Midterm Project
CNN on Hand Gestures 0 to 5

The code for this project can be found on the GitHub:
https://github.com/koarakawaii/Introduction_to_Deep_Learning

d07222009 蘇冠銘

Data Set

For data preprocessing, we first cut each collected picture into a square such that the hand gesture can be as centralized as possible. Then we rescaled the square to 256*256 pixel (with 3 channels RGB) and save the picture in another folder (train_data_resize and test_data_resize), see the code Data_Resized.ipynb. All these processes were done by OpenCV, but one need to be careful that OpenCV has reversed channels order with matplotlib convention. The training data thus obtained has 567 pictures: 98 zeros, 99 ones, 100 twos, 97 threes, 88 fours, and 85 fives. Some of them are took by cell phone, and the others are downloaded from the Internet. The original picture (without brightness and contrast adjustment) in the validation data given by TA is also included. By applying the data augmentation, including changing the brightness and contrast, flipping left and right, and transposing, the train set is further increased to 10,044 images (see folder Train_Data and code Data_Augmentation.ipynb).

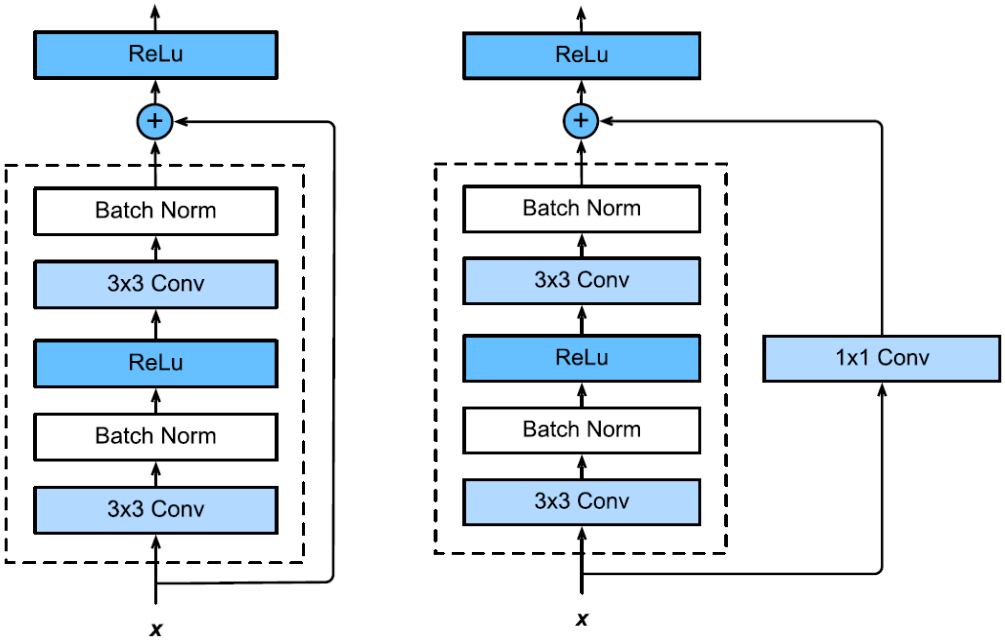
For training the network, the 10,044 images were separated into training and validation data set with ratio roughly 2:1, and the test accuracy on validation data is monitored after each epoch. After training completed, we used two test data set to assess the performance: one is the 300 images data set given by TA (see folder Test_Data). However, since part of the data might be already learned by the NN, we prepare another 56 totally new images to serve as an extra validation data (see Extra_Validation_Data). We will show the accuracy of each NN structure on both data set. As for testing the performance with new images, I think it will be better if tehy can be processed before hand as described in the 1st paragraph.

