Home Work 5

Name : Sai Teja Chava
Course Name : Advanced Computer Vision(CSCI 677)
USC ID : 5551847788
Instructor Name : Prof. Ram Nevatia

Steps in Code:

1. Import all the required packages.
2. Read the required parameters (epochs, batch size and path to save plot) as arguments.
3. Open the train file and load it using pickle and extract the data and labels from it.
4. Split the training dataset into training and validation sets with 40,000 in training and 10,000 in validation.
5. Reformat the training dataset i.e. convert to gray scale and then perform data augmentation -Enlarge the image by 10% and crop 90% of the image in left-top, right-top, left-bottom, right-bottom and central. Also do vertical flip for each of the 5 images generated from above based on random number generated between 0 and 1.
6. One-hot encode the training labels.
7. Randomize the training dataset.
8. Compute mean for entire training data and subtract it from the training dataset (Mean Subtraction).
9. Apply similar steps to the validation dataset too but do not apply augmentation to validation dataset i.e. convert to grayscale, one-hot encode followed by mean subtraction.
10. Apply similar steps as validation dataset to test dataset too.
11. Start training the LeNet network in mini-batches for specified no of epochs.
12. Keep track of training and validation losses in each epoch so that you can plot them and visualize later.
13. Once training is done, do a forward pass on the test dataset and print the accuracy obtained.
14. Compute the confusion matrix using the ground truth labels and the predicted labels and print them.
15. Compute rank 1 accuracies for each class and super class using the meta information in the meta file provided and display/print them.
16. Compute rank 5 accuracies for each class and super class using the meta information in the meta file provided and display/print them.
17. Compute overall rank 5 accuracy and display/print it.
18. Plot the training loss and validation loss over epochs and see if it is overfitting.

Note:

1. **I have also used gray scale images instead of color images which everyone used since LeNet was designed for 32 x 32 x 1 images. I have also spoken about this to TA(Yiqi) and she agreed to that.**

2. **All the meta information pertaining to each of the experiments is included in the Appendix section at the end of the document.**

Results:

Changing the no of epochs and not changing the architecture:

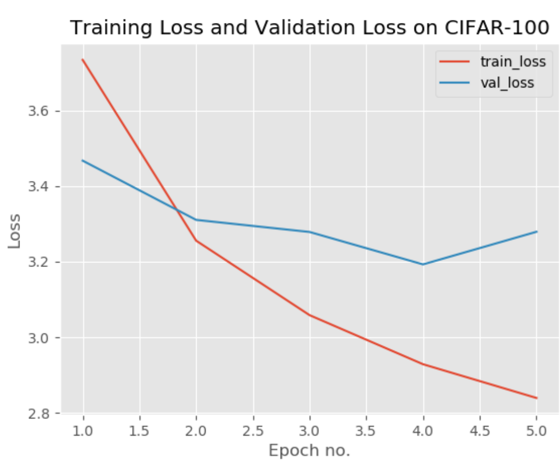
Other parameters are as follows:

Batch size: 64
Learning rate: 0.001
Optimizer: Adam

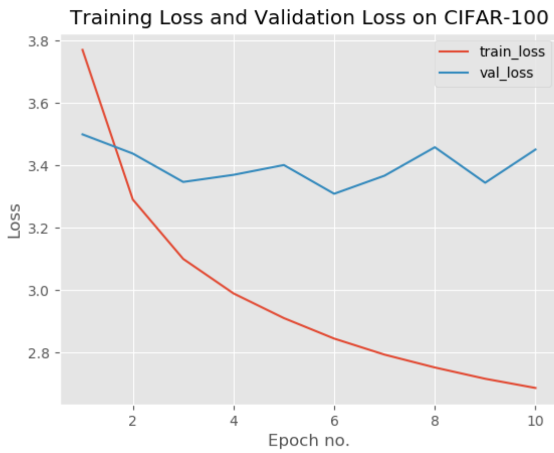
No of epochs	Rank 1 accuracy	Rank 5 accuracy
5	24.8	50.12
10	23.9	49.49
20	26.6	52.71

Plots:

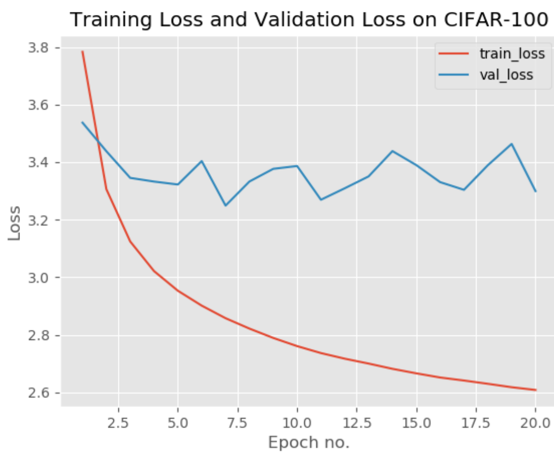
For 5 Epochs:



For 10 Epochs:



For 20 Epochs:



Discussion:

- As you can see from the plots as you went on increasing no of epochs validation loss stayed pretty much the same but the training loss decreases.
- This means that network started to overfit the training data
- As validation loss kept on fluctuating, it means that the network is no longer learning, and we need to try other things.
- But as you can see from the table, rank 1 accuracy and rank 5 accuracy were observed highest when trained for 20 epochs.
- Though we are training for certain no of epochs, accuracy was not always highest at the end of 5, 10, 20^{th} epochs. The highest accuracy occurs somewhere in the middle as validation loss is highly fluctuating which you can see from the plot

Increasing the batch size without changing any other in the architecture

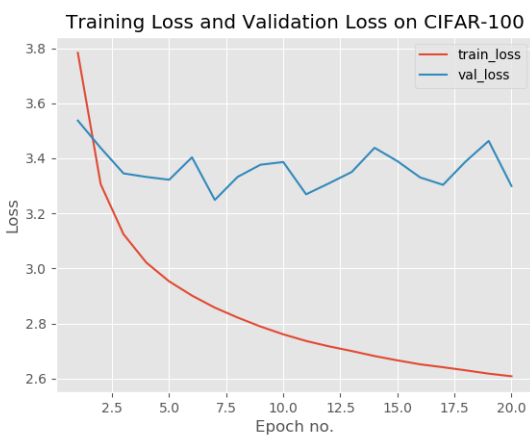
Other parameters are as follows:

Batch size: 64
Learning rate: 0.001
Optimizer: Adam
Epochs: 20

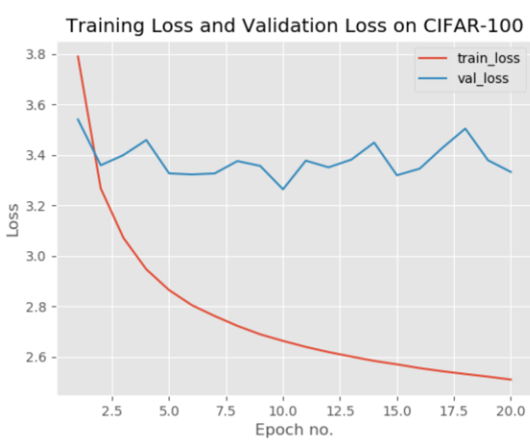
Batch size	Rank 1 accuracy	Rank 5 accuracy
64	26.6	52.71
128	26.5	53.14

Plots:

Batch Size 64:



Batch Size 128:



Discussion:

- As you can see from the table, increasing the batch size from 64 to 128 did not have much effect in rank accuracy.
- But rank 5 accuracies went by very slightly by 0.5%.
- Time it took to train was little less with larger batch size than with smaller batch size.
- As you can see from the plots, the validation loss was still fluctuating with no of epochs.
- Though we are training for certain no of epochs, accuracy was not always highest at the end of 20^{th} epochs highest observed validation accuracy sometimes occurs somewhere in the middle as validation loss is highly fluctuating which you can see from the plot

Changing the no of nodes in the last but one fully connected layer

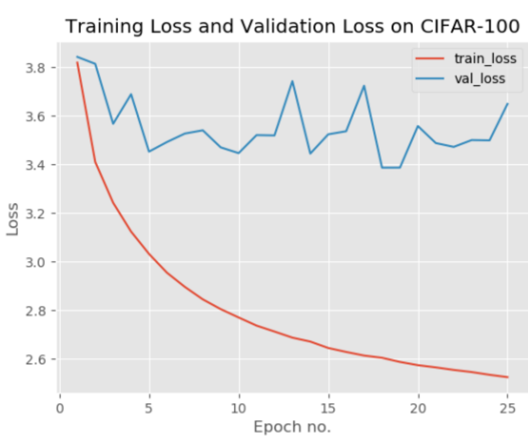
Other parameters are as follows:

Batch size: 128
Learning rate: 0.001
Optimizer: Adam
Epochs: 25

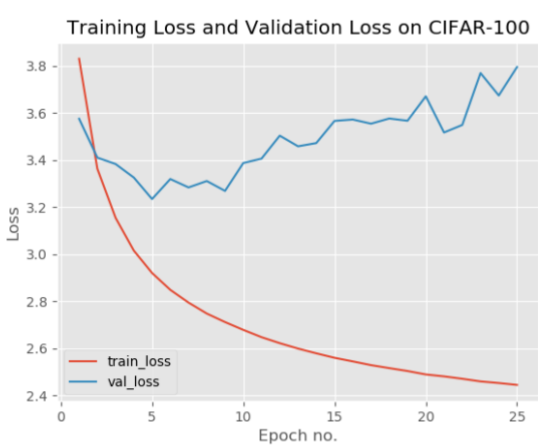
Nodes in last but one FC	Rank 1 accuracy	Rank 5 accuracy
84	22.4	48.01
100	23.1	48.58
200	24.7	51.19

