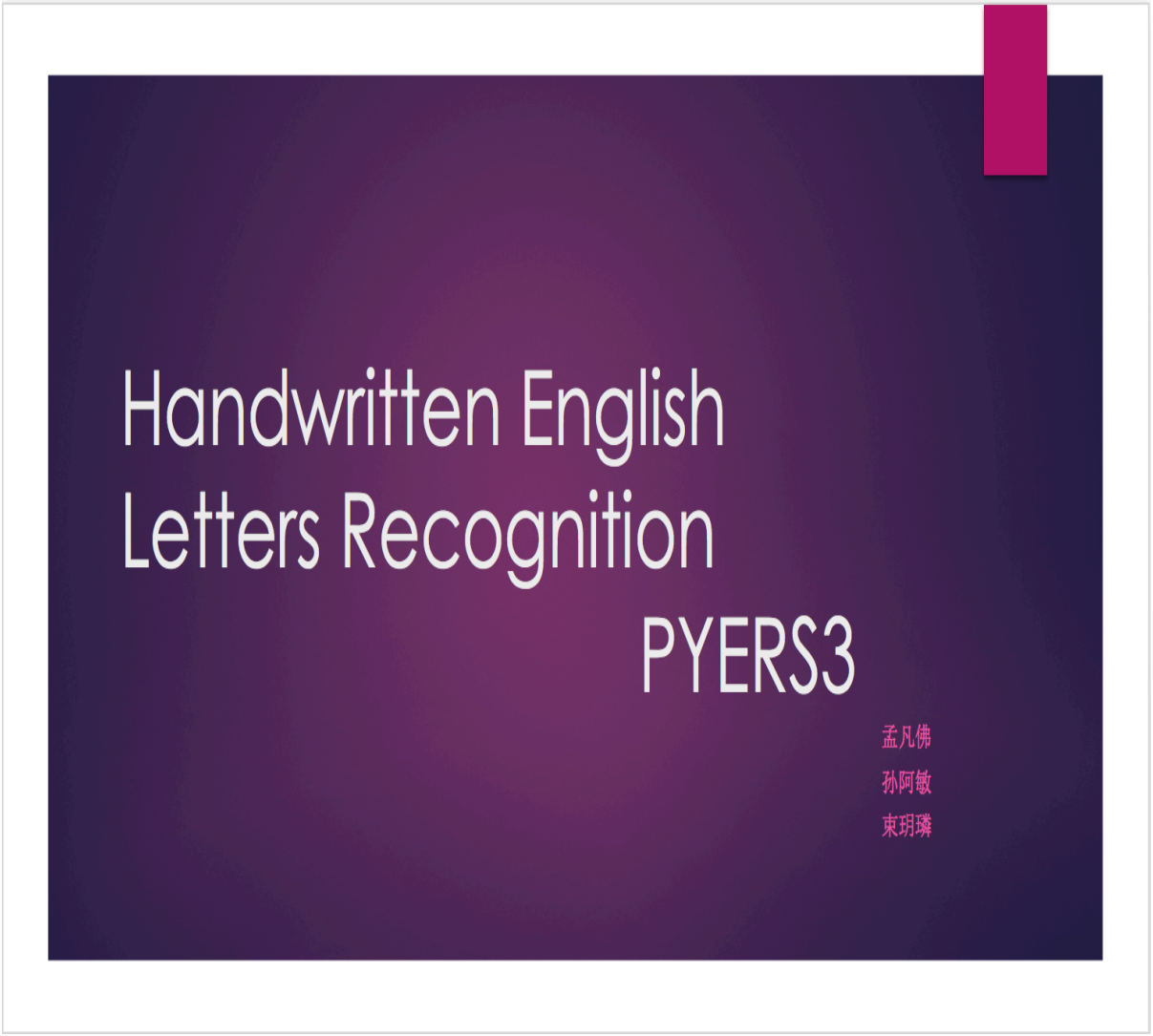
Team Project Report

Project name: **Wonder Words——Handwriting English Letters Recognition**

Team name: Pyers3

Work allocation: 孟凡佛（codes and report）

孙阿敏、束玥璘（PPT）



1. INTRODUCTION

We intend to create a small program called Wonder Words. Wonder Words can identify printed or clear handwritten English letters.

