Machine Learning
Convolution neural network

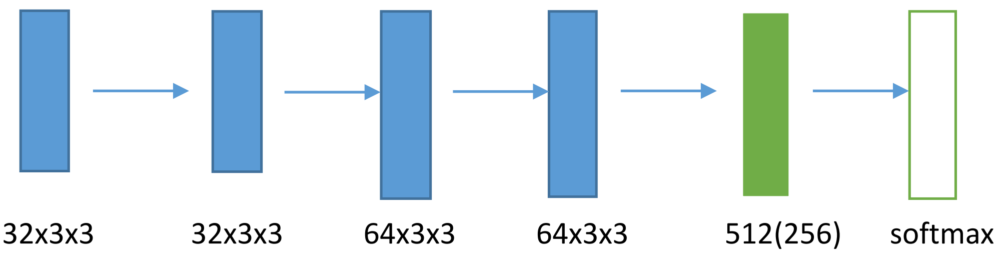
B02901031 陳緯哲

- ## My CNN Architecture

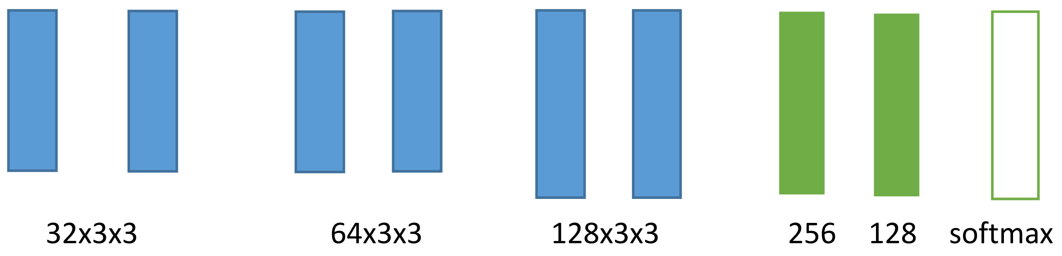
I build two different CNN and do some expriments on them.

Blue blocks: Conv layer, Green blocks: FC layer, White blocks: softmax

CNN1 : 4 conv layer and 2 fully-connected layer with ReLU activation



CNN2 : 6 conv layer and 2 fully-connected layer with ReLU activation and Batch-Normalization.



- ## Tricks

I. Validation data

I split the labeled data by 450 training data and 50 validation data class-by-class. And shuffle them randomly.

II. Data generator

Using the keras dategen class to generate different size, angle, and width shift data. By doing so, CNN can get amazing improvements

III. Batch-Normalization(BN)

By adding batch normalization layer after all non-linear layer. We can improve validation accuracy by about 3~5%

IV. Self training(semi-supervised training)

After (1) training the model on labeled data, I (2)predict the unlabeled(test data is included) data. (3)Add the data which have probability of a class over 0.95 to training data, and train them for one epoch.(4) To adjust the model back, retrain the correct data(4500 labeled data) for 3 epochs. Repeat 2~4 three times

Supervised learning

Using only 4500 labeled data to predict data.

	CNN1:FC 512	CNN1:FC 256	CNN2	CNN2+BN
Without datagen	0.4910	0.4876	0.5620	0.5792
With datagen*10	0.6530	0.6600	0.6990	0.7512

Epochs = 30 , Adam

Unsupervised learning

Method is illustrated in the Tricks part

	CNN1:FC512	CNN1:FC256	CNN2	CNN2+BN
Without datagen	0.2310	0.2560	0.5601	0.5810
With datagen*10	0.5231	0.5201	0.7692	0.8180

Epochs = 20, Adam

Discussion:

Adding the unlabeled data to the model pretrained by labeled data.

The model training without datagen seen to have too low accuracy to have correctly label the data. As a result, adding the uncorrect data to training data will deterioration the model. However, after apply datagen, the model accuracy is high enough to create reliable data, so the accuracy can be improved.

Auto-encoder

Firstly, I train the auto-encoder network to fit the image by using all the data (labeled + unlabeled +test). Then extract the 256 dimension representation in the middle layer, which is the image feature. Place the labeled layer in the 4 layers DNN network to classify the image. We can get the classification result.

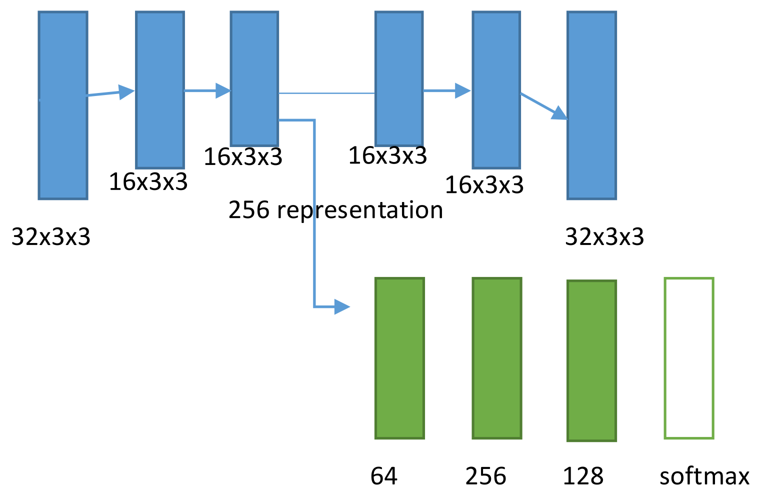
Expriment on autoecoder:

	10 epochs	20 epochs	100epochs
loss	0.022	0.011	0.0088

Finally I choose the 30 epochs model as my encoder

Expriment on classification DNN:

	10 epochs	50 epochs	100epochs
Val acc	0.2680	0.4210	0.4128


32x3x3
16x3x3
16x3x3
256 representation
16x3x3
16x3x3
32x3x3
64
256
128
softmax