Machine Learning ID: r05942042

<HW3 Report> Name: 王奕翔

1. Supervised Learning

My best result obtained from supervised learning is based on the structure in Table I. It could achieve 73.86% accuracy from the public set.

#	Input 3×32×32 RGB image	Attributes
1	64 Conv2D ReLU 64 Conv2D ReLU	Conv2D: kernel size = 3×3 border mode = same
2	Max-pooling	Max-pooling: pool_size = 2×2
3	128 Conv2D ReLU 128 Conv2D ReLU	
4	Max-pooling	Note that in #8, #9, and #10 layers, I added a BatchNormalization layer before the activation function.
5	256 Conv2D ReLU 256 Conv2D ReLU	
6	Max-pooling	
7	flatten	
8	512 FC ReLU	
9	512 FC ReLU	
10	10 FC softmax	

Optimizer	SGD	Batch size	50	Epoch#	500

Table I

I also did some alternatives, such as replacing the SGD with Adam. However, it would have ~15% Accuracy drop compared with the one with SGD. Based on my observation, using Adam does help with the convergence of training error, it converges to 98% training accuracy within 50 epochs. However, based on the testing results, it seems that it may have fallen into some local minimum.

I also tried a lot of different structures, such as some all-conv nets described in https://arxiv.org/abs/1412.6806. However, the results aren't superior to the one in Table. I.

2. Semi-supervised Learning (1)

My best result obtained from self-training is based on the structure in Table I , except that it uses 2 1024 FC at the end of the network instead of 2 512 FC. It could achieve 73.86% accuracy from the public set.

The self-training is done by using the 5000 labeled data to train model#1, then use model#1 to predict the whole set of 45000 unlabeled data, then sort these prediction results based on their probability, extract the 5000 most confident ones, and combine them with the labeled data. My best result was obtained by going through 8 iterations, which means that even though the trained models see the whole set of unlabeled data, it only uses a portion of it to build-up the model. Note that the 1st iteration goes through 500 epochs, whereas the rest of the iteration uses only 200 epochs.

I also tried assigning sample weights to each predicted unlabeled data, yet it had slightly poorer accuracy. Additionally, relabeling previous labeled data also yielded a poorer result. Hence, the predictions are only done once for each unlabeled data, and it is done with full confidence (without weighting).

3. Semi-supervised Learning (2): Autoencoder

I did this part by combining Autoencoder with SVM, I tried 2 different structures of Autoencoder, deep FC and CNN based. The CNN based Autoencoder seems to have a better result. Yet they both yield poorer results than that of self-training. My guess is that in order to make clustering be beneficial, some other techniques should be used to compensate the mis-labeling error.

The structure of my Autoencoder using CNN is shown below:

#	Input 3×32×32 RGB image	Attributes			
1	64 Conv2D ReLU	Conv2D: kernel size = 3×3 border mode = same Max-pooling: pool_size = 2×2			
2	Max-pooling				
3	32 Conv2D ReLU				
4	Max-pooling				
5	16 Conv2D ReLU				
6	Max-pooling				
	Output Code (16,4,4)				
Optimizer	Adam	Batch size	50	Epoch#	5

Table II

The decoder end is symmetric to the encoder, except that Max-pooling is replaced with up-sampling.

The Autoencoder is trained by feeding all the data into it, including the test data, since all valid 32×32 images are all beneficial to the Autoencoder. Subsequently, a selected set of the unlabeled data and the labeled data would be fed into the encoder end of the Autoencoder to generate its code. The output code is then flattened into a 256 code and sent to a SVM model to perform multi-class classification to predict the unlabeled data. After labeling the selected set of unlabeled data, the data would be sent to a CNN with the same structure in Table I to do supervised training.

	Autoencoder(deep) + SVM	Autoencoder(CNN)+SVM
Unlabeled Data used	100%	~10%
Accuracy	27.46%	51.70%

We can see that CNN + only a portion of the unlabeled data yields a slightly better result than the baseline.

4. Compare and analyze your results:

Comparison between supervised and variations of Self-Training:

[1]	Best supervised:	73.86%
[2]	Using all-conv net + 4×5000 most confident unlabeled data:	74.96%
[3]	Same structure as [1] + 4×5000 unlabeled data (include test data):	77.80%
[4]	Same structure as [1] + 6×5000 unlabeled data (include test data):	78.12%
[5]	Same structure as [1] + 6×5000 unlabeled data (without test data):	78.88%
[6]	Same structure as [1] + 8×5000 unlabeled data (without test data) and the FC is changed into size 1024:	79.08%
[7]	Same structure as [1] + 8×5000 unlabeled data (include test data) and the FC is changed into size 1024:	79.52%
[8]	Same as [7], but epoch# is changed from 200 to 100	77.56%
[9]	Same structure as [7] but with 1.5 times the filter bank size + data augmentation is used in the 1st iteration: + 3 FC of size 1024 in the end	79.78%

We can see that self-training does make some improvements on the accuracy. Besides, based on the results, if the self-training were to incorporate more data, the network structure could be altered a bit to acquire some more gain, even if that new structure might not be good for supervised training.

Note that even though [9] is slightly superior than [7], the one described in section 2 is [7], and it's also the version in my git repository. The reason is that [9] takes about half a day to train, and I accidentally overwrote its model. Besides, I really need some sleep ;_;