Keras CNN in practice: hand-written digit recognition

In the second part of the assignment, I use Convolution neural network (CNN) for training. There are three main reasons why CNN is preferred compared to ANN.

Firstly, for many images, few features determine the label of the image and these features can be recognized without scanning the whole image.

Secondly, CNN can track features with different orientations. As a result, one filter can perform the tracking of the same feature.

Lastly, CNN has computation efficiency over ANN in image processing because the neurons in the convolution layers are partially connected.

The structure of the network is such that: sets of Con&MaxPooling → ANN → Output.

I choose increasing filter set up with 3*3 filter size because layer further down the network can detect more refined features. The filter will perform dot product with the input layer and result in a s 3D tensor. ReLU layers add nonlinearity to the system without making significant difference to the accuracy.

In the max-pooling layer, the input is divided by square of n*n. Picking the maximum inside each square can be considered as picking the most significant pattern with respect to each filter.

In the end, the reduced image is stretched and feed in a fully connected ANN. The activation function for the final output is “softmax” as it is the commonly preferred activation function in the classification problem.

There are few empirical discoveries during the training:

1. The accuracy is positively related to the number of filter, epoch and layers in general
2. The accuracy deteriorates significantly if simple dimension reduction (e.g. reduce the pixel by half) is performed.
3. Too many epochs for a same setting might result in overfitting and thus a decrease in accuracy.

Interestingly, less fitted submissions outperform the seemingly more accurate result in the end. The main difference is the number of epoch and the number of neurons in the fully connected layer.

In the end, my result for the private leaderboard is 96.4% (96.8% public). It was likely to be over fitted as I run 40 epochs on a rather simple model. Private results with similar setting but far less epochs have higher accuracy. Measures such as drop out and should be considered for future reference.

Varying epochs

Setting: 2 * Conv2D (64,3,3) & Maxpool (2,2) , FC1 (512) , FC2(10) , Training size = 32500 , Testing size 5*500 , lr = 6e-4

Epoch	Training Accuracy	Average Testing Accuracy
10	99.14%	95.2%
20	99.61%	94.7%
30	99.68%	95.4%

Varying filter number

Setting: Epoch = 20 , Maxpool(2,2) , FC1 (512) , FC2(10) , Traing size = 32500 , Testing size 5*500 , lr = 6e-4

Conv2D 1	Conv2D 2	Training Accuracy	Average Testing Accuracy
16	32	99.46%	93.23%
32	64	99.54%	94.6%
64	128	99.57%	95.2%
128	256	99.61%	95.3%

Kaggle results

Fixed Setting: Maxpool(2,2) , FC2(10) ,Traing size = 35000 , lr = 6e-4

Conv2D 1	Conv2D 2	Conv2D 3	Epoch	FC 1	Drop out	Public Accuracy	Private Accuracy
64	32	16	10	4096	0	94.8%	98%
64	64	0	10	4096	0	94%	96%
64	64	0	10	4096	0	93.2%	95.6%
32	64	0	10	4096	0	92.4%	97.6%
32	64	0	10	512	0	94%	96.4%
32	64	0	10	512	0	93.6%	97.6%
32	64	0	10	512	0	92.4%	96.4%
32	64	0	15	256	0.1	96%	97.2%
32	64	0	30	256	0.1	95.2%	97.6%
64 (submission)	128	0	40	256	0.1	96.8%	96.4%