Handwriting recognition has a long history and is widely used in modern society. In our daily life, we often run into situation where we need to convert printed or handwritten paper into text files that can be easily edited by computers. Therefore, we think that it's useful and fun to make a small program that can recognize handwritten letters by applying what we have learned.

2. CHALLENGES

The biggest challenge we met during this project is understanding deep learning. Since deep learning is such a big and complicated subject, it took us nearly 3 weeks to get a brief overview of its mechanism and application. We read a lot of online articles about deep learning and neural networks, for example the article <A quick introduction to neural network>. We also viewed many tutorials and introduction videos on YouTube. The series on neural network by 3blue1brown was particularly helpful.

Besides understanding deep learning, finding and processing training data was another big challenge. Finding data on handwritten English letters is not as easy as finding data on handwritten digits. We looked up in the UCI Machine Learning Repository, GitHub, Kaggle. Finally, we narrowed down to 3 ideal training datasets. The first training dataset is a png picture set including 3000 handwritten letters. The second dataset is a csv file containing the greyscale information of nearly 370000 handwritten letters. The last dataset is a set of 74000 natural images

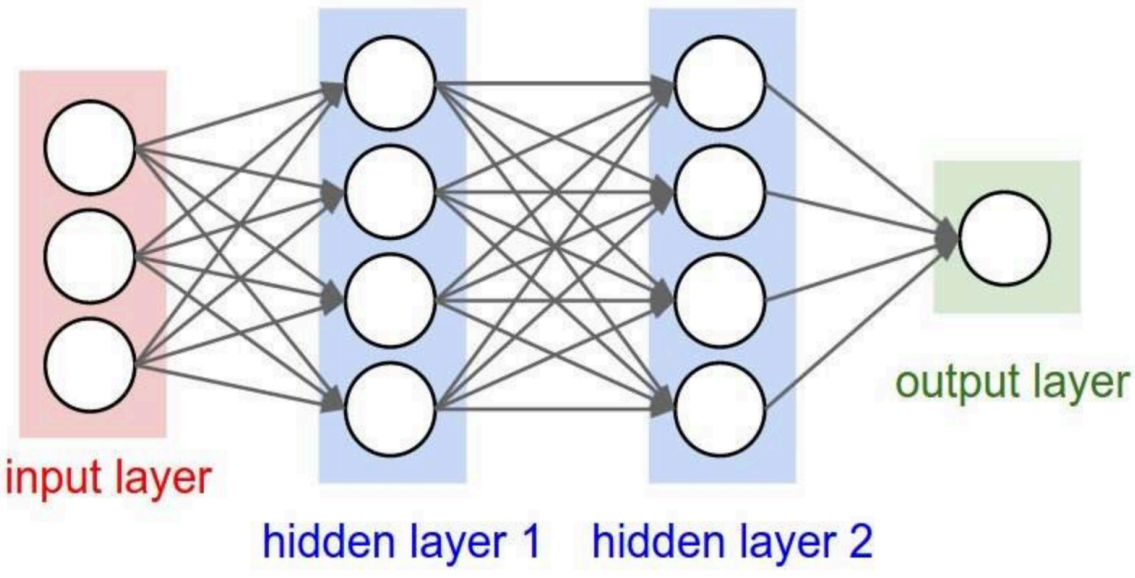
containing letters. In the end, we choose to train our model on the csv file. After selecting our dataset, it took us some time to figure out the structure and contents of our training data. The overview of the data is covered in the remaining part of the report.

3. METHODS & TECHNIQUES

Overall, we adopted deep learning and CNN model.

By definition, deep learning is a subset of machine learning in Artificial Intelligence that has networks capable of learning unsupervised data that is unstructured or unlabeled. Its algorithms are inspired by the structure and function of the brain called neural networks.

