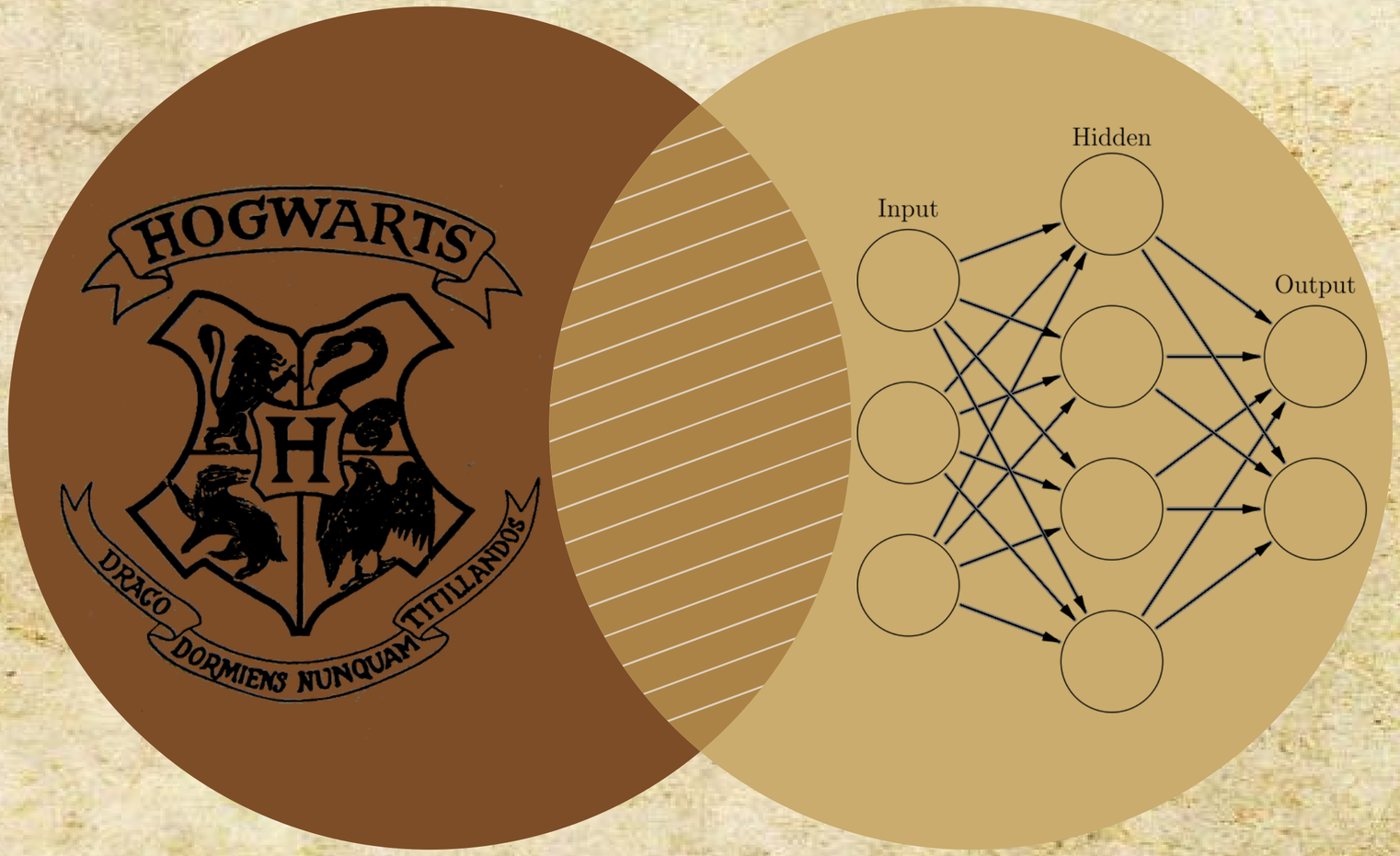
Sorting Hat

A MAGICAL NEURAL NETWORK PROJECT

CANDELA GARCÍA FERNÁNDEZ

WHAT PROJECT COULD I DO? BRAINSTORMING

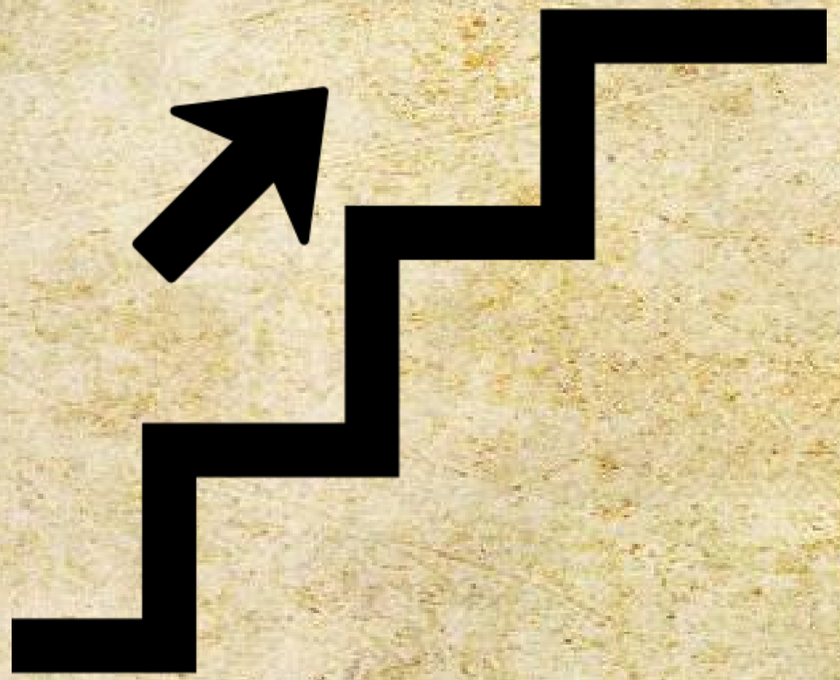
MAYBE...


HOGWARTS
DRACO DORMIENS NUNQUAM TITILLANDOS
Input
Hidden
Output

THE CHOSEN ONE: SORTING HAT
(IMAGE CLASSIFICATION USING NEURAL NETWORKS)

STEP BY STEP:

- Collecting data
- Preprocessing data
- Cleaning data
- Splitting data
- Choosing model
- Training model
- Testing model
- Creating GUI



COLLECTING DATA

Web Scraping & Manual Search

```
import requests
from bs4 import BeautifulSoup
import re
import pandas as pd

from urllib.request import urlopen

gryf = urlopen("https://harrypotter.fandom.com/wiki/Category:Gryffindors")
soup = BeautifulSoup(gryf)
images_gryf = soup.find_all('img', {'data-src':re.compile('.jpg')})
print(*map(lambda x: x['data-src'],images_gryf), sep='\n')
```

PREPROCESSING DATA

```
houses = [
    "Gryffindor",
    "Hufflepuff",
    "Slytherin",
    "Ravenclaw"
]

path = "C:/Users/cande/Desktop/Sorting-Hat/Houses/"

for house in houses:
    count = 0
    for img in glob.glob(path+house+"/*"):
        try:
            img = img.replace("\\","/")
            img = cv2.imread(img,0)
            img2= img.copy()
