ML Hw3 B03202047 王昱翔

In the whole report, I'll abbreviate the following words in Table 1

Convolution2D	C2D
Maxpooling	MP
Dense	D
Dropout	DP
Flatten	F
sigmoid	Sig
UpPooling	UP

Table 1

(1) Supervised Learning

- Method

Training	4500
Validation	500
epoch	70
CNN structure(↓)	
Input -> C2D(30,3,3)->MP(2,2)->C2D(60,3,3)->MP(2,2)-> F->D(689)->Sig->DP(0.25)->D(10)->softmax->Output	

I cut 500 data from given labeled dataset for validation. I'll check the validation score before submit the result to Kaggle. In addition, I chose Adam for my optimizer of three models.

- Results on Kaggle

As for the convolution layer, I have tried two and three C2D layers, and two performed better. For the fully connected NN, I also have tried one, two and three layers. Surprisingly, only one layer got the best performance.

C2D(25) + C2D(50) + D(100)	0.52 (val)
C2D(26) + C2D(52) + D(300)	0.53 (val)
C2D(30) + C2D(60 + C2D(90) + D(689)	0.49 (val)
C2D(30) + C2D(60) + D(689) + DP(0.25)	0.55 (val)

Training loss	Training acc	Validation acc	Kaggle Public	Kaggle Private
0.1847	~1	0.5547	0.54140	0.55260

(2) Semi-supervised Learning -- Self-training

- Method

I trained the CNN model (the same as Supervised Model) with labeled data first. Then predict on the unlabeled data. During the prediction, I only added the data to training data which maximum value > 0.95 after softmax. After prediction, the size of training data is 16384. The only difference between new CNN and the Supervised CNN model is in the fully connected NN part.

F->D(300)->relu->DP(0.25)->D(689)->Sig->DP(0.25)->D(10)->softmax->Output

- Results on Kaggle

Threshold	Dataset Size	Val acc

0.8	33457	0.49
0.9	23756	0.53
0.95	16384	0.56
0.995	6681	0.55

Training size	Threshold	Kaggle Public	Kaggle Private
16384	0.95	0.55980	0.56700

(3) Semi-supervised Learning – Autoencoder

- Method

  I trained a Deep CNN autoencoder with label, unlabeled and test data.

  The Deep autoencoder structure is shown below

Deep Convolution Autoencoder
Input -> C2D(16,3,3)->MP(2,2)->C2D(16,3,3)->MP(2,2)-> C2D(16,3,3)->MP(2,2) -> C2D(16,3,3)->UP(2,2)-> C2D(16,3,3)->UP(2,2)->C2D(16,3,3)->UP(2,2)->C2D(3,3,3)

  (red part: encode)

  I use the auto encoder to calculate 10 mean values of the 10 class labeled encoded values. If min(Euclidean dist($mean_{encoded\_i(0\leq i\leq 9)}, unlabedld_{encoded}$)) < 3, I'll add the unlabeled data as training data.

  Then I build a CNN model, which is composed of the encoded + Fully connected NN. As for the Fully connected NN, it is composed of F->D(689)->Sig->DP(0.25)->D(10)->softmax->Output.

- Results on Kaggle

Training data	Kaggle public	Kaggle private
8897	0.53580	0.53820

(4) Compare and Analyze Results

It is obviously that three methods in my report didn't show significant difference on Kaggle. The only work semi-supervised method is Self-training. I think self-training can do better, if I do more iteration on collecting unlabeled data. However, I thought Autoencoder is the most probable model to improve the task. However, not only training autoencoder took a lot of time but also got worse performance. Maybe I should add more filters in Autoencoder in each layers. The following table summary three methods.

Model	Supervised	Self-training	Autoencoder
Training time	< 5 min	10 min	30 min
Training size	4500	4500 + 16384	60000 + 8897
epoch	70	70 + 120	40 + 120
Train Acc	~1	0.73	0.43