Intelligent Sorting Hat

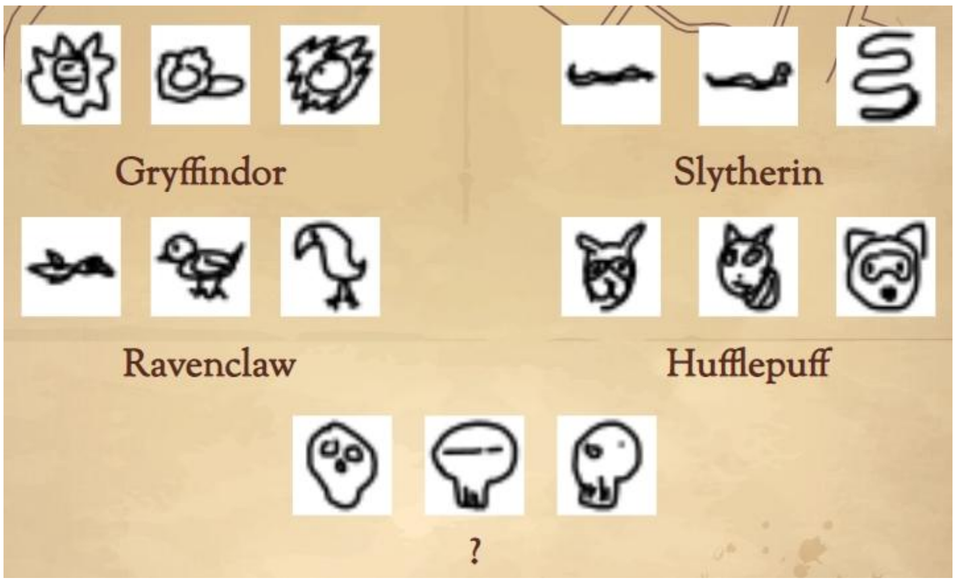
Team 21
0716081 葉晨 / 0716083 羅子涵

1. Problem

Our problem is trying to make an intelligence sorting hat. What we want to achieve is given a drawing, the sorting hat will give a house to the new student according to the drawing.

The dataset we use is from a Google's game called Quick draw. There are millions of drawings from people who played this game. Because some of the symbolic animals of the academies are difficult to draw, we chose ten animals easy to draw to train the model. We make lions and tigers for Gryffindor, snakes and snails for Slytherin, birds, owls and parrots for Ravenclaw, and raccoons and squirrels for Hufflepuff. The last category, skulls, for a mystery house.

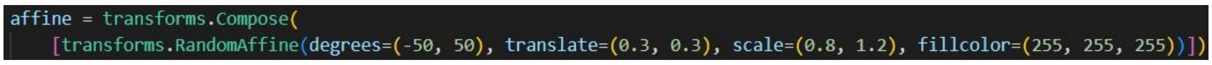
Here are some examples of each house.



2. Data Preprocessing

The reason to do the data deformation is that we want to expand the training dataset, and use them to train several CNNs. In our project, we make two more training datasets additionally to train 3 CNNs.

We use the function from transforms, which is random affine, to do the data deformation. It can do rotation, horizontal and vertical translation, and scaling at the same time. The function will randomly choose the coefficients of these affine transformations in the given ranges.



```
affine = transforms.Compose(
    [transforms.RandomAffine(degrees=(-50, 50), translate=(0.3, 0.3), scale=(0.8, 1.2), fillcolor=(255, 255, 255))])
```

Here are some examples before and after affine transformation.