            faces =  faceClassifier.detectMultiScale(img2)

            for (x,y,w,h) in faces:
                cv2.rectangle(img2,(x,y),(x+w,y+h),(128.0,255),2)
                rostro = img2[y:y+h,x:x+w]
                rostro = cv2.resize(rostro,(64,64), interpolation=cv2.INTER_CUBIC)
                cv2.imwrite("{}_{}.jpg".format(house,count),rostro)
                count = count+1
                if count % 10 == 0:
                    print(count)

                cv2.imshow("rostro",rostro)
    #             cv2.imshow("image",img2)
                cv2.waitKey(1)
        except AttributeError:
            print(f"da error {img}")
```

– Creating houses

– Processing image color & size

– Locating faces

– Saving faces

– Splitting by house

CLEANING DATA:

Wrongly identified faces:

- Hair
- Objects
- Clothes
- Wrongly cut faces
- Duplicated faces

SPLITTING DATA

– Organizing faces by house

– Selecting my split size (training data, testing data, validation data)

– Using splitfolders

```
import splitfolders

input_folder = "Faces_by_house"
output = "Faces_by_house_processed"
splitfolders.ratio(input_folder,output, seed=42, ratio=(.8,.1,.1))
```

```
Copying files: 392 files [00:02, 155.92 files/s]
```

```
help(splitfolders.ratio)
```

