Ridge-I assignment
Supervised and unsupervised training

Definition

Assignment Overview

The main objective of this assignment is to design and construct a combination of supervised and unsupervised architectures in a single model to do a classification task. The unsupervised model must be either an autoencoder or a stacked autoencoders. The supervised model is a classifier which takes input from the encoder output and determines classified labels as shown in the Fig-1. The combined architecture trains by CIFAR-10 dataset under the condition that it can uses only 50% of training data of bird, deer and truck classes while for other classes data size is arbitrary.

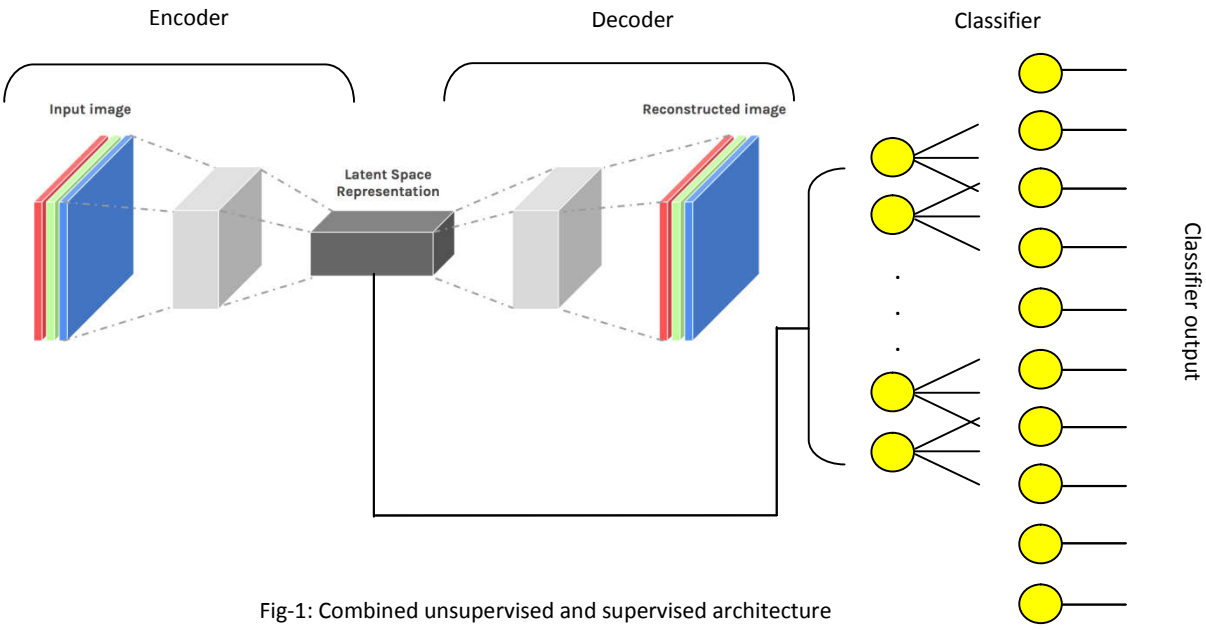

Fig-1: Combined unsupervised and supervised architecture

Problem Statement

The goal is to create a supervised and unsupervised model for classification; the task involved the following steps:

1. Download and preprocess the CIFAR-10 dataset
2. Split the train data of CIFAR-10 for training and validation by following data splitting condition given in assignment
3. Construct the whole architecture which consists autoencoder and classifier
4. Build the autoencoder and train it using training dataset and validate it using validation dataset.
5. After training save the model weights
6. Follow the step 2 using different randomization parameter for classifier training purpose
7. Build the classifier by removing the decoder of the autoencoder and adding a couple of Conv2D layers & dense layer with activation function at the top of the encoder as stated in the assignment

8. Load the encoder's weights of the trained autoencoder in to the few layers of the new classification model
9. In autoencoder the encoder part produces lower dimensional representation of input images and feed it to the classifier part
10. First make lower layers non-trainable and train only the classifier layers
11. Then train the whole classifier by making all layers trainable and save the model
12. Test the classifier using test data of CIFAR-10 and show the results

