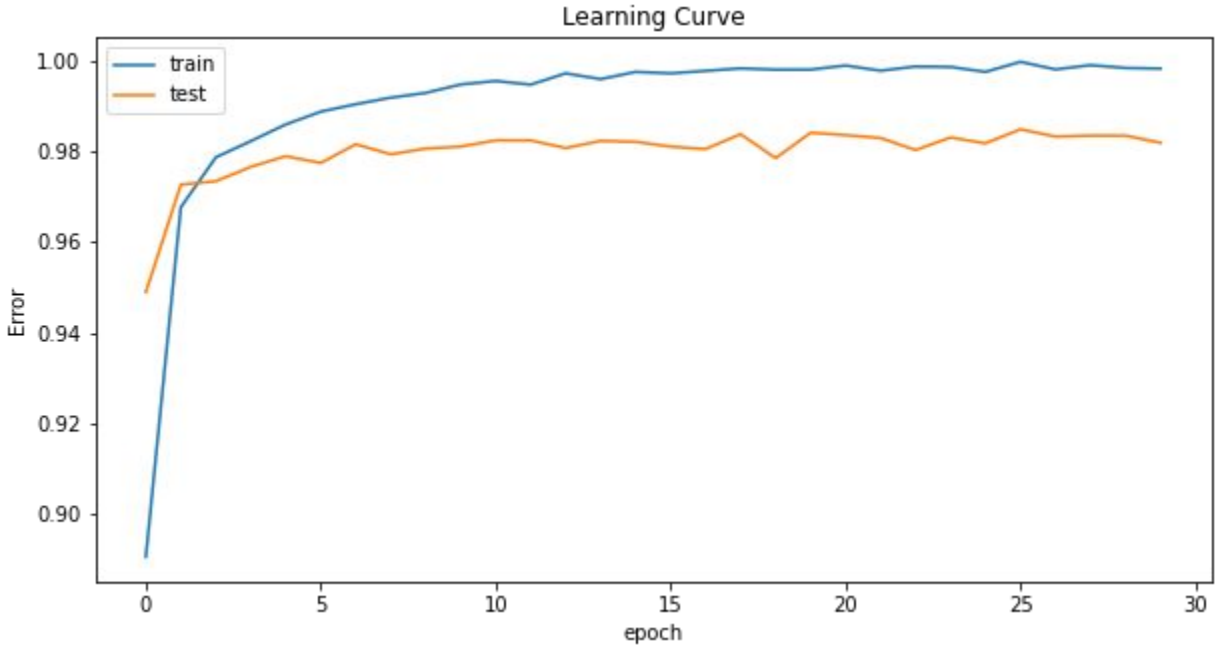
```
model.compile(optimizer="adam", loss='categorical_crossentropy', metrics=['accuracy'])
```

```
model.fit(X_train, y_train, batch_size=32, epochs=30, validation_split=0.3)
```