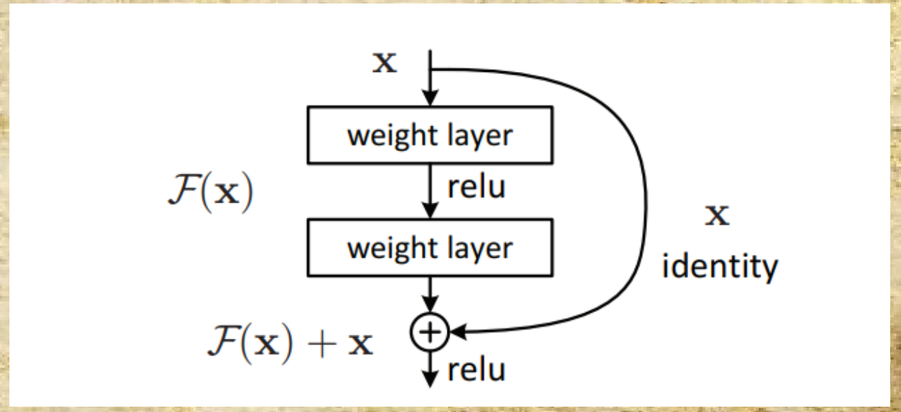
```
Help on function ratio in module splitfolders.split:

ratio(input, output='output', seed=1337, ratio=(0.8, 0.1, 0.1), group_prefix=None)
```

CHOOSING MY MODEL:

Failed models:
– Model from AutoEncoder class
– CNN (Convolutional Neural Network
– HouseClassifier (based in a Rock Paper Scissor Classifier)

Final ~~countdown~~ model:

– Residual Neural Network



TRAINING MODEL:

```
base_model = ResNet50(include_top=False,weights="imagenet")
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation="relu")(x)
predictions = Dense(train_generator.num_classes, activation="softmax")(x)
model = Model(inputs=base_model.input, outputs=predictions)

for layer in base_model.layers:
    layer.trainable=False
model.compile(optimizer="adam",loss="categorical_crossentropy", metrics = ["accuracy"])
model.fit(train_generator, epochs=10)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94773248/94765736 [==============================] - 7s 0us/step
Epoch 1/10
6/6 [==============================] - 5s 275ms/step - loss: 9.5888 - accuracy: 0.2936
Epoch 2/10
6/6 [==============================] - 1s 186ms/step - loss: 4.4531 - accuracy: 0.3414
Epoch 3/10
6/6 [==============================] - 1s 188ms/step - loss: 3.7800 - accuracy: 0.4390
Epoch 4/10
6/6 [==============================] - 1s 197ms/step - loss: 1.5590 - accuracy: 0.4283
Epoch 5/10
6/6 [==============================] - 1s 197ms/step - loss: 1.0490 - accuracy: 0.5494
Epoch 6/10
6/6 [==============================] - 1s 194ms/step - loss: 0.9420 - accuracy: 0.6375
Epoch 7/10
6/6 [==============================] - 1s 188ms/step - loss: 0.8302 - accuracy: 0.7085
Epoch 8/10
6/6 [==============================] - 1s 191ms/step - loss: 0.7945 - accuracy: 0.6780
Epoch 9/10
6/6 [==============================] - 1s 193ms/step - loss: 0.8063 - accuracy: 0.6775
Epoch 10/10
6/6 [==============================] - 1s 189ms/step - loss: 0.6724 - accuracy: 0.7885

<tensorflow.python.keras.callbacks.History at 0x2678a22fb80>
```

NVIDIA

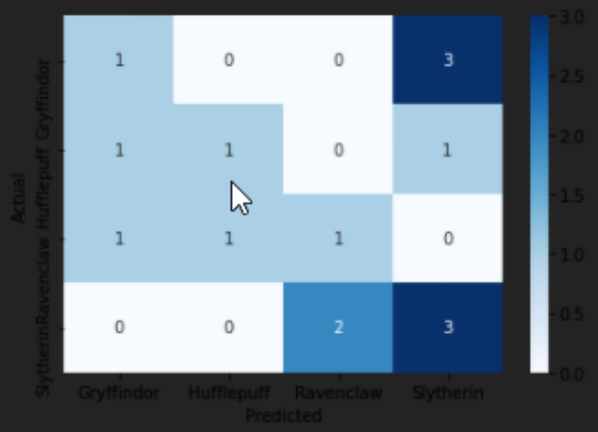
TESTING MODEL:

```
model=tf.keras.models.load_model("Model/ResNet50_faces.h5")
filenames = test_generator.filenames
nb_samples=len(test_generator)
y_prob=[]
y_act=[]
test_generator.reset()
for _ in range(nb_samples):
    X_test,Y_test=test_generator.next()
    y_prob.append(model.predict(X_test))
    y_act.append(Y_test)
predicted_class = [list(train_generator.class_indices.keys())[i.argmax()] for i in y_prob]
actual_class = [list(train_generator.class_indices.keys())[i.argmax()] for i in y_act]
```