Data Exploration

The CIFAR-10 dataset consists of 60000 tiny RGB images with dimension 32x32 and divided in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images [1]. The classes correspond to mostly objects (such as ships, cars, airplanes, etc.) and animals (cats, dogs, frogs, etc.). The images are indeed very small; objects within classes has extremely varying size, pose and visibility. For example "bird" class contains many different type of bird (big, small, flying, non-flying, etc.); also some time only parts of the bird are shown. Almost same things are true for other classes.

Algorithm and Techniques

Convolutional Neural Networks are used for both the autoencoder and classifier building. These type of networks need large amount to training data; fortunately CIFAR-10 is large enough. However the following parameters can be tuned to optimize the classifier. I have applied them in different degrees in various learning situation to get best optimized model.

- Training parameters
  - Number of epochs
  - Batch size
  - Learning rate
  - Decay of learning rate
- Network architecture
  - Number of layers
  - Layer type (such as Conv, Pooling, Batch Normalization, Dropout, Dense)
  - Layer parameters (such as kernel number & size, activation function, stride, padding, etc.)
  - Regularizations
- Preprocessing parameter
  - Augmentation

Methodology

Data Preprocessing

The CIFAR-10 images are all pre-segmented (e.g. each image contains a single object), that the images all have the same square size of 32x32 pixels, and that the images are color. Therefore, it is ready to load the images and use them for modeling almost immediately. There are 10 classes are represented as unique integers. Therefore, I used a one hot encoding for the class element of each sample, transforming the integer into a 10 elements binary vector with a 1 for the index of the class. The pixel values of images are rescaled to the range [0, 1].

After all of these, it is necessary to partition the data. In order for the model generalize well, usually it is good to split the training data into two parts: a training and a validation set in around 80% and 20%. For the data constraint given on the class of bird, deer and truck I split whole training data equally; 50% for training and 50% for validation for all classes to make them balanced. At first I did not go through resampling, synthetic sampling or other techniques to combat with imbalanced datasets.

Model Definition

The autoencoder model summary is

```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 32, 32, 3)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 32, 32)        896
_________________________________________________________________
batch_normalization_1 (Batch (None, 32, 32, 32)        128
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 32, 32, 32)        9248
_________________________________________________________________
batch_normalization_2 (Batch (None, 32, 32, 32)        128
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 16, 64)        18496
_________________________________________________________________
batch_normalization_3 (Batch (None, 16, 16, 64)        256
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 16, 16, 64)        36928
_________________________________________________________________
batch_normalization_4 (Batch (None, 16, 16, 64)        256
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 8, 8, 64)          0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 8, 8, 128)         73856
_________________________________________________________________
batch_normalization_5 (Batch (None, 8, 8, 128)         512
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 8, 8, 128)         147584
_________________________________________________________________
batch_normalization_6 (Batch (None, 8, 8, 128)         512
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 4, 4, 128)         0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 4, 4, 128)         147584
_________________________________________________________________
batch_normalization_7 (Batch (None, 4, 4, 128)         512
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 4, 4, 128)         147584
_________________________________________________________________
batch_normalization_8 (Batch (None, 4, 4, 128)         512
_________________________________________________________________
up_sampling2d_1 (UpSampling2 (None, 8, 8, 128)         0
```

```
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 8, 8, 64)          73792
_________________________________________________________________
batch_normalization_9 (Batch (None, 8, 8, 64)          256
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 8, 8, 64)          36928
_________________________________________________________________
batch_normalization_10 (Batc (None, 8, 8, 64)          256
_________________________________________________________________
up_sampling2d_2 (UpSampling2 (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 16, 16, 32)        18464
_________________________________________________________________
batch_normalization_11 (Batc (None, 16, 16, 32)        128
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 16, 16, 32)        9248
_________________________________________________________________
batch_normalization_12 (Batc (None, 16, 16, 32)        128
_________________________________________________________________
up_sampling2d_3 (UpSampling2 (None, 32, 32, 32)        0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 32, 32, 3)         867
=================================================================
Total params: 725,059
Trainable params: 723,267
Non-trainable params: 1,792
```

