Machine Learning Homework 3 Report

R04921039 彭俊人

1. Supervised learning:

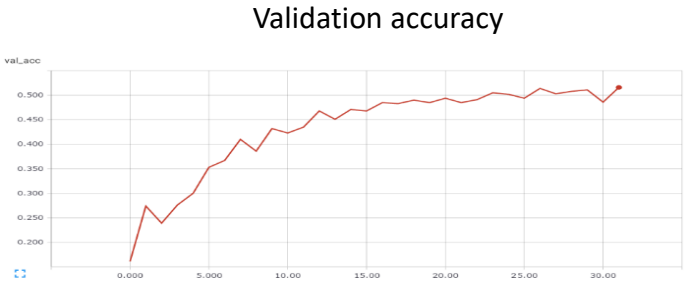
Layer (type)	Output Shape	Param #	Connected to
convolution2d_1 (Convolution2D)	(None, 32, 32, 32)	896	convolution2d_input_1[0][0]
activation_1 (Activation)	(None, 32, 32, 32)	0	convolution2d_1[0][0]
convolution2d_2 (Convolution2D)	(None, 30, 30, 32)	9248	activation_1[0][0]
activation_2 (Activation)	(None, 30, 30, 32)	0	convolution2d_2[0][0]
maxpooling2d_1 (MaxPooling2D)	(None, 15, 15, 32)	0	activation_2[0][0]
dropout_1 (Dropout)	(None, 15, 15, 32)	0	maxpooling2d_1[0][0]
convolution2d_3 (Convolution2D)	(None, 15, 15, 64)	18496	dropout_1[0][0]
activation_3 (Activation)	(None, 15, 15, 64)	0	convolution2d_3[0][0]
convolution2d_4 (Convolution2D)	(None, 13, 13, 64)	36928	activation_3[0][0]
activation_4 (Activation)	(None, 13, 13, 64)	0	convolution2d_4[0][0]
maxpooling2d_2 (MaxPooling2D)	(None, 6, 6, 64)	0	activation_4[0][0]
dropout_2 (Dropout)	(None, 6, 6, 64)	0	maxpooling2d_2[0][0]
flatten_1 (Flatten)	(None, 2304)	0	dropout_2[0][0]
dense_1 (Dense)	(None, 512)	1180160	flatten_1[0][0]
activation_5 (Activation)	(None, 512)	0	dense_1[0][0]
dropout_3 (Dropout)	(None, 512)	0	activation_5[0][0]
dense_2 (Dense)	(None, 10)	5130	dropout_3[0][0]
activation_6 (Activation)	(None, 10)	0	dense_2[0][0]

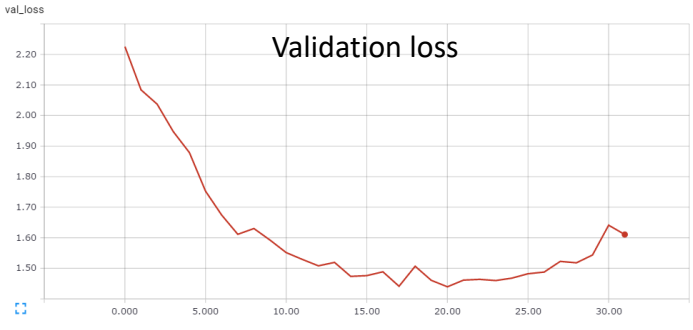
Total params: 1250858

```
model = Sequential()
model.add(Convolution2D(32, 3, 3, border_mode = 'same',
                        input_shape = (X_train.shape[1:])))
model.add(Activation('relu'))
model.add(Convolution2D(32, 3, 3))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Dropout(0.25))

model.add(Convolution2D(64, 3, 3, border_mode = 'same'))
model.add(Activation('relu'))
model.add(Convolution2D(64, 3, 3))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(output_dim = 512))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(output_dim = 10))
model.add(Activation('softmax'))
```

My model derived from the official keras **cifar10_cnn.py**, containing 4 convolution2D layers, 2 Maxpooling layers and 2 Dense layers to flatten network for final 10 outputs classification. In the beginning, I split the 5000 labeled data into 4000 training data set and 1000 validation data set. Then I train the model with the 4000 training data and regulated by 1000 validation loss.

I imported keras callback module: “EarlyStopping” to monitor “validation loss” with 10 epoch patience delay. I also used the Tensorboard module to record both training and validation status as following.