- optimizer : adam
- loss : categorical_crossentropy
- metric : accuracy

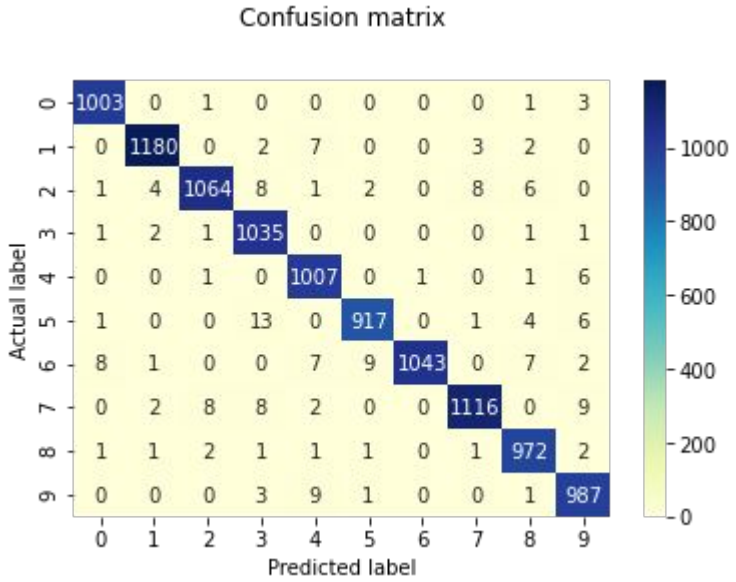
- batch_size : 32
- epoch = 30

Learning Curve



Classification Report & Confusion Matrix

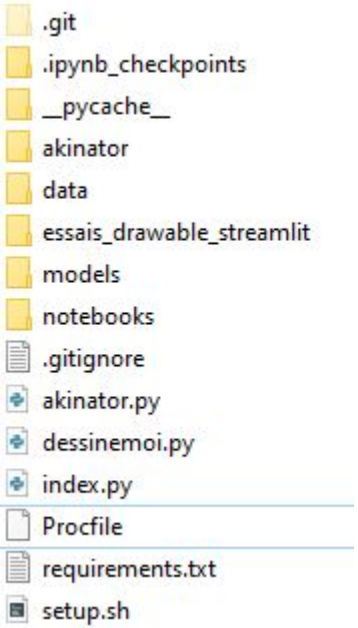
	precision	recall	f1-score	support
0	1.00	0.99	0.99	1015
1	0.99	0.99	0.99	1190
2	0.97	0.99	0.98	1077
3	0.99	0.97	0.98	1070
4	0.99	0.97	0.98	1034
5	0.97	0.99	0.98	930
6	0.97	1.00	0.98	1044
7	0.97	0.99	0.98	1129
8	0.99	0.98	0.98	995
9	0.99	0.97	0.98	1016
accuracy			0.98	10500
macro avg	0.98	0.98	0.98	10500
weighted avg	0.98	0.98	0.98	10500



Axes d’amélioration du modèle ?

- Early-stopping
- Dropout
- Data augmentation
- Padding
- Tester d’autres optimizer
- fonctions de regularization
- …

Architecture de l'application



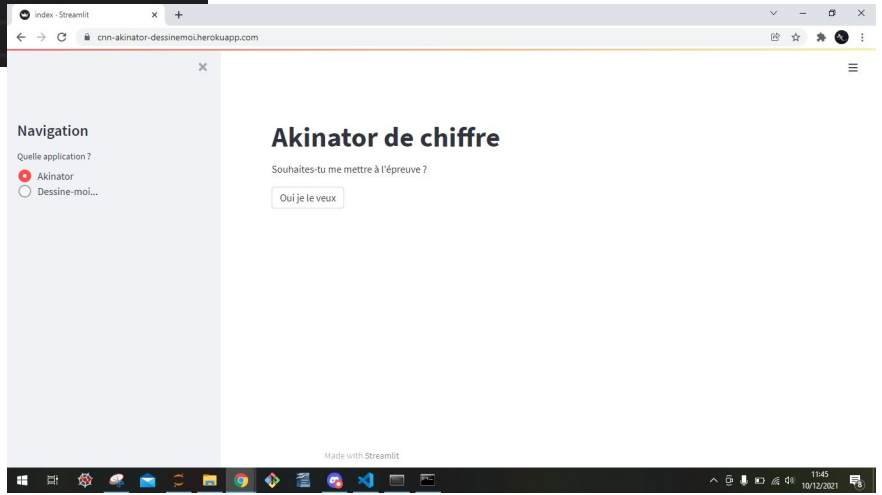
- Dossiers non push sur Heroku :

→ .ipynb_checkpoints, __pycache__, akinator, essais_drawable_streamlit & notebooks

- **index.py** apelle les 2 fichiers d'application : akinator.py & dessinemoi.py
- **akinator.py** : application qui prédit un chiffre à partir d'une image aléatoire d'un dataset
- **dessinemoi.py** :application qui identifie un chiffre dessiné par l'utilisateur

Accueil de l'application

```python
#index.py
import akinator
import dessinemoi
import streamlit as st

#LAYOUT
st.set_page_config(layout="wide")

PAGES = {
    "Akinator": akinator,
    "Dessine-moi...": dessinemoi
}
st.sidebar.title('Navigation')
selection = st.sidebar.radio("Quelle application ?", list(PAGES.keys()))
page = PAGES[selection]
page.app()
```