The classifier model summary is

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 32, 32, 3)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 32, 32)        896
_________________________________________________________________
batch_normalization_1 (Batch (None, 32, 32, 32)        128
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 32, 32, 32)        9248
_________________________________________________________________
batch_normalization_2 (Batch (None, 32, 32, 32)        128
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 16, 64)        18496
_________________________________________________________________
batch_normalization_3 (Batch (None, 16, 16, 64)        256
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 16, 16, 64)        36928
_________________________________________________________________
batch_normalization_4 (Batch (None, 16, 16, 64)        256
_________________________________________________________________
```

```
max_pooling2d_2 (MaxPooling2 (None, 8, 8, 64)          0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 8, 8, 128)         73856
_________________________________________________________________
batch_normalization_5 (Batch (None, 8, 8, 128)         512
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 8, 8, 128)         147584
_________________________________________________________________
batch_normalization_6 (Batch (None, 8, 8, 128)         512
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 4, 4, 128)         0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 4, 4, 256)         295168
_________________________________________________________________
batch_normalization_7 (Batch (None, 4, 4, 256)         1024
_________________________________________________________________
activation_1 (Activation)    (None, 4, 4, 256)         0
_________________________________________________________________
dropout_1 (Dropout)          (None, 4, 4, 256)         0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 4, 4, 256)         590080
_________________________________________________________________
batch_normalization_8 (Batch (None, 4, 4, 256)         1024
_________________________________________________________________
activation_2 (Activation)    (None, 4, 4, 256)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 4, 4, 256)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 4096)              0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               524416