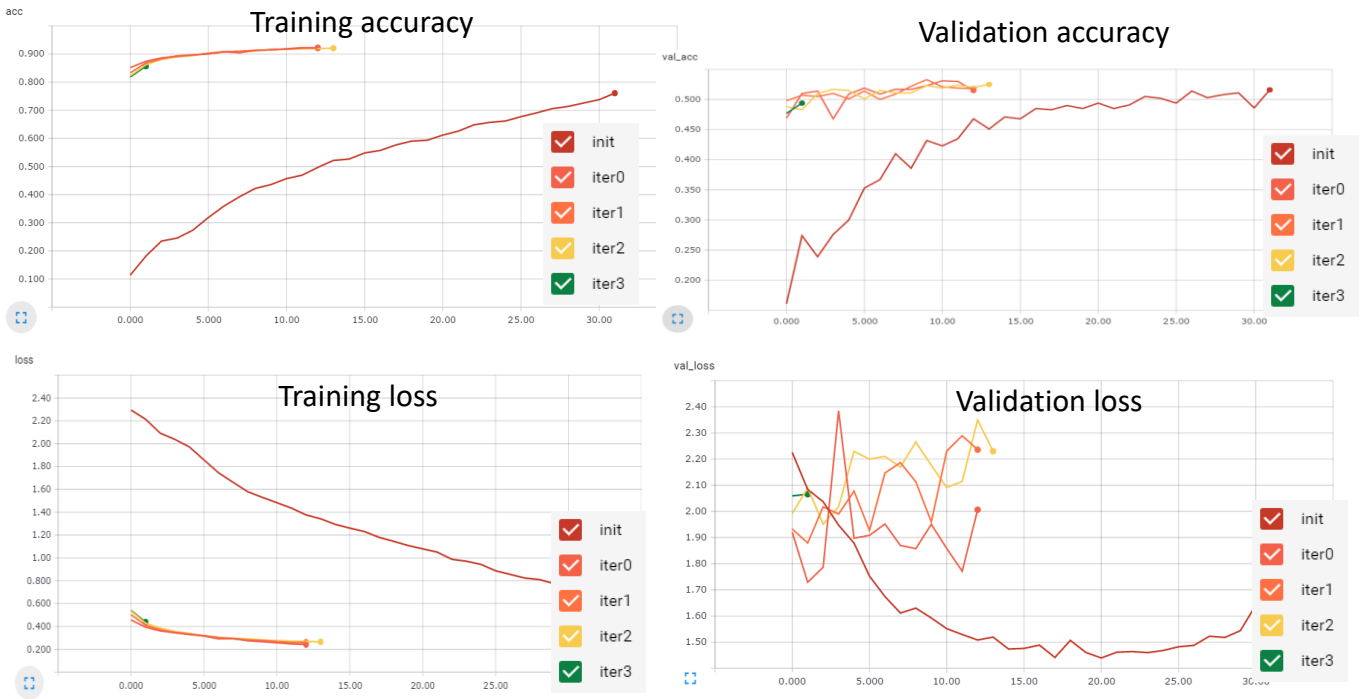


The graph on the left is training accuracy in every step, and the graph on the right is the validation accuracy. We can see clearly that validation data converges around 50% of accuracy, and training stopped due to early stopping criteria. In order to increase validation accuracy. We need to train with unlabeled data.

2. **Semi-supervised learning (1):**

I used **self-training** method to increase my training data set. First, I predicted all unlabeled data by the model we trained with labeled data in part 1. Later, for each predicted vector, I set a 0.8 threshold for each maximum class probability. For those maximum probability over 0.8, I added the unlabeled data into training data, with the maximum probability as its label. I chose 0.8 since it contains almost 80% of unlabeled data in the first iteration.

Normally, this process should continue until the model is good enough that all the unlabeled data could have a label with possibility above threshold. Here I only repeat the process for 10 times. I also use "EarlyStopping" to monitor "validation accuracy" with 10 epoch patience delay. The result is recorded and showed by Tensorboard module.



Here I only showed the first three iterations, since the rest iterations did not differ much. I found out that although we could add more and more unlabeled data as training data, the accuracy did not increase significantly. Also, the increment of accuracy does not always result in loss decrease. So we need to monitor validation accuracy instead of validation loss for early stopping. Moreover, each iteration when new data appeared in the training data set, we need to start from almost the same accuracy again. I suspect it's because my model is not complicated enough, so the accuracy could not improve even with more training data. We should add more layers of convolution2D for feature extraction.

3. Semi-supervised learning (2):

Layer (type)	Output Shape	Param #	Connected to
convolution2d_1 (Convolution2D)	(None, 32, 32, 32)	896	convolution2d_input_1[0][0]
convolution2d_2 (Convolution2D)	(None, 32, 32, 32)	9248	convolution2d_1[0][0]
maxpooling2d_1 (MaxPooling2D)	(None, 16, 16, 32)	0	convolution2d_2[0][0]
convolution2d_3 (Convolution2D)	(None, 16, 16, 64)	18496	maxpooling2d_1[0][0]
convolution2d_4 (Convolution2D)	(None, 16, 16, 64)	36928	convolution2d_3[0][0]
maxpooling2d_2 (MaxPooling2D)	(None, 8, 8, 64)	0	convolution2d_4[0][0]
flatten_1 (Flatten)	(None, 4096)	0	maxpooling2d_2[0][0]
dense_1 (Dense)	(None, 512)	2097664	flatten_1[0][0]
dense_2 (Dense)	(None, 10)	5130	dense_1[0][0]
activation_1 (Activation)	(None, 10)	0	dense_2[0][0]
dense_3 (Dense)	(None, 512)	5632	activation_1[0][0]
dense_4 (Dense)	(None, 4096)	2101248	dense_3[0][0]
reshape_1 (Reshape)	(None, 8, 8, 64)	0	dense_4[0][0]
upsampling2d_1 (UpSampling2D)	(None, 16, 16, 64)	0	reshape_1[0][0]
convolution2d_5 (Convolution2D)	(None, 16, 16, 32)	18464	upsampling2d_1[0][0]
convolution2d_6 (Convolution2D)	(None, 16, 16, 32)	9248	convolution2d_5[0][0]
upsampling2d_2 (UpSampling2D)	(None, 32, 32, 32)	0	convolution2d_6[0][0]
convolution2d_7 (Convolution2D)	(None, 32, 32, 3)	867	upsampling2d_2[0][0]