Plots:

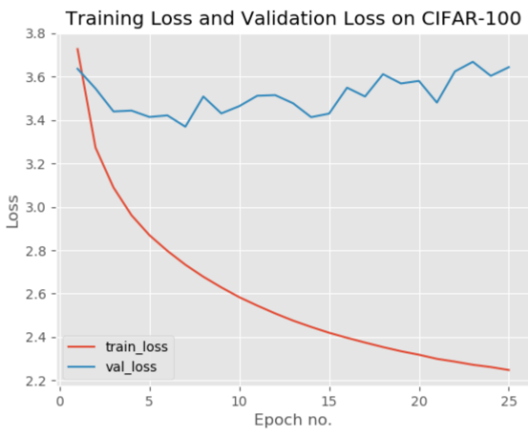
Nodes in last but one FC layer = 84:



Nodes in last but one FC layer = 100:



Nodes in last but one FC layer = 120:



Discussion:

- As you can see from the table, increasing the no of nodes in last one FC layer from 84 to 200 leads to increase in rank 1 accuracy from 22.4% to 24.7%
- Rank 5 accuracies also went from 48% to 51%.
- As you can see from the plots, the validation loss was still fluctuating with no of epochs.
- I observe a strange thing that the validation loss is going up when we increase the nodes in the FC layer. This is not an impossible situation and I discussed about this with TA(Yiqi). I suspect that network is trying to get the things right that were predicted as false, but the values are really close which in turns might throw some values that are predicted as wrong even farther. Also, we need to keep in mind that we are trying to optimize training loss and not the validation loss and since the training loss is going down and also validation accuracy going up it should be fine.
- I have also read about this on stack overflow and here is the link: https://stats.stackexchange.com/questions/282160/how-is-it-possible-that-validation-loss-is-increasing-while-validation-accuracy
- Though we are training for certain no of epochs, accuracy was not always highest at the end of 25th epochs. The highest observed validation accuracy sometimes occurs somewhere in the middle as validation loss is highly fluctuating which you can see from the plot

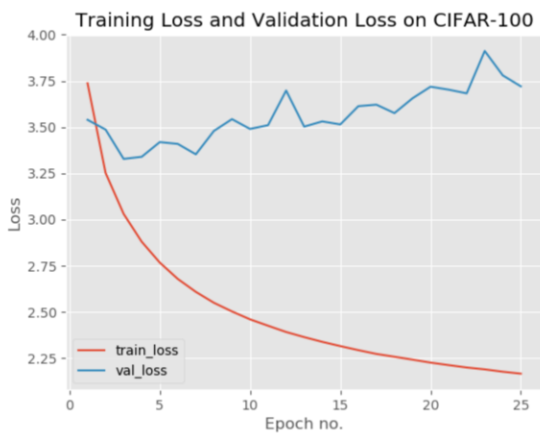
Adding one more conv layer to original architecture and changing the no of filters by still having last one FC layer size as 200

Batch size: 128
Learning rate: 0.001
Optimizer: Adam
Epochs: 25

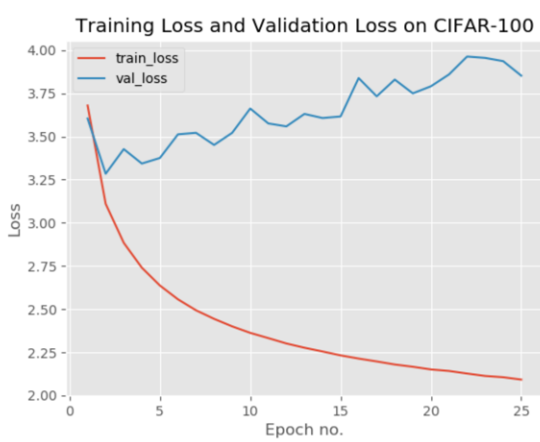
Conv Layer 1 no of Filters	Conv layer 2 no of filters	Rank 1 accuracy	Rank 5 accuracy
6	16	0.261	51.61
10	25	0.268	52.73
25	50	0.290	56.12

Plots:

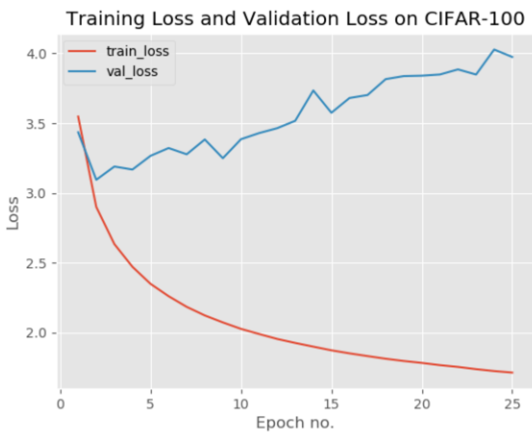
For 6,16 and 200 in last FC Layer:



For 10, 25 and 200 in last FC Layer:



For 25, 50 and 200 in last FC Layer:



Discussion:

- As you can see from the table, increasing the no filters in first and second conv layers lead to increase in rank 1 accuracy from 26% to 29%. These are one the highest accuracies observed till now.
- Rank 5 accuracies also went from 51% to 56%. These are one the highest accuracies observed till now.
- So, as we increase the no of filters the network started to learn more than overfit.
- As you can see from the plots, the validation loss was still fluctuating with no of epochs.
- Surprisingly validation accuracy went on increasing especially in second and third plots.
- I observe a strange thing that the validation loss is going up when we increase the nodes in the FC layer. This is not an impossible situation and I discussed about this with TA(Yiqi). I suspect that network is trying to get the things right that were predicted as false, but the values are really close which in turns might throw some values that are predicted as wrong even farther. Also, we need to keep in mind that we are trying to optimize training loss and not the validation loss and since the training loss is going down and also validation accuracy going up it should be fine.
- I have also read about this on stack overflow and here is the link: https://stats.stackexchange.com/questions/282160/how-is-it-possible-that-validation-loss-is-increasing-while-validation-accuracy
- Though we are training for certain no of epochs, accuracy was not always highest at the end of 25th epochs. highest observed validation accuracy sometimes occurs in the middle as validation loss is highly fluctuating which you can see from the plot.

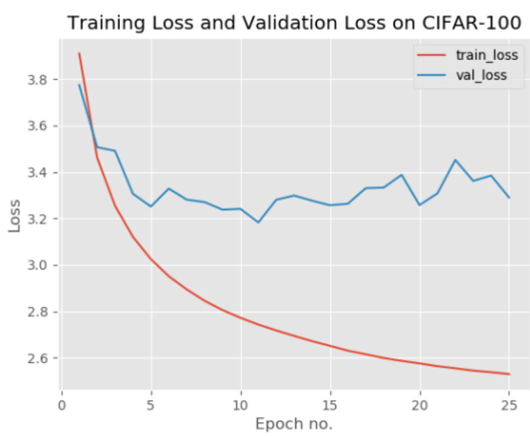
Adding one more conv layer to the original architecture and changing the size of filters in different layers:

Batch size: 128
Learning rate: 0.001
Optimizer: Adam
Epochs: 25
No of filters in conv layer 1: 6
No of filters in conv layer 2: 16
No of filters in conv layer 3: 25

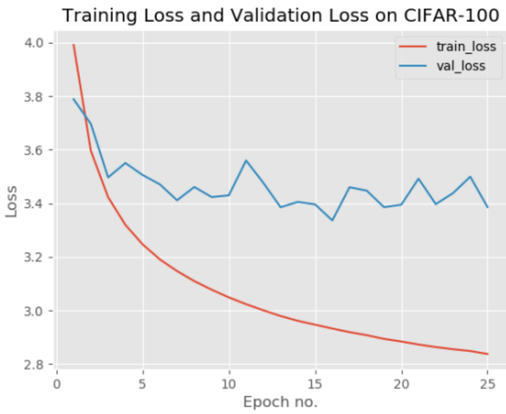
size of filter in conv 1	**size of filter in conv 2**	**size of filter in conv 3**	**Rank 1 Accuracy**	**Rank 5 Accuracy**
5	3	1	27.5	54.2
1	5	3	23.7	49.12

Plots:

For 5,3,1 size conv filters in conv1, con2, conv3



For 1, 5, 3 size conv filters in conv1, con2, conv3



Discussion:

- As you can see from the table, adding one more conv layer to the original architecture and changing the size of filters gave one of the best accuracies observed till now in case 1(row 1 of table) with rank 1 accuracy of 27.5 %
- rank 5 accuracy of 54.2% was also a good number
- However, a different combination of numbers (row 2 in table) did not give us good numbers.
- So, this means changing the size of filters does not always guaranteed an increase in accuracy and you need to be careful in doing this.
- As you can see from the plots, the validation loss was still fluctuating with no of epochs.
- Though we are training for certain no of epochs, accuracy was not always highest at the end of 25th epochs. highest observed validation accuracy sometimes occurs somewhere in the middle as validation loss is highly fluctuating which you can see from the plot