_________________________________________________________________
batch_normalization_9 (Batch (None, 128)               512
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                1290
=================================================================
Total params: 1,702,314
Trainable params: 1,412,234
Non-trainable params: 290,080
```

Implementation

At the first stage the autoencoder is trained using training-data, training ground-truth and validation-data, validation ground-truth. The training curve (loss vs. epochs) is shown in Fig-2 and Fig-3 showing some reconstructed samples.

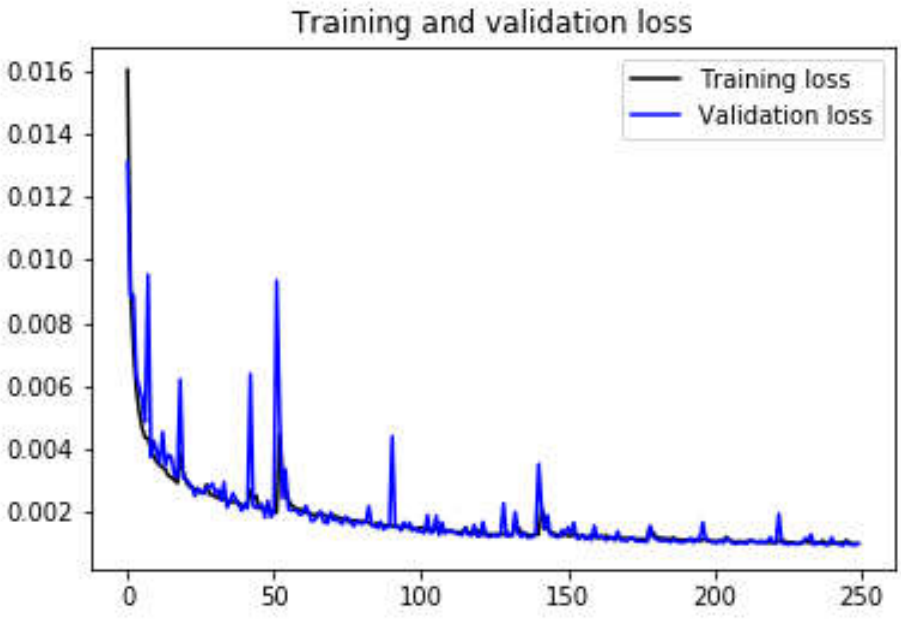

Fig-2: Autoencoder training and validation loss curves

Top row contains few samples of original test images and the bottom row is the corresponding reconstructed images

Top row contains few samples from the validation set and the bottom row is the corresponding reconstructed images

Fig-3: Original images and the corresponding reconstructed images produced by autoencoder

The initial learning curve (loss vs. epochs and accuracy vs. epochs) only for classifier layers while freezing the encoder part is shown in Fig-4. The fig-5 shows the learning curve produced when all layers are trainable.

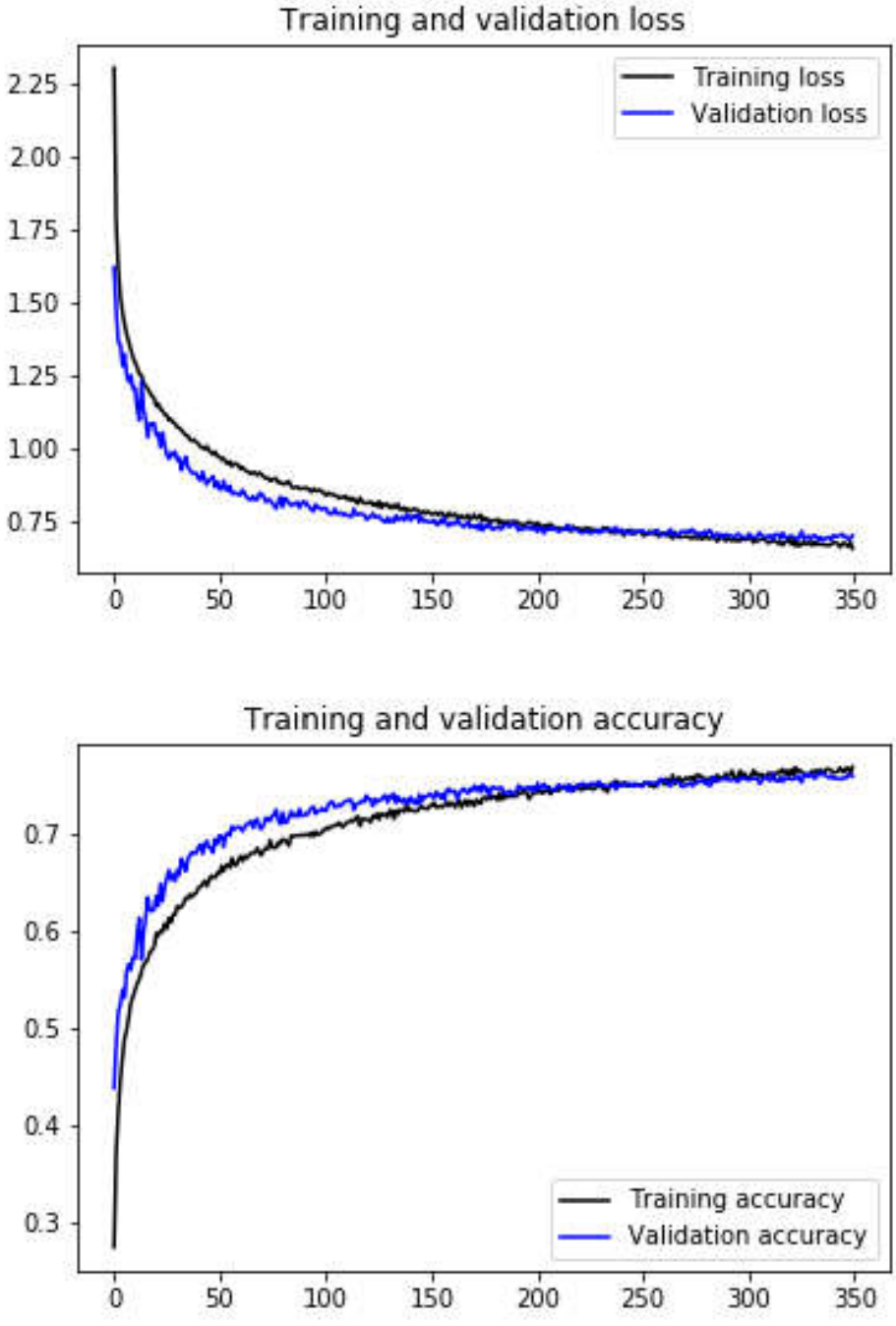

Fig-4: Initial learning curve of classifier