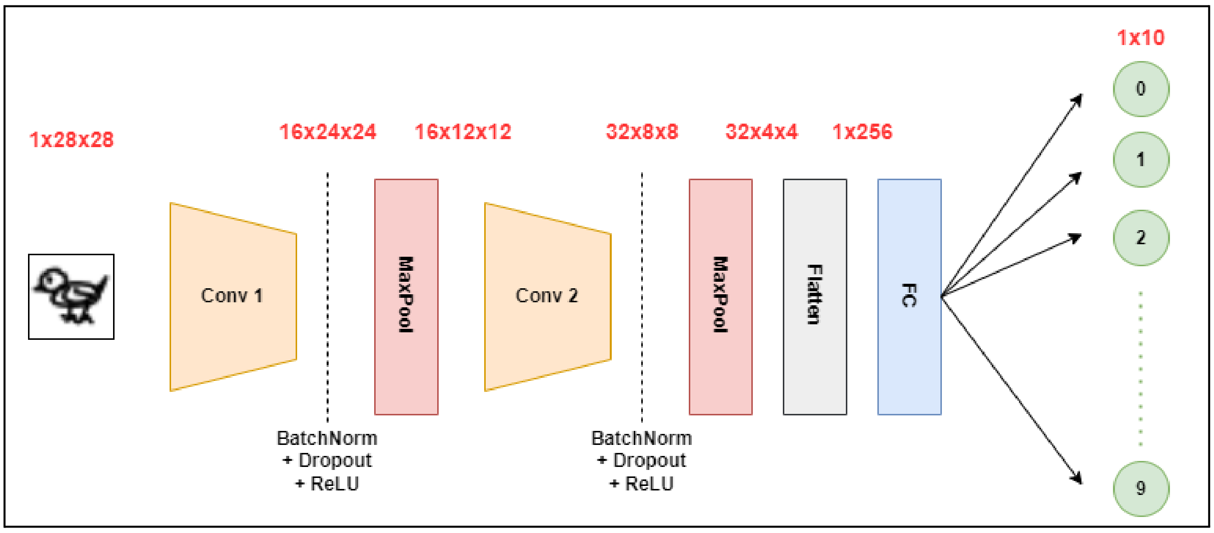
Before
After

3. Models

We tried three different models, CNN, DNN, and ResNet-18 for classification. And we found that CNN got the best result, so we chose **CNN** for our model.

And below is the structure of the CNN, the input is a 1*28*28 image, finally we got 1*10 vector, which represents the probability of each class.

Through this network, our validation accuracy can reach 85.44%.

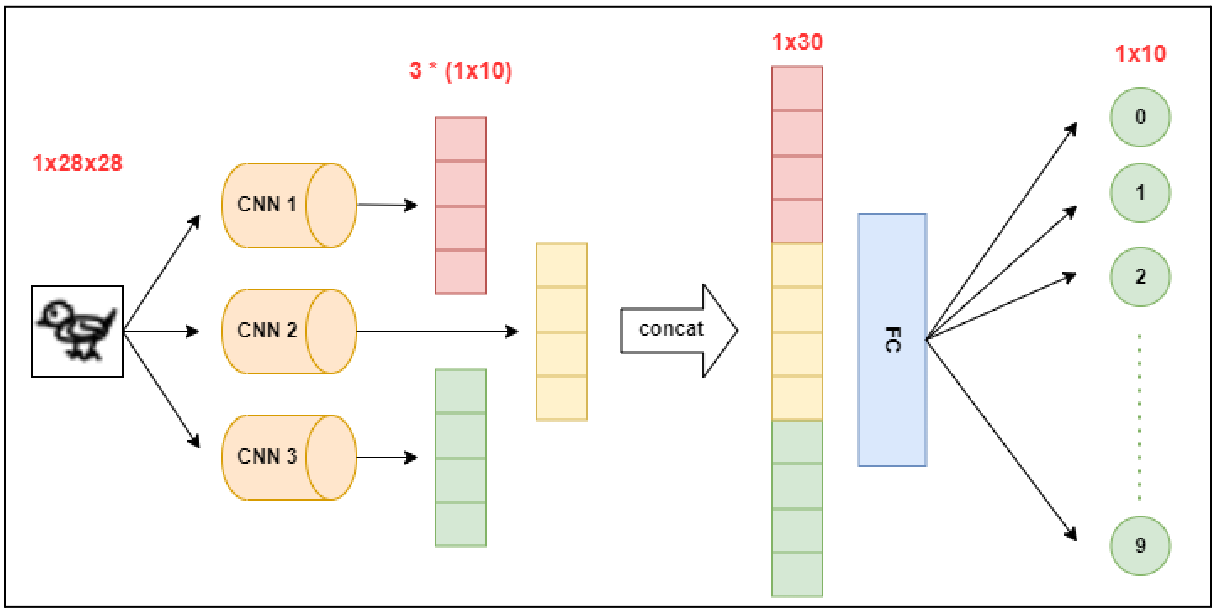


But we weren't satisfied with the result, so we used the skill of the **ensemble** to try to improve the performance again.