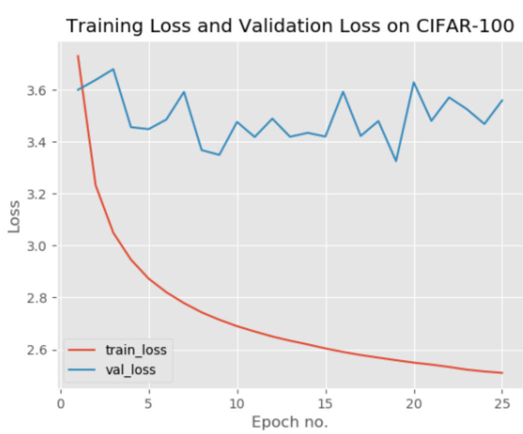
Adding batch normalization after RELU in both conv layers vs classic LeNet architecture

Batch size: 64
Learning rate: 0.001
Optimizer: Adam
Epochs: 25

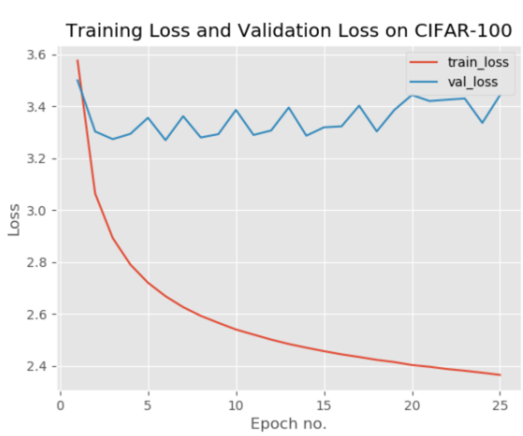
Batch Normalization Included	Rank 1 accuracy	Rank 5 accuracy
No	24.8	49.63
Yes	26.1	52.62

Plots:

Without Batch Normalization



With Batch Normalization



Discussion:

- As you can see from the table, adding batch normalization to the original architecture increased the accuracy by about 2% when compared to the one without batch normalization.
- rank 5 accuracy also went by 3%.
- As you can see from the plots, the validation loss was still fluctuating with no of epochs.
- However, validation loss was fluctuating less with batch normalization than the one without batch normalization.
- Though we are training for certain no of epochs, accuracy was not always highest at the end of 25^{th} epochs. The highest accuracy occurs somewhere in the middle as validation loss is highly fluctuating which you can see from the plot

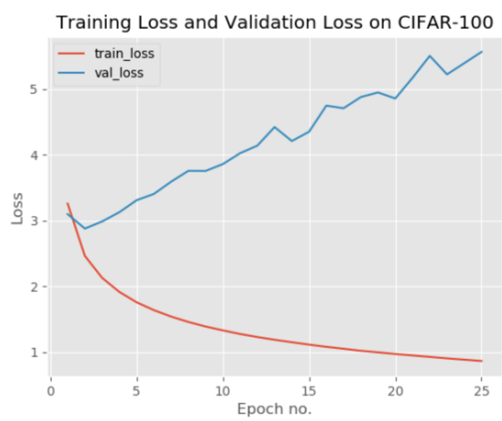
Combining best of all till now.

Other parameters are as follows:

Batch size: 128
Epochs: 25
No of filters in conv layer 1: 25
Size of filter in conv layer 1: 1
No of filters in conv layer 1: 50
Size of filter in conv layer 1: 5
No of filters in conv layer 1: 75
Size of filter in conv layer 1: 5
Size of last but one FC layer : 200

Rank 1 accuracy: 30.9
Rank 5 accuracy: 57.26

Plot:



Discussion

- The highest test accuracy of 30% was observed in any of the experiments performed till now.
- Highest rank 5 accuracy of 55.78% was observed in any of the experiments performed till now.
- Surprisingly the validation loss kept on increasing with the no of epochs.
- Though we are training for certain no of epochs, accuracy was not always highest at the end of 25th epochs. The highest accuracy occurs somewhere in the middle as validation loss is highly fluctuating which you can see from the plot.
- I observe a strange thing that the validation loss is going up when we increase the nodes in the FC layer. This is not an impossible situation and I discussed about this with TA(Yiqi). I suspect that network is trying to get the things right that were predicted as false, but the values are really close which in turns might throw some values that are predicted as wrong even farther. Also, we need to keep in mind that we are trying to optimize training loss and not the validation loss and since the training loss is going down and also validation accuracy going up it should be fine.
- I have also read about this on stack overflow and here is the link:

https://stats.stackexchange.com/questions/282160/how-is-it-possible-that-validation-loss-is-increasing-while-validation-accuracy.

Appendix:

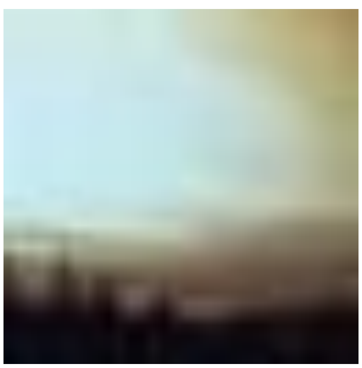
Data Augmentation is done on the training dataset in all the experiments performed.

Some correctly classified examples include:

Correct label: Sea

Correct label: Camel

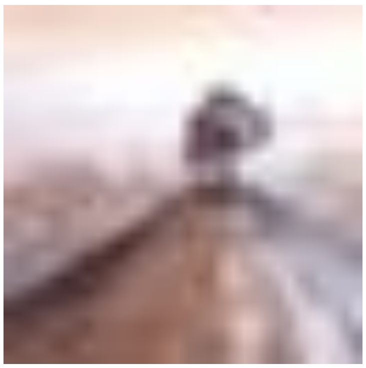
Correct label: butterfly



Correct label: Cloud

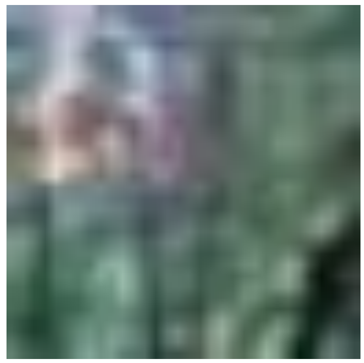
Correct label: Apple

Some incorrectly classified examples include:



Correct label: Snake.

Predicted label: Mountain



Correct label: Tiger

Predicted label: Forest

Correct label: mouse

Predicted label: seal

Correct label: Mushroom

Predicted label: Pickup truck

Correct label: Tulip

Predicted label: bee

Code:

Basic one

```
import tensorflow as tf
import argparse
import random
import pickle
import cv2
import numpy as np
import matplotlib
matplotlib.use("Agg")
from tensorflow.contrib.layers import flatten
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.utils import shuffle
import matplotlib.pyplot as plt

ap = argparse.ArgumentParser()
ap.add_argument("-e","--epochs",required = True, type = str, help = "Enter the no of epochs")
ap.add_argument("-b","--batch_size",required = True, type = str, help = "Enter the batch size")
ap.add_argument("-p","--plot",required = True, type = str, help = "Enter the path to the plot")
args = vars(ap.parse_args())

def randomize(dataset, labels):
    permutation = np.random.permutation(labels.shape[0])
    shuffled_dataset = dataset[permutation, :, :]
    shuffled_labels = labels[permutation]
    return shuffled_dataset, shuffled_labels

def one_hot_encode(np_array):
    return (np.arange(100) == np_array[:,None]).astype(np.float32)

def reformat_data1(dataset, labels, image_width, image_height, image_depth):
    grayscale = 0.21*dataset[:,0:1024] + 0.72*dataset[:,1024:2048] + 0.07*dataset[:,2048:3072]

    np_dataset_ = np.array([np.array(image_data).reshape(image_width, image_height, image_depth) for
image_data in grayscale])
    np_labels_ = one_hot_encode(np.array(labels, dtype=np.float32))
    np_dataset, np_labels = randomize(np_dataset_, np_labels_)
    return np_dataset, np_labels

def reformat_data2(dataset, labels, image_width, image_height, image_depth):
    grayscale = 0.21*dataset[:,0:1024] + 0.72*dataset[:,1024:2048] + 0.07*dataset[:,2048:3072]
```

```
    np_dataset_ = np.array([np.array(image_data).reshape(image_width, image_height, image_depth) for
image_data in grayscale])
    np_labels_ = one_hot_encode(np.array(labels, dtype=np.float32))
    np_dataset, np_labels = np_dataset_, np_labels_
    return np_dataset, np_labels

def reformat_data(dataset, labels, image_width, image_height, image_depth):
    grayscale = 0.21*dataset[:,0:1024] + 0.72*dataset[:,1024:2048] + 0.07*dataset[:,2048:3072]
    print("\ngrayscale shape is :"+str(grayscale.shape))
    cv2.imwrite("train2.png",np.array(grayscale[1,:]).reshape(32,32,1)) # .astype("uint8"))
    cv2.imwrite("train1.png",np.transpose(np.reshape(dataset[1,:],(3,32,32)),(1,2,0)))
#.transpose(2,1,0).astype("uint8"))
    np_dataset_ = np.array([np.array(image_data).reshape(image_width, image_height, image_depth) for
image_data in grayscale])
    print("np_dataset_ shape is :"+str(np_dataset_.shape))
    #np_dataset_1 = np.array([np.array(cv2.resize(x,(int(x.shape[0]*1.1),int(x.shape[1]*1.1)))).reshape(35,
35, image_depth) for x in np_dataset_])
    temp_dataset = []
    temp_labels = []
    for x,y in zip(np_dataset_,labels):
        temp_d = np.array(cv2.resize(x,(36,36))).reshape(36, 36, image_depth)
        temp_l = y
        top_left = temp_d[0:32,0:32]
        temp_dataset.append(top_left)
        temp_labels.append(temp_l)
        random_number = random.uniform(0,1)
        if random_number>0.5:
            flip_top_left = cv2.flip(top_left,1).reshape(32,32,1)
            temp_dataset.append(flip_top_left)
            temp_labels.append(temp_l)
        top_right = temp_d[0:32,4:]
        temp_dataset.append(top_right)
        temp_labels.append(temp_l)
        random_number = random.uniform(0,1)
        if random_number>0.5:
            flip_top_right = cv2.flip(top_right,1).reshape(32,32,1)
            temp_dataset.append(flip_top_right)
            temp_labels.append(temp_l)
        bottom_left = temp_d[4:,0:32]
        temp_dataset.append(bottom_left)
        temp_labels.append(temp_l)
        random_number = random.uniform(0,1)
        if random_number>0.5:
            flip_bottom_left = cv2.flip(bottom_left,1).reshape(32,32,1)
            temp_dataset.append(flip_bottom_left)
            temp_labels.append(temp_l)
        bottom_right = temp_d[4:,4:]
        temp_dataset.append(bottom_right)
        temp_labels.append(temp_l)
        random_number = random.uniform(0,1)
        if random_number>0.5:
```

```
            flip_bottom_right = cv2.flip(bottom_right,1).reshape(32,32,1)
            temp_dataset.append(flip_bottom_right)
            temp_labels.append(temp_l)
        center = temp_d[2:34,2:34]
        temp_dataset.append(center)
        temp_labels.append(temp_l)
        random_number = random.uniform(0,1)
        if random_number>0.5:
            flip_center = cv2.flip(center,1).reshape(32,32,1)
            temp_dataset.append(flip_center)
            temp_labels.append(temp_l)
    np_dataset_ = np.array(temp_dataset)
    print("temp dataset shape is :"+str(np_dataset_.shape))
    np_labels_ = one_hot_encode(np.array(temp_labels, dtype=np.float32))
    np_dataset, np_labels = randomize(np_dataset_, np_labels_)
    return np_dataset, np_labels

# tf.truncated_normal outputs random values from a truncated normal distribution.
# Genereated values follow a normal distribution with specified mean and standard deviation.
# Initialize weights variable with random values.
def init_weight(shape):
    #w = tf.truncated_normal(shape=shape, mean = 0, stddev = 0.1)
    initializer = tf.contrib.layers.xavier_initializer()
    return tf.Variable(initializer(shape))

# Initialize the bias variable with zeros.
def init_bias(shape):
    b = tf.zeros(shape)
    return tf.Variable(b)

def LeNet(x):
    # name:      conv5-6
    # structure: Input = 32x32x1. Output = 28x28x6.
    # weights:   (5*5*1+1)*6
    # connections: (28*28*5*5+28*28)*6
    conv1_W = init_weight((5,5,1,6))
    conv1_b = init_bias(6)
    conv1   = tf.nn.conv2d(x, conv1_W, strides=[1, 1, 1, 1], padding='VALID') + conv1_b
    conv1 = tf.nn.relu(conv1)

    #Input = 28x28x6. Output = 14x14x6.
    conv1 = tf.nn.max_pool(conv1, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='VALID')

    #conv5-16
    #input 14x14x6 Output = 10x10x16.
    #weights: (5*5*6+1)*16 ---real Lenet-5 is (5*5*3+1)*6+(5*5*4+1)*9+(5*5*6+1)*1
    conv2_W = init_weight((5, 5, 6, 16))
    conv2_b = init_bias(16)
    conv2   = tf.nn.conv2d(conv1, conv2_W, strides=[1, 1, 1, 1], padding='VALID') + conv2_b
    conv2 = tf.nn.relu(conv2)
```

```
    #Input = 10x10x16. Output = 5x5x16.
    conv2 = tf.nn.max_pool(conv2, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='VALID')

    #Input = 5x5x16. Output = 400.
    fc0   = flatten(conv2)

    #Input = 400. Output = 120.
    fc1_W = init_weight((400,120))
    fc1_b = init_bias(120)
    fc1   = tf.matmul(fc0, fc1_W) + fc1_b
    fc1    = tf.nn.relu(fc1)

    #Input = 120. Output = 84.
    fc2_W  = init_weight((120,84))
    fc2_b  = init_bias(84)
    fc2    = tf.matmul(fc1, fc2_W) + fc2_b
    fc2 = tf.nn.relu(fc2)

    #Input = 84. Output = 100.
    fc3_W  = init_weight((84,100))
    fc3_b  = init_bias(100)
    logits = tf.matmul(fc2, fc3_W) + fc3_b

    return logits

with open('cifar-100-python/train','rb') as f:
    c100_training_dict = pickle.load(f,encoding='bytes')

print("\n###################################################")
print("\nSome meta information")
c100_training_dataset, c100_training_labels = c100_training_dict[b'data'],
c100_training_dict[b'fine_labels']
print("\nNo of training examples in c100_training_dataset are :"+str(len(c100_training_dataset)))

c100_training_dataset,c100_validation_dataset, c100_training_labels, c100_validation_labels =
c100_training_dataset[:40000],c100_training_dataset[40000:],c100_training_labels[:40000],c100_trainin
g_labels[40000:]

training_dataset_cifar1001, training_labels_cifar100 = reformat_data(c100_training_dataset,
c100_training_labels, 32, 32, 1)
#print("\nType of training_dataset_cifar100 is :"+str(type(training_dataset_cifar100)))
print("training_dataset_cifar100 shape is :"+str(training_dataset_cifar1001.shape))

# scale the raw pixel intensities to the range [0, 1]
training_dataset_cifar100 = np.array(training_dataset_cifar1001, dtype="float") / 255.0

# apply mean subtraction to the data
mean = np.mean(training_dataset_cifar100, axis=0)
training_dataset_cifar100 -= mean

#cut_off = int(training_dataset_cifar100.shape[0]*0.8)
```

```
#train_dataset, validation_dataset = training_dataset_cifar100[:cut_off,:], training_dataset_cifar100[cut_off:,:]
#print("\nType of train_dataset :"+str(type(train_dataset)))
train_dataset = training_dataset_cifar100
print("train_dataset shape is :"+str(train_dataset.shape))