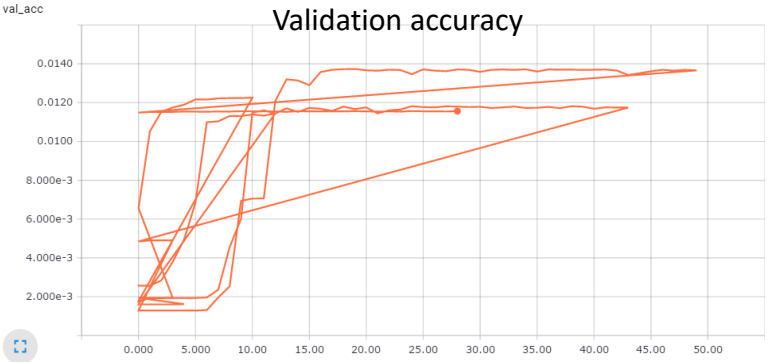
Total params: 4303821

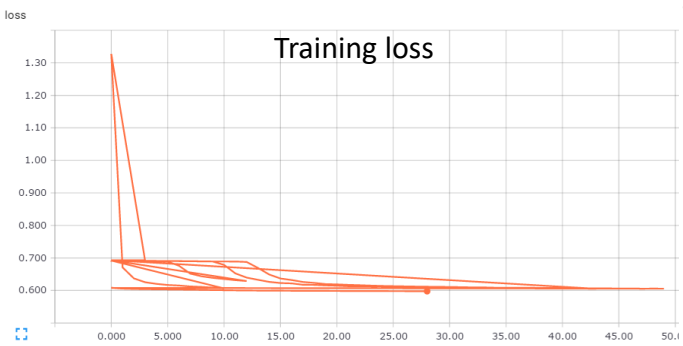
I used **clustering** method to classify unlabeled data set. I used the previous CNN model and extend it by UpSampling, Convolution2D and Reshape layers to construct an autoencoder. Then I use both labeled data and unlabeled data to train the autoencoder to reconstruct the image and learn features in the process. When the autoencoder learned how to reconstruct images, it had also learned some crucial features. I pop half of the model to resume to the previous CNN model. Then I repeat the process in part 2. This step is called pre-training. The reason why we trained the autoencoder is to let the model identify and construct useful features with unlabeled data. Then it would be easier to classify those unlabed data with the CNN model.

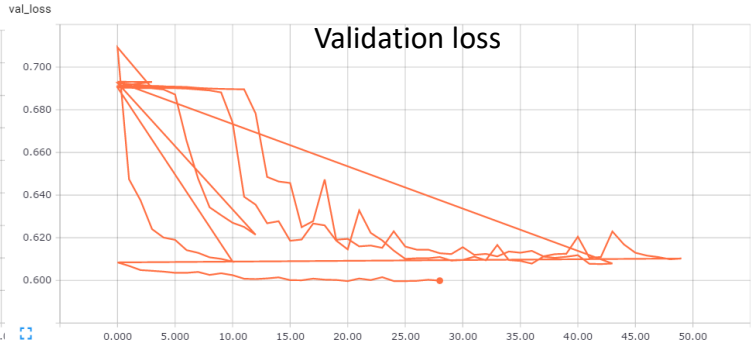
Following figures are the results of autoencoder training.

Training accuracy

Validation accuracy







Experiment results showed that autoencoder converged quickly at around 0.012 accuracy. I tried different optimizers and also split unlabeled data into smaller batches and repeat it several times (this is the reason for recursive lines in the figures). However, it didn't help. Again, I suspect it was because my model is not complicate enough, so that it would be more difficult for image reconstruction. Later, it was the CNN model training doing most of the learning and the result is the same as figures in part 2.

4. Compare and analyze results

In my experiment, I found that semi-supervised learning could help increase number of training data by classifying unlabeled data. This is a useful and important method since labeling data is expensive and not always possible. However, we still need good and complicate models to improve accuracy while training or pre-training the model with larger data set. In my experiments, both self-learning and clustering methods succeed in increasing training data set, but did not help to improve accuracy. Most of the learning was done by supervised learning due to the simplicity of my model. A deeper CNN and autoencoder should be used for better performance.