Final Project - Individual report
Xi Qian

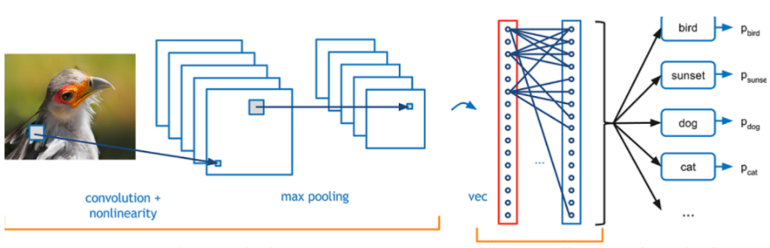
1. **Introduction. An overview of the project and an outline of shared network.**
   This project addresses an image classification problem, and further application of the image classification explored in class. The IAM Handwriting Dataset chosen for this project is comprised of a collection of handwritten passages by various writers. The goal of this project is to use deep learning to classify the writers by their writing styles. The results of this project can be used in future applications, such as identifying criminals by signature in fraudulent cases.

   We decided to use a Convolutional Neural Network (CNN) for this image classification problem. We wrote the code for the groupwork.py file. We did the architecture network design part on one of the 4 convolutional layer of the SGD one together, but each of us also designed a unique CNN by ourselves. We also decided the optimizer and parameters after discussion. The network I designed is the 4-Convolution-layer-Adam one. Our group met multiple times to write the final report, revise our code, compare the model performance, and create our presentation.

2. **Description of your individual work. Provide some background information on the development of the algorithm and include necessary equations and figures.**

   We worked on the 4-convolution-layer-SGD one together, all other things were done by ourselves individually, from the data pre-processing part, network training part. But we evaluated the model performance together.

   The reason why I chose CNN is because it is most commonly applied to analyzing visual imagery, and also because our goal of this project is to classify writers based on their writing styles.
   A CNN consists of an input and output layer, as well as multiple hidden layers. Usually, the hidden layers of a CNN typically consist of convolutional layers, pooling payers and fully connected layers. These are what I leaned from the ML 2 class, and they (knowledge) were applied when I design the 4 convolutional layers model, which is discussed in the third question.
   Here it is an example of how CNN works:



3. **Describe the portion of the work that you did on the project in detail. It can be figures, codes, explanation, pre-processing, training, etc.**

   We worked on the 4-convolution-layer-SGD one together, all other things were done by ourselves individually, from the data pre-processing part, network training part, to model performance evaluation part.

- Data Pre-Processing:

  I looked up the dataset and figured out what are these dashes connected name of the image files' meaning, and how many writers in the datasets in total. There are 657 writers in total. We decided to use only 50 writer's handwritten image to do classify in our projects, since otherwise it might be too complicated for the neural network we created. I made change to the image files and created a target list that has the writer number with respect to each image file in the

  I also investigated on the image files, and found out those image sizes are quite different, so I decided to breakdown its image size to a standard one. There is also a reference chunk of code we found online. It's a generator function that was used to scan through each sentence and generate random patches with same patch size. The final size of the cropped image is 113*113.

- Design of 4 layers of convolution neural network

  I designed the 4 layers of convolution neural network (Adam). Usually, the more number of convolutional layers, the better the information the model can capture, which leads to better prediction accuracy. The activation function I used is ReLU. The Maxpooling size I chose is 2*2 in all layers. The number of filters in each layer varies. In the first layer, I set up smaller number of filters, but I increased the number of filters in the later on layers so that the model can capture more detail information from the image. I also added dropout layer in the model, the dropout ratio is commonly set up to be 0.2-0.5, so that it can prevent the model to be overfitting.

- Training the network

  I split the dataset into 3 datasets. Train, validation and test set. The ratio is 7:1.5:1.5.
  The original plan in the training part was to change the number of samples in each epoch, the number of epoch and dropout rate so that I could compare the model performance.

4. **Results. Describe the results of your experiments, using figures and tables wherever possible. Include all results (including all figures and tables) in the main body of the report, not in appendices. Provide an explanation of each figure and table that you include. Your discussions in this section will be most important part of the report.**