validation_dataset, validation_labels = reformat_data1(c100_validation_dataset, c100_validation_labels, 32, 32, 1)
#print("\nType of validation_dataset is :"+str(type(validation_dataset)))
print("validation_dataset shape is :"+str(validation_dataset.shape))
validation_dataset =  np.array(validation_dataset, dtype="float") / 255.0
validation_dataset-=mean

train_labels = training_labels_cifar100
print("train_labels shape is :"+str(train_labels.shape))
print("validation_labels shape is :"+str(validation_labels.shape))

with open('cifar-100-python/test','rb') as f:
   c100_test_dict = pickle.load(f,encoding='bytes')

c100_test_dataset, c100_test_labels = c100_test_dict[b'data'], c100_test_dict[b'fine_labels']
print("\nNo of testing examples in c100_test_dataset are :"+str(len(c100_test_dataset)))

t_labels = c100_test_labels
test_dataset1, test_labels = reformat_data2(c100_test_dataset, c100_test_labels, 32, 32, 1)
#print("\nType of test_dataset is :"+str(type(test_dataset)))
print("test_dataset shape is :"+str(test_dataset1.shape))

print("\n##################################################")

# scale the raw pixel intensities to the range [0, 1]
test_dataset = np.array(test_dataset1, dtype="float") / 255.0
test_dataset-=mean

EPOCHS = int(args["epochs"])
BATCH_SIZE = int(args["batch_size"])

x = tf.placeholder(tf.float32, (None, 32, 32, 1))
y = tf.placeholder(tf.int32, (None))
one_hot_y = y #tf.one_hot(y, 100)

rate = 0.001

logits = LeNet(x)
cross_entropy = tf.nn.softmax_cross_entropy_with_logits(logits=logits, labels=one_hot_y)

loss_operation = tf.reduce_mean(cross_entropy)
optimizer = tf.train.AdamOptimizer(learning_rate = rate)
training_operation = optimizer.minimize(loss_operation)
```

```
correct_prediction = tf.equal(tf.argmax(logits, 1), tf.argmax(one_hot_y, 1))
accuracy_operation = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
saver = tf.train.Saver()

prediction_values = tf.argmax(logits,1)
#prediction_values_5 = tf.nn.top_k(logits,5)
#prediction_values_5_indices = prediction_values_5.indices
prediction_values_5 = logits

X_train,y_train,X_validation,y_validation,X_test,y_test = train_dataset, train_labels, validation_dataset,
validation_labels, test_dataset, test_labels
X_train, y_train = shuffle(X_train, y_train)

#validation_losses = []
validation_loss = 0

def evaluate(X_data, y_data):
    total_loss = 0
    num_examples = len(X_data)
    total_accuracy = 0
    sess = tf.get_default_session()
    for offset in range(0, num_examples, BATCH_SIZE):
        batch_x, batch_y = X_data[offset:offset+BATCH_SIZE], y_data[offset:offset+BATCH_SIZE]
        accuracy,loss_val = sess.run([accuracy_operation,loss_operation], feed_dict={x: batch_x, y:
batch_y})
        total_accuracy += (accuracy * len(batch_x))
        total_loss+=loss_val
    no_of_batches = num_examples/BATCH_SIZE
    total_loss = total_loss/no_of_batches
    return total_accuracy / num_examples,total_loss

with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    num_examples = len(X_train)
    train_loss = 0
    train_losses = []
    train_accuracies = []
    validation_losses = []
    validation_accuracies = []
    print("[INFO] Training...")
    print()
    for i in range(EPOCHS):
        X_train, y_train = shuffle(X_train, y_train)
        for offset in range(0, num_examples, BATCH_SIZE):
            end = offset + BATCH_SIZE
            batch_x, batch_y = X_train[offset:end], y_train[offset:end]
            _,loss_val = sess.run([training_operation,loss_operation], feed_dict={x: batch_x, y: batch_y})
            train_loss+=loss_val
        no_of_batches = num_examples/BATCH_SIZE
        train_loss = train_loss/no_of_batches
        train_losses.append(train_loss)
```

```
        validation_accuracy,validation_loss_for_this_epoch = evaluate(X_validation, y_validation)
        validation_losses.append(validation_loss_for_this_epoch)
        validation_accuracies.append(validation_accuracy)
        print("EPOCH {} ...".format(i+1))
        print("Training loss is :"+str(train_loss))
        print("Validation loss = {:.3f}".format(validation_loss_for_this_epoch))
        print("Validation Accuracy = {:.3f}".format(validation_accuracy))
        print()

    saver.save(sess, './lenet')
    print("Model saved")

epoch_nums = []
for i in range(1,EPOCHS+1):
    epoch_nums.append(i)

plt.style.use("ggplot")
plt.figure()
plt.plot(epoch_nums, train_losses,label="train_loss")
plt.plot(epoch_nums, validation_losses,label="val_loss")
#plt.plot(epoch_nums, epoch_losses,label="train_acc")
#plt.plot(epoch_nums, validation_losses,label="val_acc")
plt.title("Training Loss and Validation Loss on CIFAR-100")
plt.xlabel("Epoch no.")
plt.ylabel("Loss")
plt.legend()
plt.savefig(args["plot"])


with tf.Session() as sess:
    saver.restore(sess, tf.train.latest_checkpoint('.'))
    test_accuracy,test_loss = evaluate(X_test, y_test)
    #print("test accuracy is : "+str(test_accuracy))
    print("Test Accuracy = {:.3f}".format(test_accuracy))
    predict = sess.run(prediction_values, feed_dict={x: X_test})
    #predict_5 = sess.run(prediction_values_5,feed_dict={x: X_test})
    predict_5 = sess.run(prediction_values_5,feed_dict={x:X_test})
    boolean_top_5 = tf.nn.in_top_k(predictions=predict_5,
targets=tf.convert_to_tensor(t_labels,dtype=tf.int32), k=5).eval()

f = open('cifar-100-python/meta', 'rb')
datadict = pickle.load(f)#, encoding='bytes')
f.close()

fine_labels = datadict['fine_label_names']
coarse_labels = datadict['coarse_label_names']
model = LeNet

print("Predict is :")
print(predict)
#print("Len of predict is :"+str(len(predict)))
```

```
#print("Len of test_label shape is:"+str(len(test_labels)))
#print("Type of test labels is :"+str(type(test_labels)))
#print("Type of predict is :"+str(type(predict)))
#print("Test labels")
#unique, counts = np.unique(test_labels, return_counts=True)
#print(np.asarray((unique, counts)).T)
#print("predict")
#unique, counts = np.unique(predict, return_counts=True)
#print(np.asarray((unique, counts)).T)

con_mat = tf.confusion_matrix(t_labels, predictions=predict) #, num_classes=100, dtype=tf.int32,
name=None)

with tf.Session():
  #print('Confusion Matrix: \n\n',
  cm = tf.Tensor.eval(con_mat,feed_dict=None, session=None)
  print("\n#####################################################")
  print("\nConfusion Matrix is :\n")
  print(cm)
  print("\n#####################################################")

cm1 = np.zeros((10000,10000), dtype=int)
j=0
for i in boolean_top_5:
  if i == True:
    cm1[t_labels[j],t_labels[j]]+=1
    j+=1

print("\n#####################################################")
print("\nClassification accuracy for each class are :\n")

for i in range(100):
  print(fine_labels[i]+" : "+str(cm[i,i])+" %")

print("\n#####################################################")

coarse_label_names_to_count = {}
coarse_label_names_to_count['aquatic mammals'] = 0
coarse_label_names_to_count['fish'] = 0
coarse_label_names_to_count['flowers'] = 0
coarse_label_names_to_count['food containers'] = 0
coarse_label_names_to_count['fruit and vegetables'] = 0
coarse_label_names_to_count['household electrical devices'] = 0
coarse_label_names_to_count['household furniture'] = 0
coarse_label_names_to_count['insects'] = 0
coarse_label_names_to_count['large carnivores'] = 0
coarse_label_names_to_count['large man-made outdoor things'] = 0
coarse_label_names_to_count['large natural outdoor scenes'] = 0
coarse_label_names_to_count['large omnivores and herbivores'] = 0
coarse_label_names_to_count['medium-sized mammals'] = 0
coarse_label_names_to_count['non-insect invertebrates'] = 0
```

```
coarse_label_names_to_count['people'] = 0
coarse_label_names_to_count['small mammals'] = 0
coarse_label_names_to_count['trees'] = 0
coarse_label_names_to_count['reptiles'] = 0
coarse_label_names_to_count['vehicles 1'] = 0
coarse_label_names_to_count['vehicles 2'] = 0

no_of_classes_that_each_coarse_label_has = 5

other = 0
for i in range(100):
  if fine_labels[i] in ['beaver', 'dolphin', 'otter', 'seal','whale']:
     coarse_label_names_to_count['aquatic mammals']+=cm[i,i]
  elif fine_labels[i] in ['aquarium_fish', 'flatfish', 'ray', 'shark', 'trout']:
     coarse_label_names_to_count['fish']+=cm[i,i]
  elif fine_labels[i] in ['orchid', 'poppy', 'rose', 'sunflower', 'tulip']:
     coarse_label_names_to_count['flowers']+=cm[i,i]