CNN Structure

Four CNN models are implanted for this project: three are similar to RESNET-18, RESNET-34 and RESNET-50, and one with DENSENET structure (all use relus as activation function). For the RESNET_18 and RESNET-34, the residual unit is show as below (the 1X1 convolutional layer is applied for matching the output channel of the

short circuit to that of the in-series convolutional layers), which basically composed of two convolutional layers. For RESNET-18, 2 residual blocks form a unit, and 4 unit with total (64, 128, 256, 512) features maps are applied; for RESNET-34, the number convolutional layer in a unit and their feature maps number are the same as before, but with (3, 4, 6, 3) residual blocks forming one unit. For each of them, the very first convolutional layer will half the size of feature layer (except for the first unit, which is already halved by the MaxPooling before it)

For RESNET-50, we changed number of convolutional layers in the residual block to 3, and the feature maps number of the 3rd layer is 4 times larger than 1st and 2nd layers. Those we need to use a 1X1 convolution layer with the same dimension for the short circuit. The number of residual blocks forming a unit is (3, 4, 6, 3) with feature maps number (32, 64, 128, 128) for the 1st and 2nd convolutional layers in the residual



blocks, and the very first convolutional layer also half the size of feature layer as mentioned above. For the detail of striding and padding, please see the code. And dropout layer is inserted between each unit for all three structure, with rate 0.1.

For the DENSENET structure, the number of convolution layers in each dense block are (4, 8, 16, 32), all with feature maps number equals 32, and the growing rate is 20. After the output of each layer is concatenated with those outputs of layers before, a transition layer which halves the channel dimension is applied. No dropout is utilized in this structure.

Training Procedure:

All four structures are trained over 10 epochs with larger learning rate, followed

by another 10 epochs with smaller learning rate. For RESNET structure, the larger rate is set as 5e-4, and the small one is 1e-4; for DENSENET, learning rates are 1e-4 and 5e-5 respectively. The batch size is always set as 64, and Adam trainer is selected, with β_1 and β_2 both equal 0.9. Four structures are all initialized by Xavier initialization, with gaussian type, and factor_type = in, magnitude = 2.0. The training and validation accuracy for RESNET-18 and RESNET-34 can both reaches 99% to 100%; RESNET-50 gives training accuracy roughly 99% along with 97% to 98% validation accuracy; for DENSENT can reach about 97% for training and 92% for validation. While all of them are higher than 90%, when using data sets they never learned before (like the extra validation dataset mentioned earlier), the performance is quite different. The final weights for each CNN structure is saved in the folder Weights_Train_Test_Mixed.

Performance:

	RESNET-18	**RESNET-34**	**RESNET-50**	**DENSENET**
Testing Data	89.67%	93.00%	82.33%	75.00%
Extra Validation	64.29%	73.21%	58.93%	33.93%
Average Time/epochs	36.89 s	65.93 s	77.13 s	100.38 s
Trainable Parameters	11,182,598	21,294,470	3,291,462	4,197,536

Owing to that the trainable parameters of RESNET-50 and DENSENET is nearly an order smaller than that of RESNET-18 and RESNET-34, we can know why the performance of RESNET-18 and RESNET-34 is much better than the other two structures. Inferring from that RESNET-34 has slight edge over RESNET-18, we can say that if the NN structure is designed adequately (say using residual blocks to learn residual), adding more layers seems giving better result, if the trainable parameters are kept roughly a constant. One thing I am curious is that although the parameters are fewer, when trying to add more parameters to RESNET-50 or DENSENET, they cannot be trained because GPU seems to be out of memory, but what occupied the memory, considering that RESNET-18 and RESNET-34 can both be trained with roughly 10 times larger weights?

We also plot out the density matrix as well as an interactive display on both test and extra validation data set to visualize the performance of each CNN. From the images in Test_Data set that RESNET-18 and RESNET-34 fail to recognize, we can see tt if the picture is too bright such that the edge of the hands cannot be easily distinguish, the convolutional network will perform badly. As for on the Extra_Validation_Data set, they both cannot recognized 1 very well (50% for RESNET-18 and 66.67% for RESNET-34), compared to the accuracy of other gestures. And we also designed a program

which can let user put their picture and make prediction for him/her, using the trained RESNET-34 (see RESNET_34_Look_at_Picture.ipynb).

Confusion Matrix for RESNET-34:

```
[ 2  0  0  0  0  0]
[ 0  4  2  0  0  0]
[ 0  1  5  0  0  0]
[ 0  2  4 19  2  0]
[ 0  0  0  2  3  0]
[ 0  0  0  1  1  8]
```

Confusion Matrix for RESNET-18:

```
[ 2  0  0  0  0  0]
[ 0  3  2  1  0  0]
[ 0  1  3  2  0  0]
[ 0  4  2 16  3  2]
[ 0  0  0  2  3  0]
[ 0  0  0  1  0  9]
```