RQ :
pour déploiement final, enlever la barre de développement

Reconnaissance de chiffre

```python
# preprocessing des inputs
def preprocess(input):
    input_preprocess = input / 255
    input_preprocess = np.array(input).reshape((-1, 28, 28, 1))
    return input_preprocess

# afficher l'image en input

def see_img(input):
    image = np.array(input).reshape([28,28])
    fig, ax = plt.subplots()
    ax.imshow(image, cmap=plt.get_cmap('gray'))
    st.pyplot(fig)
```

```python
def my_prediction():

    # choisir une ligne au hasard dans le dataframe test

    my_input = csv_dowloaded.sample(n=1)

# afficher l'image en input

    see_img(my_input)

# afficher la sortie cad le numero predit : print(argmax)
    my_predict = np.argmax(reconstructed_model.predict(preprocess(my_input)), axis=1)
    return str(my_predict)[1]

st.write("Souhaites-tu me mettre à l'épreuve ? ")

if st.button('Oui je le veux'):
    st.write("Je vois, je vois ... ")
    st.write("Je vois un "+my_prediction())
```

Fonction d'évaluation du modèle

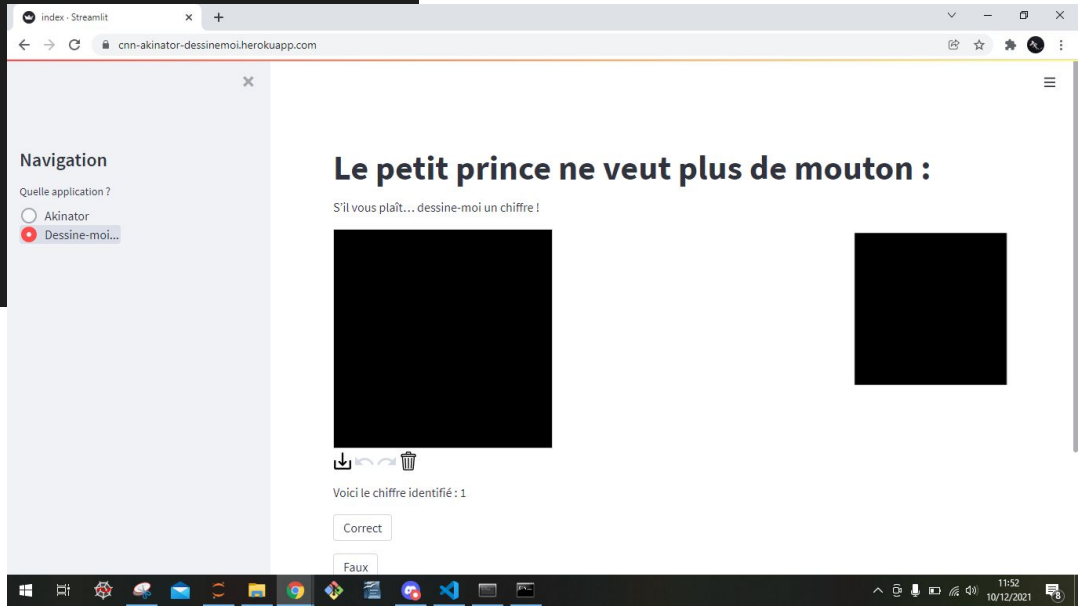
```
if 'juste' not in st.session_state:
    st.session_state.juste = 0

if 'faux' not in st.session_state:
    st.session_state.faux = 0

if 'tentatives' not in st.session_state:
    st.session_state.tentatives = 0

def raison():
    st.session_state.juste += 1
    st.session_state.tentatives += 1

def tort():
    st.session_state.faux += 1
    st.session_state.tentatives += 1
```

```
try :
    def justesse():
        justesse = st.session_state.juste/st.session_state.tentatives*100
        return justesse

    st.write("Akinator prédit juste à "+str(justesse())+"%")
except :
    st.write()
```

```
if st.button('Oui je le veux'):
    st.write("Je vois, je vois ... ")
    st.write("Je vois un "+my_prediction())
    st.button('Tu as raison Akinator',key='raison', on_click=raison)
    st.button('Tu as tort Akinator', key="tort", on_click=tort)
```

Front-end & fonction de dessin

```
c1, c2 = st.columns(( 3, 1))

# Create a canvas component
with c1 :
    canvas_result = st_canvas(
        fill_color="rgba(255, 165, 0, 0.3)",  # Fixed fill color with some opacity
        stroke_width=30,
        stroke_color="#FFFFFF",
        background_color="#000000",
        update_streamlit=True,
        width=280,
        height=280,
        drawing_mode="freedraw",
        key="canvas",
    )
```




```
res = Image.fromarray(canvas_result.image_data.astype('uint8'),'RGBA')
res = res.resize((28, 28))
res = np.array(res)
```