  elif fine_labels[i] in ['bottle', 'bowl', 'can', 'cup', 'plate']:
     coarse_label_names_to_count['food containers']+=cm[i,i]
  elif fine_labels[i] in ['apple', 'mushroom','orange', 'pear', 'sweet_pepper']:
     coarse_label_names_to_count['fruit and vegetables']+=cm[i,i]
  elif fine_labels[i] in ['clock', 'keyboard', 'lamp', 'telephone', 'television']:
     coarse_label_names_to_count['household electrical devices']+=cm[i,i]
  elif fine_labels[i] in ['bed', 'chair', 'couch', 'table', 'wardrobe']:
     coarse_label_names_to_count['household furniture']+=cm[i,i]
  elif fine_labels[i] in ['bee', 'beetle', 'butterfly', 'caterpillar', 'cockroach']:
     coarse_label_names_to_count['insects']+=cm[i,i]
  elif fine_labels[i] in ['bear', 'leopard', 'lion', 'tiger', 'wolf']:
     coarse_label_names_to_count['large carnivores']+=cm[i,i]
  elif fine_labels[i] in ['bridge', 'castle', 'house', 'road', 'skyscraper']:
     coarse_label_names_to_count['large man-made outdoor things']+=cm[i,i]
  elif fine_labels[i] in ['cloud', 'forest', 'mountain', 'plain', 'sea']:
     coarse_label_names_to_count['large natural outdoor scenes']+=cm[i,i]
  elif fine_labels[i] in ['camel', 'cattle', 'chimpanzee', 'elephant', 'kangaroo']:
     coarse_label_names_to_count['large omnivores and herbivores']+=cm[i,i]
  elif fine_labels[i] in ['fox', 'porcupine', 'possum', 'raccoon', 'skunk']:
     coarse_label_names_to_count['medium-sized mammals']+=cm[i,i]
  elif fine_labels[i] in ['crab', 'lobster', 'snail', 'spider', 'worm']:
     coarse_label_names_to_count['non-insect invertebrates']+=cm[i,i]
  elif fine_labels[i] in ['baby', 'boy', 'girl', 'man', 'woman']:
     coarse_label_names_to_count['people']+=cm[i,i]
  elif fine_labels[i] in ['crocodile', 'dinosaur', 'lizard', 'snake', 'turtle']:
     coarse_label_names_to_count['reptiles']+=cm[i,i]
  elif fine_labels[i] in ['hamster', 'mouse', 'rabbit', 'shrew', 'squirrel']:
     coarse_label_names_to_count['small mammals']+=cm[i,i]
  elif fine_labels[i] in ['maple_tree', 'oak_tree', 'palm_tree', 'pine_tree', 'willow_tree']:
     coarse_label_names_to_count['trees']+=cm[i,i]
  elif fine_labels[i] in ['bicycle', 'bus', 'motorcycle', 'pickup_truck', 'train']:
     coarse_label_names_to_count['vehicles 1']+=cm[i,i]
  elif fine_labels[i] in ['lawn_mower', 'rocket', 'streetcar', 'tank', 'tractor']:
     coarse_label_names_to_count['vehicles 2']+=cm[i,i]
```

```
  else:
      other+=1

#print("\nNo of examples not classified into any of the super classes are :"+str(other))

#print("\n#####################################################")

print("\nClassification accuracy for each Super class are as follows :\n")
for i in coarse_label_names_to_count.keys():
  print(i + " : "+str(coarse_label_names_to_count[i]/5) + " %")

print("\n########################################################\n")

print(coarse_label_names_to_count)

print("\n########################################################")
print("\nRank 5 Classification accuracy for each class are :\n")

for i in range(100):
  print(fine_labels[i]+" : "+str(cm1[i,i])+" %")

coarse_label_names_to_count1 = {}
coarse_label_names_to_count1['aquatic mammals'] = 0
coarse_label_names_to_count1['fish'] = 0
coarse_label_names_to_count1['flowers'] = 0
coarse_label_names_to_count1['food containers'] = 0
coarse_label_names_to_count1['fruit and vegetables'] = 0
coarse_label_names_to_count1['household electrical devices'] = 0
coarse_label_names_to_count1['household furniture'] = 0
coarse_label_names_to_count1['insects'] = 0
coarse_label_names_to_count1['large carnivores'] = 0
coarse_label_names_to_count1['large man-made outdoor things'] = 0
coarse_label_names_to_count1['large natural outdoor scenes'] = 0
coarse_label_names_to_count1['large omnivores and herbivores'] = 0
coarse_label_names_to_count1['medium-sized mammals'] = 0
coarse_label_names_to_count1['non-insect invertebrates'] = 0
coarse_label_names_to_count1['people'] = 0
coarse_label_names_to_count1['small mammals'] = 0
coarse_label_names_to_count1['trees'] = 0
coarse_label_names_to_count1['reptiles'] = 0
coarse_label_names_to_count1['vehicles 1'] = 0
coarse_label_names_to_count1['vehicles 2'] = 0

other1 = 0
overall_top_5_accuracy = 0

for i in range(100):
  if fine_labels[i] in ['beaver', 'dolphin', 'otter', 'seal','whale']:
      coarse_label_names_to_count1['aquatic mammals']+=cm1[i,i]
      overall_top_5_accuracy+=cm1[i,i]
  elif fine_labels[i] in ['aquarium_fish', 'flatfish', 'ray', 'shark', 'trout']:
```

```
    coarse_label_names_to_count1['fish']+=cm1[i,i]
    overall_top_5_accuracy+=cm1[i,i]
elif fine_labels[i] in ['orchid', 'poppy', 'rose', 'sunflower', 'tulip']:
    coarse_label_names_to_count1['flowers']+=cm1[i,i]
    overall_top_5_accuracy+=cm1[i,i]
elif fine_labels[i] in ['bottle', 'bowl', 'can', 'cup', 'plate']:
    coarse_label_names_to_count1['food containers']+=cm1[i,i]
    overall_top_5_accuracy+=cm1[i,i]
elif fine_labels[i] in ['apple', 'mushroom','orange', 'pear', 'sweet_pepper']:
    coarse_label_names_to_count1['fruit and vegetables']+=cm1[i,i]
    overall_top_5_accuracy+=cm1[i,i]
elif fine_labels[i] in ['clock', 'keyboard', 'lamp', 'telephone', 'television']:
    coarse_label_names_to_count1['household electrical devices']+=cm1[i,i]
    overall_top_5_accuracy+=cm1[i,i]
elif fine_labels[i] in ['bed', 'chair', 'couch', 'table', 'wardrobe']:
    coarse_label_names_to_count1['household furniture']+=cm1[i,i]
    overall_top_5_accuracy+=cm1[i,i]
elif fine_labels[i] in ['bee', 'beetle', 'butterfly', 'caterpillar', 'cockroach']:
    coarse_label_names_to_count1['insects']+=cm1[i,i]
    overall_top_5_accuracy+=cm1[i,i]
elif fine_labels[i] in ['bear', 'leopard', 'lion', 'tiger', 'wolf']:
    coarse_label_names_to_count1['large carnivores']+=cm1[i,i]
    overall_top_5_accuracy+=cm1[i,i]
elif fine_labels[i] in ['bridge', 'castle', 'house', 'road', 'skyscraper']:
    coarse_label_names_to_count1['large man-made outdoor things']+=cm1[i,i]
    overall_top_5_accuracy+=cm1[i,i]
elif fine_labels[i] in ['cloud', 'forest', 'mountain', 'plain', 'sea']:
    coarse_label_names_to_count1['large natural outdoor scenes']+=cm1[i,i]
    overall_top_5_accuracy+=cm1[i,i]
elif fine_labels[i] in ['camel', 'cattle', 'chimpanzee', 'elephant', 'kangaroo']:
    coarse_label_names_to_count1['large omnivores and herbivores']+=cm1[i,i]
    overall_top_5_accuracy+=cm1[i,i]
elif fine_labels[i] in ['fox', 'porcupine', 'possum', 'raccoon', 'skunk']:
    coarse_label_names_to_count1['medium-sized mammals']+=cm1[i,i]
    overall_top_5_accuracy+=cm1[i,i]
elif fine_labels[i] in ['crab', 'lobster', 'snail', 'spider', 'worm']:
    coarse_label_names_to_count1['non-insect invertebrates']+=cm1[i,i]
    overall_top_5_accuracy+=cm1[i,i]
elif fine_labels[i] in ['baby', 'boy', 'girl', 'man', 'woman']:
    coarse_label_names_to_count1['people']+=cm1[i,i]
    overall_top_5_accuracy+=cm1[i,i]
elif fine_labels[i] in ['crocodile', 'dinosaur', 'lizard', 'snake', 'turtle']:
    coarse_label_names_to_count1['reptiles']+=cm1[i,i]
    overall_top_5_accuracy+=cm1[i,i]
elif fine_labels[i] in ['hamster', 'mouse', 'rabbit', 'shrew', 'squirrel']:
    coarse_label_names_to_count1['small mammals']+=cm1[i,i]
    overall_top_5_accuracy+=cm1[i,i]
elif fine_labels[i] in ['maple_tree', 'oak_tree', 'palm_tree', 'pine_tree', 'willow_tree']:
    coarse_label_names_to_count1['trees']+=cm1[i,i]
    overall_top_5_accuracy+=cm1[i,i]
elif fine_labels[i] in ['bicycle', 'bus', 'motorcycle', 'pickup_truck', 'train']:
```

```
      coarse_label_names_to_count1['vehicles 1']+=cm1[i,i]
      overall_top_5_accuracy+=cm1[i,i]
   elif fine_labels[i] in ['lawn_mower', 'rocket', 'streetcar', 'tank', 'tractor']:
      coarse_label_names_to_count1['vehicles 2']+=cm1[i,i]
      overall_top_5_accuracy+=cm1[i,i]
   else:
      other1+=1

print("\nExamples not classified into any of the super classes are :"+str(other1))
print("\n####################################################")
print("\nRank 5 Classification accuracy for each Super class are as follows:\n")
for i in coarse_label_names_to_count.keys():
   print(i + " : "+str(coarse_label_names_to_count1[i]/5) + " %")
print("\n####################################################")

print("\n####################################################")
print("\nOverall rank 5 accuracy is :"+str(overall_top_5_accuracy/100))
print("\n####################################################")
```