TESTING MODEL:

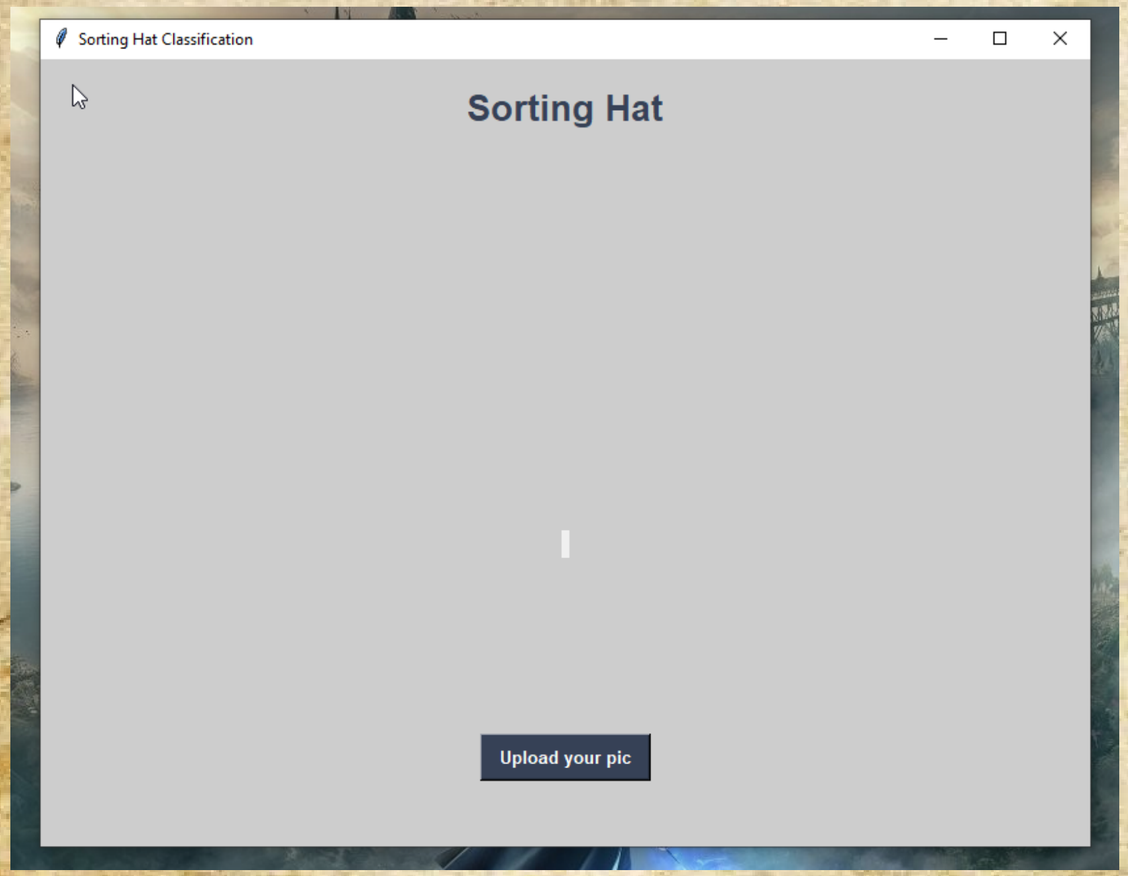
```
out_df = pd.DataFrame(np.vstack([predicted_class,actual_class]).T,
                      columns=["predicted_class","actual_class"])
confusion_matrix=pd.crosstab(out_df["actual_class"], out_df["predicted_class"],
                             rownames=["Actual"], colnames=["Predicted"])
sns.heatmap(confusion_matrix, cmap="Blues", annot=True, fmt="d")
plt.show
print("Test accuracy:{}".format((np.diagonal(confusion_matrix).sum()/confusion_matrix.sum().sum()*100)))
```