So Here we use three different datasets (generated from data deformation) to **train three CNNs** with the same structure. After getting the three 1*10 output vectors of each CNN, we concatenate them into a 1*30 vector, and finally pass through a FC layer and get the final result vector.

Using this method, we can consider three results of different CNNs, so the performance can be improved again.

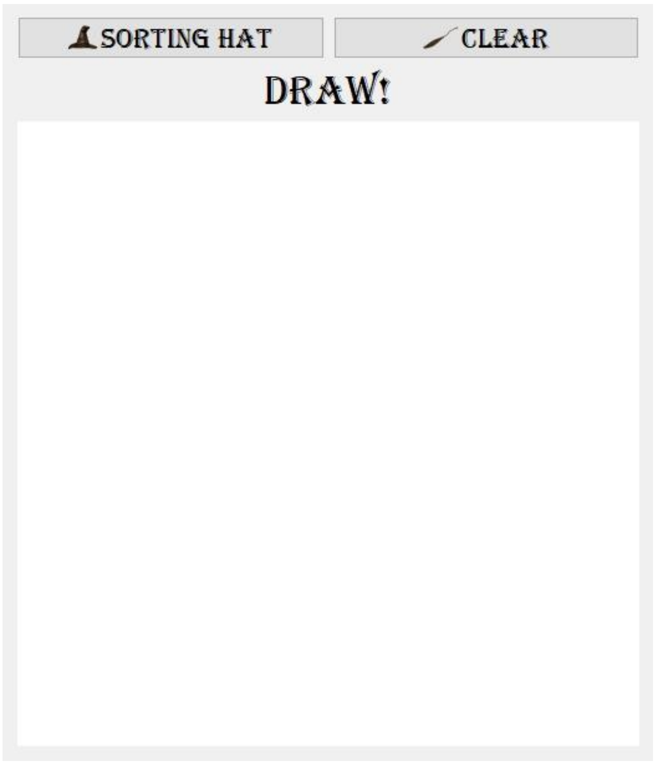
Due to our effort, we finally got accuracy up to 88.94%!