Code (Trying out different things):

```
import tensorflow as tf
import argparse
import random
import pickle
import cv2
import numpy as np
import matplotlib
matplotlib.use("Agg")
from tensorflow.contrib.layers import flatten
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.utils import shuffle
import matplotlib.pyplot as plt

ap = argparse.ArgumentParser()
ap.add_argument("-e","--epochs",required = True, type = str, help = "Enter the no of epochs")
ap.add_argument("-b","--batch_size",required = True, type = str, help = "Enter the batch size")
ap.add_argument("-p","--plot",required = True, type = str, help = "Enter the path to the plot")
args = vars(ap.parse_args())

def randomize(dataset, labels):
   permutation = np.random.permutation(labels.shape[0])
   shuffled_dataset = dataset[permutation, :, :]
   shuffled_labels = labels[permutation]
   return shuffled_dataset, shuffled_labels

def one_hot_encode(np_array):
   return (np.arange(100) == np_array[:,None]).astype(np.float32)

def reformat_data1(dataset, labels, image_width, image_height, image_depth):
   grayscale = 0.21*dataset[:,0:1024] + 0.72*dataset[:,1024:2048] + 0.07*dataset[:,2048:3072]

   np_dataset_ = np.array([np.array(image_data).reshape(image_width, image_height, image_depth) for
image_data in grayscale])
   np_labels_ = one_hot_encode(np.array(labels, dtype=np.float32))
   np_dataset, np_labels = randomize(np_dataset_, np_labels_)
   return np_dataset, np_labels

def reformat_data2(dataset, labels, image_width, image_height, image_depth):
   grayscale = 0.21*dataset[:,0:1024] + 0.72*dataset[:,1024:2048] + 0.07*dataset[:,2048:3072]

   np_dataset_ = np.array([np.array(image_data).reshape(image_width, image_height, image_depth) for
image_data in grayscale])
   np_labels_ = one_hot_encode(np.array(labels, dtype=np.float32))
```

```
    np_dataset, np_labels = np_dataset_, np_labels_
    return np_dataset, np_labels

def reformat_data(dataset, labels, image_width, image_height, image_depth):
    grayscale = 0.21*dataset[:,0:1024] + 0.72*dataset[:,1024:2048] + 0.07*dataset[:,2048:3072]
    print("\ngrayscale shape is :"+str(grayscale.shape))
    np_dataset_ = np.array([np.array(image_data).reshape(image_width, image_height, image_depth) for
image_data in grayscale])
    print("np_dataset_ shape is :"+str(np_dataset_.shape))
    #np_dataset_1 = np.array([np.array(cv2.resize(x,(int(x.shape[0]*1.1),int(x.shape[1]*1.1)))).reshape(35,
35, image_depth) for x in np_dataset_])
    temp_dataset = []
    temp_labels = []
    for x,y in zip(np_dataset_,labels):
        temp_d = np.array(cv2.resize(x,(36,36))).reshape(36, 36, image_depth)
        temp_l = y
        top_left = temp_d[0:32,0:32]
        temp_dataset.append(top_left)
        temp_labels.append(temp_l)
        random_number = random.uniform(0,1)
        if random_number>0.5:
            flip_top_left = cv2.flip(top_left,1).reshape(32,32,1)
            temp_dataset.append(flip_top_left)
            temp_labels.append(temp_l)
        top_right = temp_d[0:32,4:]
        temp_dataset.append(top_right)
        temp_labels.append(temp_l)
        random_number = random.uniform(0,1)
        if random_number>0.5:
            flip_top_right = cv2.flip(top_right,1).reshape(32,32,1)
            temp_dataset.append(flip_top_right)
            temp_labels.append(temp_l)
        bottom_left = temp_d[4:,0:32]
        temp_dataset.append(bottom_left)
        temp_labels.append(temp_l)
        random_number = random.uniform(0,1)
        if random_number>0.5:
            flip_bottom_left = cv2.flip(bottom_left,1).reshape(32,32,1)
            temp_dataset.append(flip_bottom_left)
            temp_labels.append(temp_l)
        bottom_right = temp_d[4:,4:]
        temp_dataset.append(bottom_right)
        temp_labels.append(temp_l)
        random_number = random.uniform(0,1)
        if random_number>0.5:
            flip_bottom_right = cv2.flip(bottom_right,1).reshape(32,32,1)
            temp_dataset.append(flip_bottom_right)
            temp_labels.append(temp_l)
        center = temp_d[2:34,2:34]
        temp_dataset.append(center)
        temp_labels.append(temp_l)
```

```
        random_number = random.uniform(0,1)
        if random_number>0.5:
            flip_center = cv2.flip(center,1).reshape(32,32,1)
            temp_dataset.append(flip_center)
            temp_labels.append(temp_l)
    np_dataset_ = np.array(temp_dataset)
    print("temp dataset shape is :"+str(np_dataset_.shape))
    np_labels_ = one_hot_encode(np.array(temp_labels, dtype=np.float32))
    np_dataset, np_labels = randomize(np_dataset_, np_labels_)
    return np_dataset, np_labels

# tf.truncated_normal outputs random values from a truncated normal distribution.
# Genereated values follow a normal distribution with specified mean and standard deviation.
# Initialize weights variable with random values.
def init_weight(shape):
    #w = tf.truncated_normal(shape=shape, mean = 0, stddev = 0.1)
    initializer = tf.contrib.layers.xavier_initializer()
    return tf.Variable(initializer(shape))

# Initialize the bias variable with zeros.
def init_bias(shape):
    b = tf.zeros(shape)
    return tf.Variable(b)

def LeNet(x):
    # name:      conv5-6
    # structure: Input = 32x32x1. Output = 28x28x6.
    # weights:   (5*5*1+1)*6
    # connections: (28*28*5*5+28*28)*6
    conv1_W = init_weight((1,1,1,25))
    conv1_b = init_bias(25)
    conv1   = tf.nn.conv2d(x, conv1_W, strides=[1, 1, 1, 1], padding='VALID') + conv1_b
    conv1 = tf.nn.relu(conv1)
    conv1 = tf.layers.batch_normalization(conv1)

    #Input = 28x28x6. Output = 14x14x6.
    #conv1 = tf.nn.max_pool(conv1, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='VALID')

    #conv5-16
    #input 14x14x6 Output = 10x10x16.
    #weights: (5*5*6+1)*16 ---real Lenet-5 is (5*5*3+1)*6+(5*5*4+1)*9+(5*5*6+1)*1
    conv2_W = init_weight((5, 5, 25, 50))
    conv2_b = init_bias(50)
    conv2   = tf.nn.conv2d(conv1, conv2_W, strides=[1, 1, 1, 1], padding='VALID') + conv2_b
    conv2 = tf.nn.relu(conv2)
    conv2 = tf.layers.batch_normalization(conv2)

    #Input = 10x10x16. Output = 5x5x16.
    conv2 = tf.nn.max_pool(conv2, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='VALID')

    conv3_W = init_weight((5, 5, 50, 75))
```

```
    conv3_b = init_bias(75)
    conv3   = tf.nn.conv2d(conv2, conv3_W, strides=[1, 1, 1, 1], padding='VALID') + conv3_b
    conv3 = tf.nn.relu(conv3)
    conv3 = tf.layers.batch_normalization(conv3)

    #Input = 10x10x16. Output = 5x5x16.
    conv3 = tf.nn.max_pool(conv3, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='VALID')

    #Input = 5x5x16. Output = 400.
    fc0   = flatten(conv3)

    #Input = 400. Output = 120.
    fc1_W = init_weight((1875,120))
    fc1_b = init_bias(120)
    fc1   = tf.matmul(fc0, fc1_W) + fc1_b
    fc1    = tf.nn.relu(fc1)

    #Input = 120. Output = 200.
    fc2_W  = init_weight((120,200))
    fc2_b  = init_bias(200)
    fc2    = tf.matmul(fc1, fc2_W) + fc2_b
    fc2 = tf.nn.relu(fc2)

    #Input = 200. Output = 100.
    fc3_W  = init_weight((200,100))
    fc3_b  = init_bias(100)
    logits = tf.matmul(fc2, fc3_W) + fc3_b

    return logits

with open('cifar-100-python/train','rb') as f:
    c100_training_dict = pickle.load(f,encoding='bytes')

print("\n####################################################")
print("\nSome meta information")
c100_training_dataset, c100_training_labels = c100_training_dict[b'data'], c100_training_dict[b'fine_labels']
print("\nNo of training examples in c100_training_dataset are :"+str(len(c100_training_dataset)))

c100_training_dataset,c100_validation_dataset, c100_training_labels, c100_validation_labels = c100_training_dataset[:40000],c100_training_dataset[40000:],c100_training_labels[:40000],c100_training_labels[40000:]

print("c100_training dataset shape is :"+str(type(c100_training_dataset)))
cv2.imwrite('train1.png',np.reshape(c100_training_dataset[1,:],(32,32,3)))

training_dataset_cifar1001, training_labels_cifar100 = reformat_data(c100_training_dataset, c100_training_labels, 32, 32, 1)
#print("\nType of training_dataset_cifar100 is :"+str(type(training_dataset_cifar100)))
print("training_dataset_cifar100 shape is :"+str(training_dataset_cifar1001.shape))
```

```
# scale the raw pixel intensities to the range [0, 1]
training_dataset_cifar100 = np.array(training_dataset_cifar1001, dtype="float") / 255.0

# apply mean subtraction to the data
mean = np.mean(training_dataset_cifar100, axis=0)
training_dataset_cifar100 -= mean

#cut_off = int(training_dataset_cifar100.shape[0]*0.8)

#train_dataset, validation_dataset = training_dataset_cifar100[:cut_off,:], training_dataset_cifar100[cut_off:,:]
#print("\nType of train_dataset :"+str(type(train_dataset)))
train_dataset = training_dataset_cifar100
print("train_dataset shape is :"+str(train_dataset.shape))
print("Type of training dataset is :"+str(type(train_dataset)))

validation_dataset, validation_labels = reformat_data1(c100_validation_dataset, c100_validation_labels, 32, 32, 1)
#print("\nType of validation_dataset is :"+str(type(validation_dataset)))
print("validation_dataset shape is :"+str(validation_dataset.shape))
validation_dataset =  np.array(validation_dataset, dtype="float") / 255.0
validation_dataset-=mean

train_labels = training_labels_cifar100
print("train_labels shape is :"+str(train_labels.shape))
print("validation_labels shape is :"+str(validation_labels.shape))

with open('cifar-100-python/test','rb') as f:
   c100_test_dict = pickle.load(f,encoding='bytes')

c100_test_dataset, c100_test_labels = c100_test_dict[b'data'], c100_test_dict[b'fine_labels']
print("\nNo of testing examples in c100_test_dataset are :"+str(len(c100_test_dataset)))