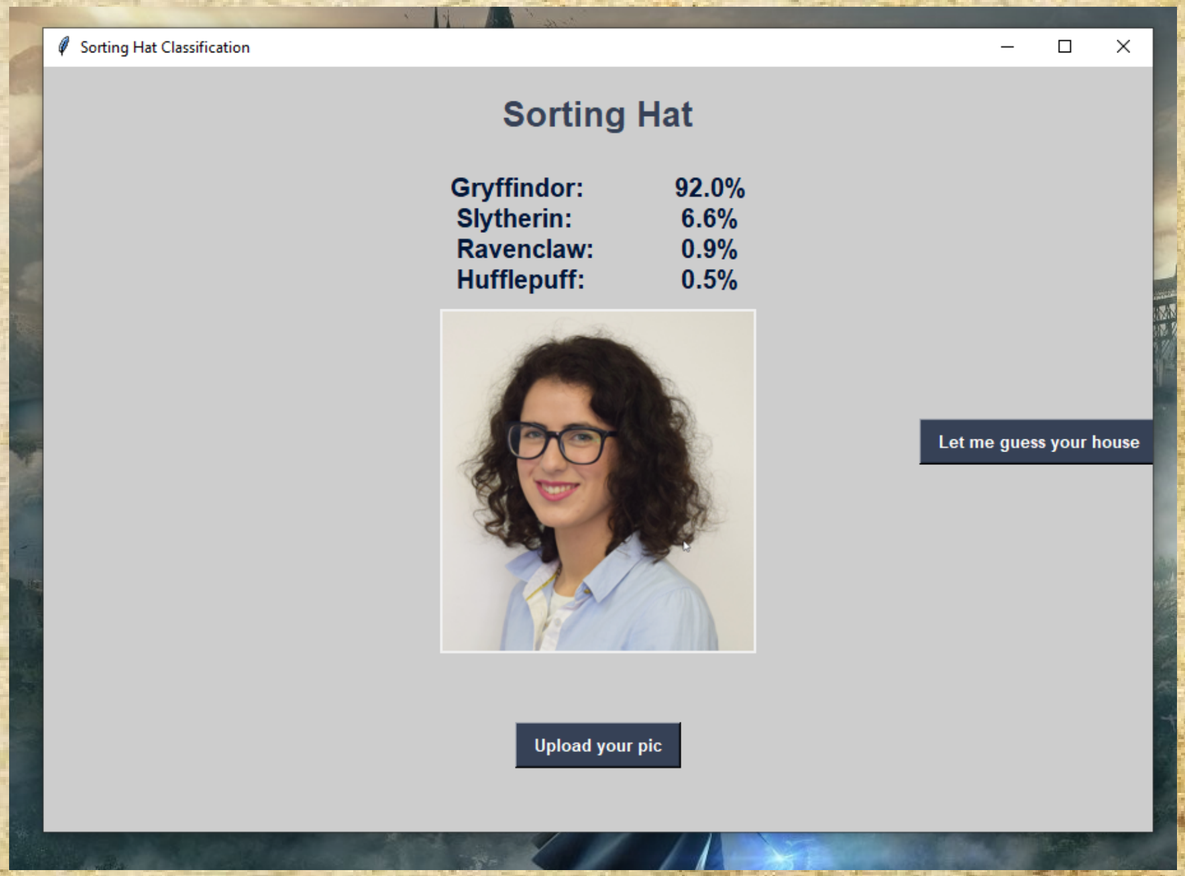
```
Test accuracy:40.0
```



CREATING GUI

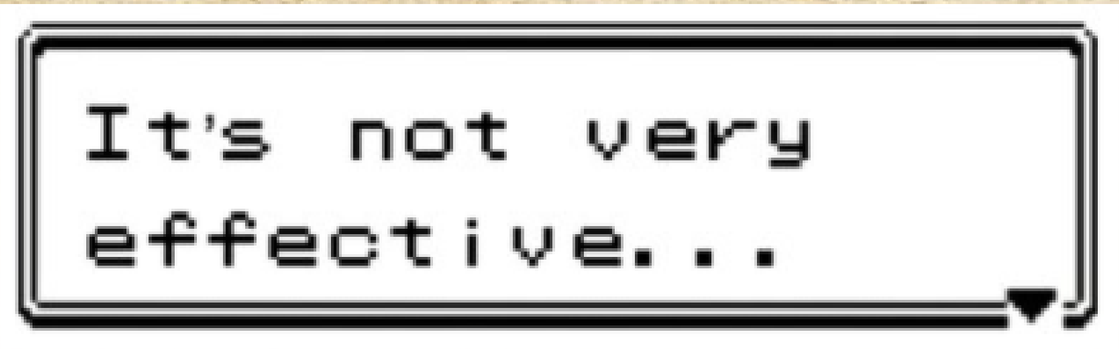
GRAPHICAL USER INTERFACE





WHAT'S NEXT?

– Increasing my dataset 'cause now...



– Improving my GUI

QUESTIONS?