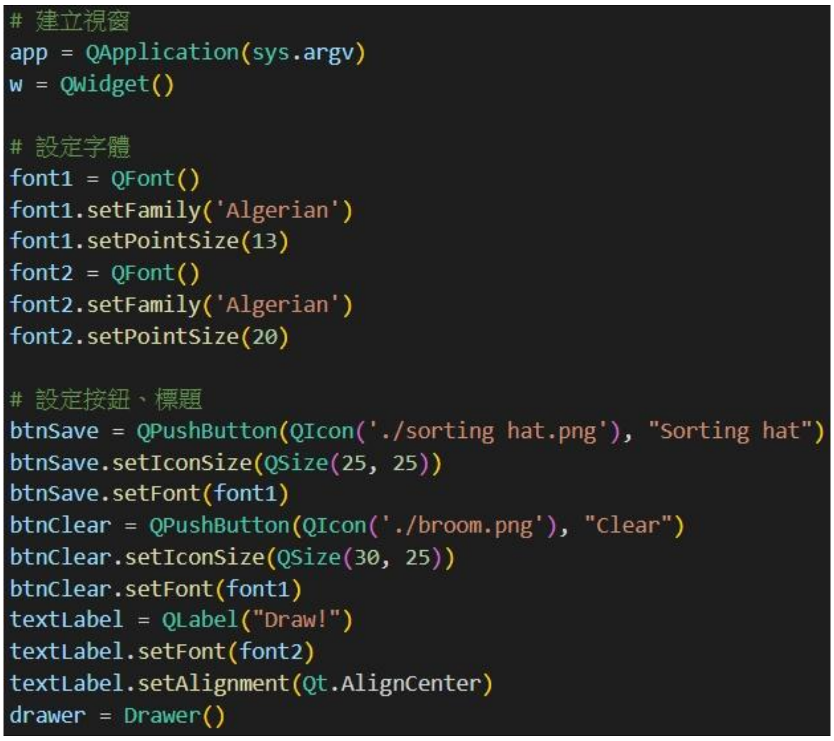


4. Front-end using PyQt5

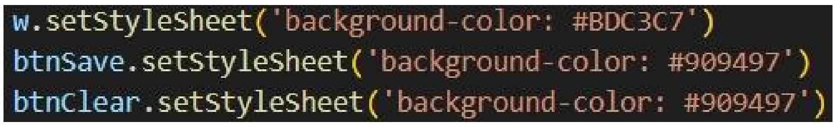
The package we used to construct the drawer is called PyQt5. It can be used to design the GUI.

First is the appearance. We use several functions listed here to construct the drawer. QWidget() for the window, QPushButton() for the buttons above, QLabel() for the text DRAW, QIcon() for the little icons in the buttons, QFont() for the cool font, and setStyleSheet() for the colors of the window and the buttons.





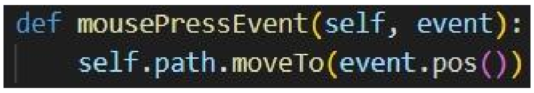
```python
# 建立視窗
app = QApplication(sys.argv)
w = QWidget()

# 設定字體
font1 = QFont()
font1.setFamily('Algerian')
font1.setPointSize(13)
font2 = QFont()
font2.setFamily('Algerian')
font2.setPointSize(20)

# 設定按鈕、標題
btnSave = QPushButton(QIcon('./sorting hat.png'), "Sorting hat")
btnSave.setIconSize(QSize(25, 25))
btnSave.setFont(font1)
btnClear = QPushButton(QIcon('./broom.png'), "Clear")
btnClear.setIconSize(QSize(30, 25))
btnClear.setFont(font1)
textLabel = QLabel("Draw!")
textLabel.setFont(font2)
textLabel.setAlignment(Qt.AlignCenter)
drawer = Drawer()
```



```python
w.setStyleSheet('background-color: #BDC3C7')
btnSave.setStyleSheet('background-color: #909497')
btnClear.setStyleSheet('background-color: #909497')
```

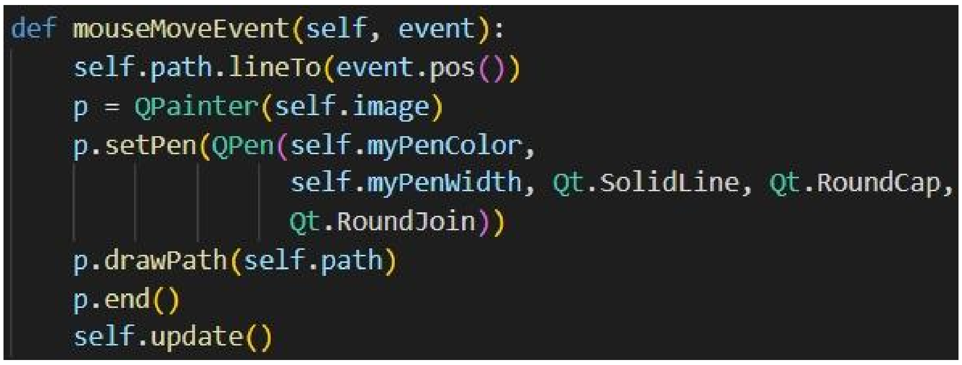
The second part is the drawer. We build a class and set all the operations it needs. The process of using the drawer is listed below.