Here is the summary of the model I designed:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
zero_padding2d_2 (ZeroPaddin (None, 115, 115, 1)       0         
_________________________________________________________________
lambda_2 (Lambda)            (None, 56, 56, 1)         0         
_________________________________________________________________
conv1 (Conv2D)               (None, 28, 28, 32)        320       
_________________________________________________________________
activation_8 (Activation)    (None, 28, 28, 32)        0         
_________________________________________________________________
pool1 (MaxPooling2D)         (None, 14, 14, 32)        0         
_________________________________________________________________
conv2 (Conv2D)               (None, 14, 14, 32)        9248      
_________________________________________________________________
activation_9 (Activation)    (None, 14, 14, 32)        0         
_________________________________________________________________
pool2 (MaxPooling2D)         (None, 7, 7, 32)          0         
_________________________________________________________________
conv3 (Conv2D)               (None, 7, 7, 64)          18496     
_________________________________________________________________
activation_10 (Activation)   (None, 7, 7, 64)          0         
_________________________________________________________________
pool3 (MaxPooling2D)         (None, 3, 3, 64)          0         
_________________________________________________________________
conv4 (Conv2D)               (None, 3, 3, 128)         73856     
_________________________________________________________________
activation_11 (Activation)   (None, 3, 3, 128)         0         
_________________________________________________________________
pool4 (MaxPooling2D)         (None, 1, 1, 128)         0         
_________________________________________________________________
flatten_2 (Flatten)          (None, 128)               0         
_________________________________________________________________
dropout_4 (Dropout)          (None, 128)               0         
_________________________________________________________________
dense1 (Dense)               (None, 512)               66048     
_________________________________________________________________
activation_12 (Activation)   (None, 512)               0         
_________________________________________________________________
dropout_5 (Dropout)          (None, 512)               0         
_________________________________________________________________
dense2 (Dense)               (None, 256)               131328    
_________________________________________________________________
activation_13 (Activation)   (None, 256)               0         
_________________________________________________________________
dropout_6 (Dropout)          (None, 256)               0         
_________________________________________________________________
output (Dense)               (None, 50)                12850     
_________________________________________________________________
activation_14 (Activation)   (None, 50)                0         
=================================================================
Total params: 312,146
Trainable params: 312,146
Non-trainable params: 0
```

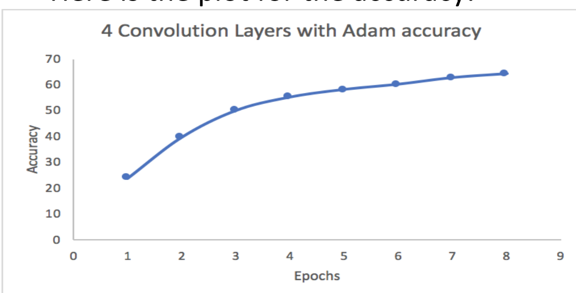
The running time of the 4-layer-network took longer time than 3 layers in average. The number of epochs I chose were 5 and 8. The number of samples I chose were 500 and 1000. Here is the result I got when I set the num of epochs=8 and num of sample=1000.

It shows that the final accuracy is around 65%, which is not so good. The loss did not change a lot after the second epochs, and the accuracy was approaching stable after the sixth of epochs.

```
Epoch 1/8
1000/1000 [==============================] - 2079s 2s/step - loss: 3.0565 - acc: 0.2387 - val_loss: 2.4752 - val_acc: 0.2715

Epoch 00001: saving model to check-01-2.4752.hdf5
Epoch 2/8
1000/1000 [==============================] - 1885s 2s/step - loss: 2.0169 - acc: 0.3965 - val_loss: 1.6918 - val_acc: 0.4886

Epoch 00002: saving model to check-02-1.6918.hdf5
Epoch 3/8
1000/1000 [==============================] - 1885s 2s/step - loss: 1.6261 - acc: 0.4993 - val_loss: 1.5528 - val_acc: 0.5243

Epoch 00003: saving model to check-03-1.5528.hdf5
Epoch 4/8
1000/1000 [==============================] - 1888s 2s/step - loss: 1.4435 - acc: 0.5510 - val_loss: 1.3610 - val_acc: 0.5760

Epoch 00004: saving model to check-04-1.3610.hdf5
Epoch 5/8
1000/1000 [==============================] - 1885s 2s/step - loss: 1.3455 - acc: 0.5802 - val_loss: 1.4090 - val_acc: 0.5680

Epoch 00005: saving model to check-05-1.4090.hdf5
Epoch 6/8
1000/1000 [==============================] - 1885s 2s/step - loss: 1.2668 - acc: 0.6050 - val_loss: 1.3654 - val_acc: 0.5880

Epoch 00006: saving model to check-06-1.3654.hdf5
Epoch 7/8
1000/1000 [==============================] - 1880s 2s/step - loss: 1.2024 - acc: 0.6257 - val_loss: 1.1765 - val_acc: 0.6391

Epoch 00007: saving model to check-07-1.1765.hdf5
Epoch 8/8
1000/1000 [==============================] - 1876s 2s/step - loss: 1.1522 - acc: 0.6409 - val_loss: 1.0508 - val_acc: 0.6802

Epoch 00008: saving model to check-08-1.0508.hdf5
```

num of epochs	accuracy	loss
1	0.2387	3.0565
2	0.3965	2.0169
3	0.4993	1.6261
4	0.5510	1.4435
5	0.5802	1.3445
6	0.6050	1.2668
7	0.6257	1.2024
8	0.6409	1.1522

Here is the plot for the accuracy:



5. **Summary and conclusions. Summarize the results you obtained, explain what have you learned, and suggest improvements that could be made in the future.**

   Compared with the neural network we designed (4-con-layer-SGD) together at the beginning, the performance of this (4-conv-layer-Adam) is much better. However, the accuracy is still not so high compared with the one with 3 conv layer Adam one.

   From this process, I learned that when design the network for the deep learning model, it's essential to choose the right number of neurons, layers at the first few layers. In other hidden layers, dropout ratio is important to keep its not too big. In the future, more number of epochs and number of sample should be chosen in doing the modeling.

6. **Calculate the percentage of the code that you found or copied from the internet.**
   The total number lines of code that I used as reference from internet is 31 lines. The number of lines I changed for this reference code part is 0. But the number of lines of code that I wrote in total by myself is 159, excluding comments lines and blank lines in the file. Hence the percentage is 31/(159+31)= 16.3%

7. **References**

   http://www.fki.inf.unibe.ch/databases/iam-handwriting-database
   https://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/
   http://cs231n.github.io/convolutional-networks/