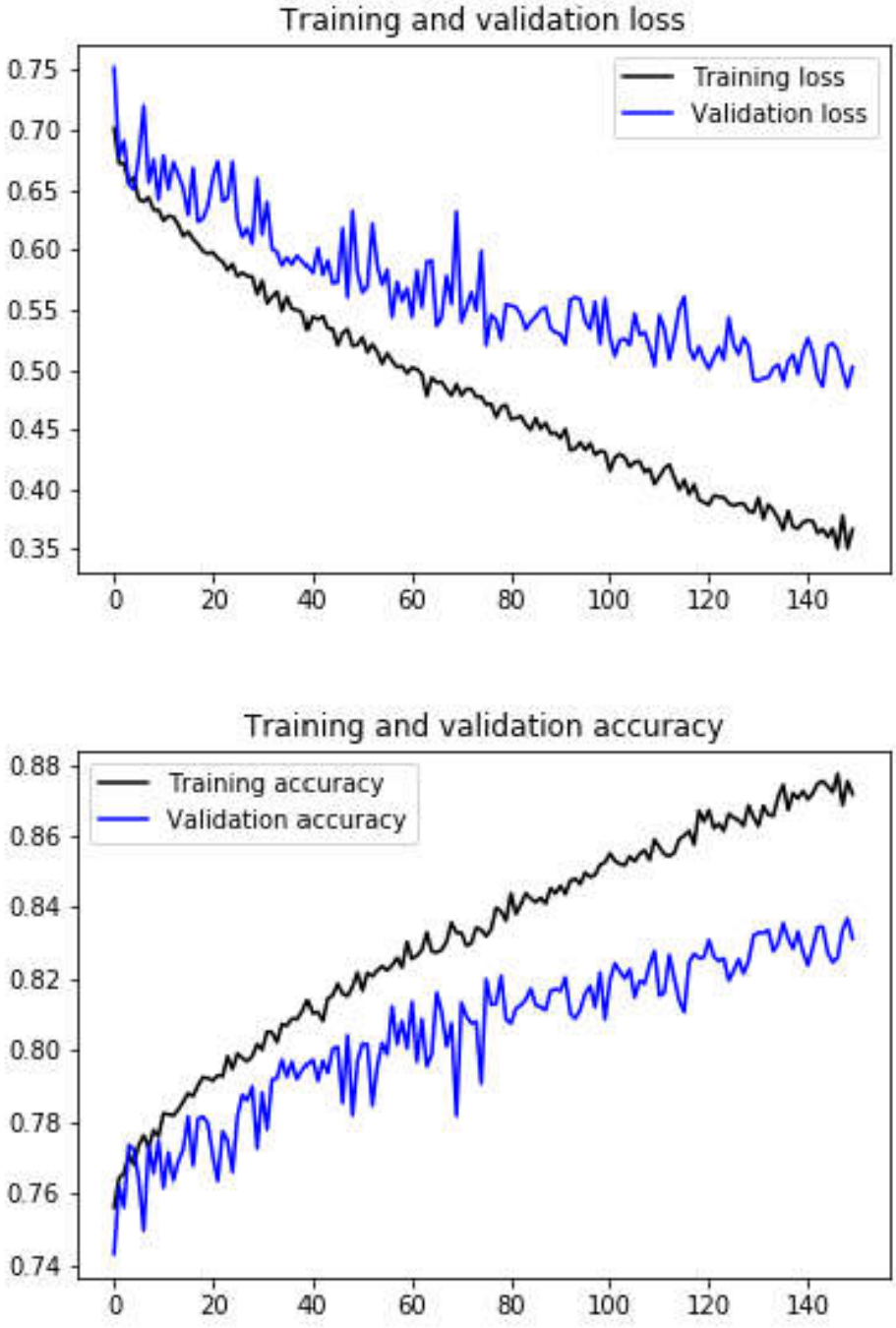

Fig-5: Final learning curves of classifier

Results and discussion

The classifier produces test accuracy is about 83% using CIFAR-10 test set; the confusion matrix shown in Fig-6. The bird, cat and dog has reasonably little low accuracy rate compare to the other classes.

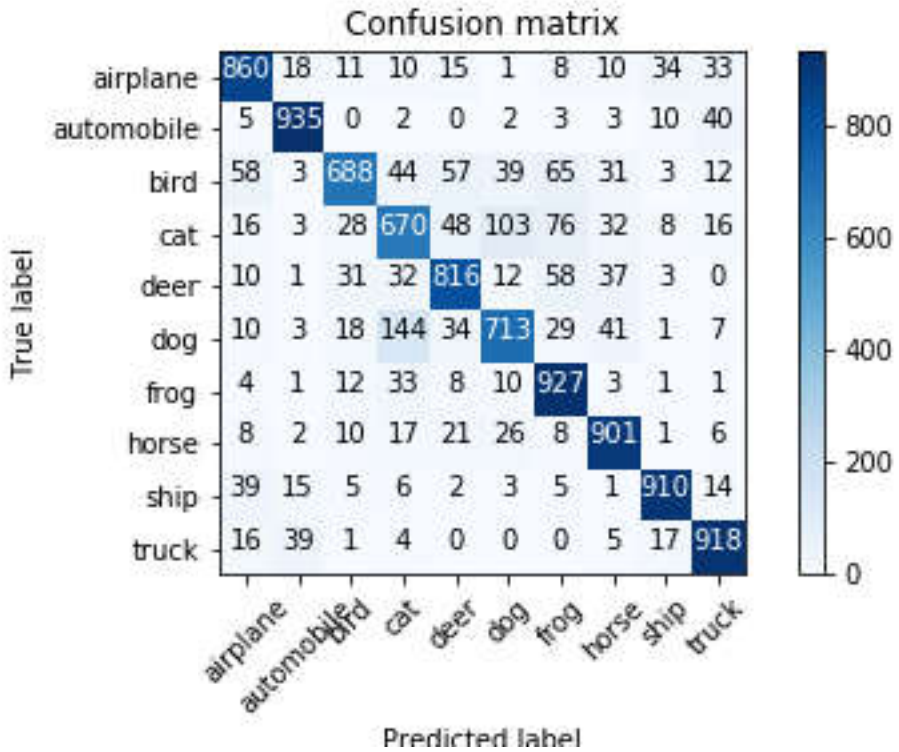

Fig-6: Confusion matrix

The classification performance is not good enough in comparison to other state-of-the-art methods. In the final learning stage little overfitting occurs after 80 epochs as shown in Fig-4 and Fig-5. I have tried several kind of regularization techniques to overcome this situation, such as strong data augmentation, changing dropout parameter, changing batch size, learning rate, applying l2, etc. I also tried by reducing network complexity, changing kernel number and size, activation function, loss function, optimizers, etc. In another approach, to make the training data size bigger I generated synthetic data from original training data by autoencoder. Unfortunately, nothing produced better results.

However, using stacked autoencoder or/and unsupervised, supervised joint training may improve the result.

Reference

1. https://www.cs.toronto.edu/~kriz/cifar.html