First, it can detect the mouse press event, which means a start of path drawing. We use moveTo() to record the starting position.



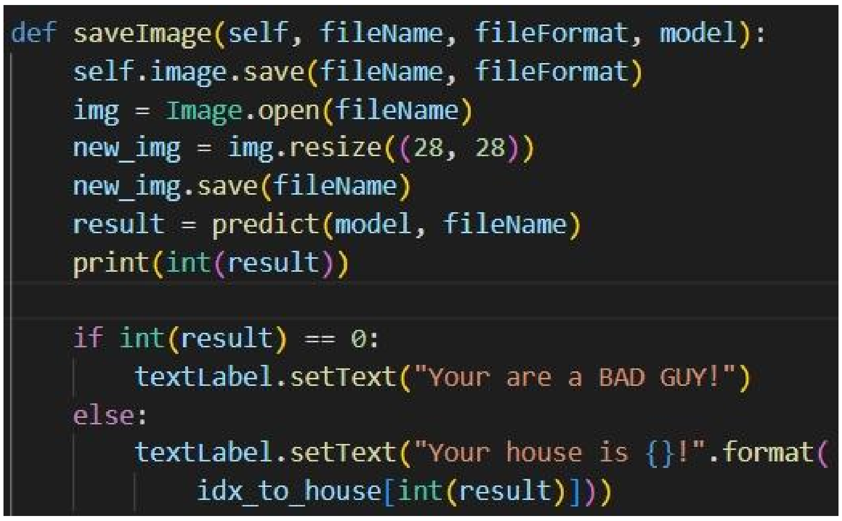
```python
def mousePressEvent(self, event):
    self.path.moveTo(event.pos())
```

When the mouse is moving around and released, the path will be saved. The first two steps can be repeated in a drawing.



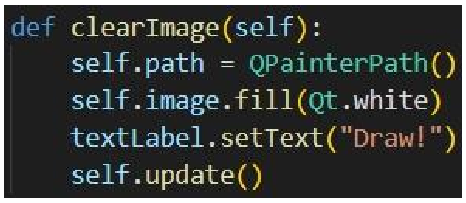
```python
def mouseMoveEvent(self, event):
    self.path.lineTo(event.pos())
    p = QPainter(self.image)
    p.setPen(QPen(self.myPenColor,
                  self.myPenWidth, Qt.SolidLine, Qt.RoundCap,
                  Qt.RoundJoin))
    p.drawPath(self.path)
    p.end()
    self.update()
```

When we finish the drawing, we press the button SORTING HAT, then the image will be saved, put into the model and the prediction will be returned. The text will change according to the prediction result.



```python
def saveImage(self, fileName, fileFormat, model):
    self.image.save(fileName, fileFormat)
    img = Image.open(fileName)
    new_img = img.resize((28, 28))
    new_img.save(fileName)
    result = predict(model, fileName)
    print(int(result))

    if int(result) == 0:
        textLabel.setText("Your are a BAD GUY!")
    else:
        textLabel.setText("Your house is {}!".format(
            idx_to_house[int(result)]))
```

Finally, if we want to draw a new picture, we just press the button CLEAR. It will cover the drawer with white color.



```python
def clearImage(self):
    self.path = QPainterPath()
    self.image.fill(Qt.white)
    textLabel.setText("Draw!")
    self.update()
```

5. Contribution

Data preprocessing / Front-end design : 羅子涵
Model construction : 葉晨

6. Reference

- A Riddikulus Dataset. Building a Harry Potter dataset for… | by Gant Laborde | Google Developers Experts | Medium
- 29 VV Romanuke TRAINING DATA EXPANSION AND BOOSTING OF CONVOLUTIONAL NEURAL NETWORKS FOR REDUCING THE MNIST DATASET ERROR
- pyqt5 i save the image but it is always empty - Stack Overflow
- https://blog.csdn.net/weixin_42066185/article/details/82225197
- https://zhuanlan.zhihu.com/p/28559136
- https://blog.csdn.net/marwenx/article/details/116064384