The basic idea behind deep learning as we understood is that you input some data into the computer and the computer will automatically look for the patterns and features and then give a output. Deep refers to that there're many hidden layers.



Convolutional Neural Networks are a category of neural network that have proven very effective in areas such as image recognition and classification.

We also used the popular Deep learning library Keras since it's relatively beginner-friendly.

4. CODES

#STEP 1 Install Keras

```
$ pip install keras
```

#STEP 2 Import libraries and models

```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras import backend as K
from keras.utils import np_utils
from sklearn.model_selection import train_test_split
import numpy as np
```

```
# seed for reproducing same results
seed = 785
np.random.seed(seed)
```

We import Sequential model. It is a linear stack of neural network layers which is useful. Next, we import Dense, Dropout, Flatten. They are the so-called "core" layers which are used in almost every CNN model. Then, we import the CNN layers to help train the data more efficiently. Also, we import some utilities to help us transform the data into the desired form. Finally, we import numpy and set a seed to reproduce the results.

#STEP 3 Load dataset and split input and output data

```
# load dataset
dataset = np.loadtxt('A_Z Handwritten Data.csv', delimiter=',')

# split into input and output variables
X = dataset[:,0:784]
Y = dataset[:,0]

(X_train, X_test, Y_train, Y_test) = train_test_split(X, Y, test_size=0.50, random_state=seed)
```

```
#Let's see the data structure
print(X.shape)
```

```
(372451, 784)
```

There are about 370000 observations in our csv file. The file contains the greyscale information of 28*28 pixels images. The outcome is labeled from 0 to 25 indicating lowercased “a” to “z”.

#STEP 4 Preprocessing input data for Keras

```
# reshape the data
X_train = X_train.reshape(X_train.shape[0], 28, 28, 1).astype('float32')
X_test = X_test.reshape(X_test.shape[0], 28, 28, 1).astype('float32')

# normalize data values to the range [0, 1]
X_train = X_train / 255
X_test = X_test / 255

Y_train = np_utils.to_categorical(Y_train)
Y_test = np_utils.to_categorical(Y_test)

num_classes = Y_test.shape[1]
```

We transformed our dataset to (n, depth, width, height) and converted our data type to **float32** and finally normalized our data values to the range [0, 1].

#STEP 5 Define model structure

```
# create model
model = Sequential()
model.add(Conv2D(32, (5, 5), input_shape=(28, 28, 1), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))
```

We only got a brief idea about what each layer is for without understanding the theory behind them. For example, the dropout layer is to prevent overfitting. MaxPooling2D is a way to reduce the number of parameters in our model. As for model parameters, we added one Convolution layer. To complete our model architecture, we added a fully connected layer and then the output layer.

#STEP 6 Compile model and fit model with training data

```
# Compile model and fit models on training datas
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(X_train, Y_train, validation_data=(X_test, Y_test), epochs=10, batch_size=200, verbose=2)
```

We compiled the model by declaring categorical_crossentropy as the loss function and adam as the optimizer. Then we chose the batch size and number of epochs to train for and trained the data.

#STEP 7 Evaluate and save the model

```
# Final evaluation of the model
scores = model.evaluate(X_test,Y_test, verbose=0)
print("CNN Error: %.2f%%" % (100-scores[1]*100))

model.save('weights.model')
```

#STEP 8 Let's predict some letters

```
from PIL import Image
s = "abcdefghijklmnopqrstuvwxyz"
img=Image.open('v.png')
img = np.array(img)
y_pred = model.predict_classes(img.reshape((1, 28, 28, 1)))
print("The Letter Is"+" "+s[y_pred[0]]+"!"+":)")
```

5. SUMMARY

Although we had taken quite a lot codes for reference, it was still quite a challenge to complete this project. But we felt amazing and proud when everything suddenly began to make sense after many days' of exploration and learning. This coding experience helped us to better understand what we have learned in class. It also enabled us to learn a lot more about machine learning. This small project is not anywhere near perfect. We hope that we can improve it in the future.

In the end, thanks for Mr. Baoyang's great help to our project.