t_labels = c100_test_labels
test_dataset1, test_labels = reformat_data2(c100_test_dataset, c100_test_labels, 32, 32, 1)
#print("\nType of test_dataset is :"+str(type(test_dataset)))
print("test_dataset shape is :"+str(test_dataset1.shape))

print("\n##################################################")

# scale the raw pixel intensities to the range [0, 1]
test_dataset = np.array(test_dataset1, dtype="float") / 255.0
test_dataset-=mean

EPOCHS = int(args["epochs"])
BATCH_SIZE = int(args["batch_size"])

x = tf.placeholder(tf.float32, (None, 32, 32, 1))
y = tf.placeholder(tf.int32, (None))
one_hot_y = y #tf.one_hot(y, 100)
```

```
rate = 0.001

logits = LeNet(x)
cross_entropy = tf.nn.softmax_cross_entropy_with_logits(logits=logits, labels=one_hot_y)

loss_operation = tf.reduce_mean(cross_entropy)
optimizer = tf.train.AdamOptimizer(learning_rate = rate)
training_operation = optimizer.minimize(loss_operation)

correct_prediction = tf.equal(tf.argmax(logits, 1), tf.argmax(one_hot_y, 1))
accuracy_operation = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
saver = tf.train.Saver()

prediction_values = tf.argmax(logits,1)
#prediction_values_5 = tf.nn.top_k(logits,5)
#prediction_values_5_indices = prediction_values_5.indices
prediction_values_5 = logits

X_train,y_train,X_validation,y_validation,X_test,y_test = train_dataset, train_labels, validation_dataset, validation_labels, test_dataset, test_labels
X_train, y_train = shuffle(X_train, y_train)

#validation_losses = []
validation_loss = 0

def evaluate(X_data, y_data):
    total_loss = 0
    num_examples = len(X_data)
    total_accuracy = 0
    sess = tf.get_default_session()
    for offset in range(0, num_examples, BATCH_SIZE):
        batch_x, batch_y = X_data[offset:offset+BATCH_SIZE], y_data[offset:offset+BATCH_SIZE]
        accuracy,loss_val = sess.run([accuracy_operation,loss_operation], feed_dict={x: batch_x, y: batch_y})
        total_accuracy += (accuracy * len(batch_x))
        total_loss+=loss_val
    no_of_batches = num_examples/BATCH_SIZE
    total_loss = total_loss/no_of_batches
    return total_accuracy / num_examples,total_loss

with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    num_examples = len(X_train)
    train_loss = 0
    train_losses = []
    train_accuracies = []
    validation_losses = []
    validation_accuracies = []
    print("[INFO] Training...")
    print()
    for i in range(EPOCHS):
```

```
        X_train, y_train = shuffle(X_train, y_train)
        for offset in range(0, num_examples, BATCH_SIZE):
            end = offset + BATCH_SIZE
            batch_x, batch_y = X_train[offset:end], y_train[offset:end]
            _,loss_val = sess.run([training_operation,loss_operation], feed_dict={x: batch_x, y: batch_y})
            train_loss+=loss_val
        no_of_batches = num_examples/BATCH_SIZE
        train_loss = train_loss/no_of_batches
        train_losses.append(train_loss)
        validation_accuracy,validation_loss_for_this_epoch = evaluate(X_validation, y_validation)
        validation_losses.append(validation_loss_for_this_epoch)
        validation_accuracies.append(validation_accuracy)
        print("EPOCH {} ...".format(i+1))
        print("Training loss is :"+str(train_loss))
        print("Validation loss = {:.3f}".format(validation_loss_for_this_epoch))
        print("Validation Accuracy = {:.3f}".format(validation_accuracy))
        print()

    saver.save(sess, './lenet')
    print("Model saved")

epoch_nums = []
for i in range(1,EPOCHS+1):
    epoch_nums.append(i)

plt.style.use("ggplot")
plt.figure()
plt.plot(epoch_nums, train_losses,label="train_loss")
plt.plot(epoch_nums, validation_losses,label="val_loss")
#plt.plot(epoch_nums, epoch_losses,label="train_acc")
#plt.plot(epoch_nums, validation_losses,label="val_acc")
plt.title("Training Loss and Validation Loss on CIFAR-100")
plt.xlabel("Epoch no.")
plt.ylabel("Loss")
plt.legend()
plt.savefig(args["plot"])


with tf.Session() as sess:
    saver.restore(sess, tf.train.latest_checkpoint('.'))
    test_accuracy,test_loss = evaluate(X_test, y_test)
    #print("test accuracy is : "+str(test_accuracy))
    print("Test Accuracy = {:.3f}".format(test_accuracy))
    predict = sess.run(prediction_values, feed_dict={x: X_test})
    #predict_5 = sess.run(prediction_values_5,feed_dict={x: X_test})
    predict_5 = sess.run(prediction_values_5,feed_dict={x:X_test})
    boolean_top_5 = tf.nn.in_top_k(predictions=predict_5,
targets=tf.convert_to_tensor(t_labels,dtype=tf.int32), k=5).eval()

f = open('cifar-100-python/meta', 'rb')
datadict = pickle.load(f)#, encoding='bytes')
```

```
f.close()

fine_labels = datadict['fine_label_names']
coarse_labels = datadict['coarse_label_names']
model = LeNet

print("Predict is :")
print(predict)
#print("Len of predict is :"+str(len(predict)))
#print("Len of test_label shape is:"+str(len(test_labels)))
#print("Type of test labels is :"+str(type(test_labels)))
#print("Type of predict is :"+str(type(predict)))
#print("Test labels")
#unique, counts = np.unique(test_labels, return_counts=True)
#print(np.asarray((unique, counts)).T)
#print("predict")
#unique, counts = np.unique(predict, return_counts=True)
#print(np.asarray((unique, counts)).T)

con_mat = tf.confusion_matrix(t_labels, predictions=predict) #, num_classes=100, dtype=tf.int32, name=None)

with tf.Session():
  #print('Confusion Matrix: \n\n',
  cm = tf.Tensor.eval(con_mat,feed_dict=None, session=None)
  print("\n######################################################")
  print("\nConfusion Matrix is :\n")
  print(cm)
  print("\n######################################################")

cm1 = np.zeros((10000,10000), dtype=int)
j=0
for i in boolean_top_5:
  if i == True:
    cm1[t_labels[j],t_labels[j]]+=1
    j+=1

print("\n######################################################")
print("\nClassification accuracy for each class are :\n")

for i in range(100):
  print(fine_labels[i]+" : "+str(cm[i,i])+" %")

print("\n######################################################")

coarse_label_names_to_count = {}
coarse_label_names_to_count['aquatic mammals'] = 0
coarse_label_names_to_count['fish'] = 0
coarse_label_names_to_count['flowers'] = 0
coarse_label_names_to_count['food containers'] = 0
coarse_label_names_to_count['fruit and vegetables'] = 0
```

```
coarse_label_names_to_count['household electrical devices'] = 0
coarse_label_names_to_count['household furniture'] = 0
coarse_label_names_to_count['insects'] = 0
coarse_label_names_to_count['large carnivores'] = 0
coarse_label_names_to_count['large man-made outdoor things'] = 0
coarse_label_names_to_count['large natural outdoor scenes'] = 0
coarse_label_names_to_count['large omnivores and herbivores'] = 0
coarse_label_names_to_count['medium-sized mammals'] = 0
coarse_label_names_to_count['non-insect invertebrates'] = 0
coarse_label_names_to_count['people'] = 0
coarse_label_names_to_count['small mammals'] = 0
coarse_label_names_to_count['trees'] = 0
coarse_label_names_to_count['reptiles'] = 0
coarse_label_names_to_count['vehicles 1'] = 0
coarse_label_names_to_count['vehicles 2'] = 0

no_of_classes_that_each_coarse_label_has = 5

other = 0
for i in range(100):
  if fine_labels[i] in ['beaver', 'dolphin', 'otter', 'seal','whale']:
     coarse_label_names_to_count['aquatic mammals']+=cm[i,i]
  elif fine_labels[i] in ['aquarium_fish', 'flatfish', 'ray', 'shark', 'trout']:
     coarse_label_names_to_count['fish']+=cm[i,i]
  elif fine_labels[i] in ['orchid', 'poppy', 'rose', 'sunflower', 'tulip']:
     coarse_label_names_to_count['flowers']+=cm[i,i]
  elif fine_labels[i] in ['bottle', 'bowl', 'can', 'cup', 'plate']:
     coarse_label_names_to_count['food containers']+=cm[i,i]
  elif fine_labels[i] in ['apple', 'mushroom','orange', 'pear', 'sweet_pepper']:
     coarse_label_names_to_count['fruit and vegetables']+=cm[i,i]
  elif fine_labels[i] in ['clock', 'keyboard', 'lamp', 'telephone', 'television']:
     coarse_label_names_to_count['household electrical devices']+=cm[i,i]
  elif fine_labels[i] in ['bed', 'chair', 'couch', 'table', 'wardrobe']:
     coarse_label_names_to_count['household furniture']+=cm[i,i]
  elif fine_labels[i] in ['bee', 'beetle', 'butterfly', 'caterpillar', 'cockroach']:
     coarse_label_names_to_count['insects']+=cm[i,i]
  elif fine_labels[i] in ['bear', 'leopard', 'lion', 'tiger', 'wolf']:
     coarse_label_names_to_count['large carnivores']+=cm[i,i]
  elif fine_labels[i] in ['bridge', 'castle', 'house', 'road', 'skyscraper']:
     coarse_label_names_to_count['large man-made outdoor things']+=cm[i,i]
  elif fine_labels[i] in ['cloud', 'forest', 'mountain', 'plain', 'sea']:
     coarse_label_names_to_count['large natural outdoor scenes']+=cm[i,i]
  elif fine_labels[i] in ['camel', 'cattle', 'chimpanzee', 'elephant', 'kangaroo']:
     coarse_label_names_to_count['large omnivores and herbivores']+=cm[i,i]
  elif fine_labels[i] in ['fox', 'porcupine', 'possum', 'raccoon', 'skunk']:
     coarse_label_names_to_count['medium-sized mammals']+=cm[i,i]
  elif fine_labels[i] in ['crab', 'lobster', 'snail', 'spider', 'worm']:
     coarse_label_names_to_count['non-insect invertebrates']+=cm[i,i]
  elif fine_labels[i] in ['baby', 'boy', 'girl', 'man', 'woman']:
     coarse_label_names_to_count['people']+=cm[i,i]
  elif fine_labels[i] in ['crocodile', 'dinosaur', 'lizard', 'snake', 'turtle']:
```

```
        coarse_label_names_to_count['reptiles']+=cm[i,i]
    elif fine_labels[i] in ['hamster', 'mouse', 'rabbit', 'shrew', 'squirrel']:
        coarse_label_names_to_count['small mammals']+=cm[i,i]
    elif fine_labels[i] in ['maple_tree', 'oak_tree', 'palm_tree', 'pine_tree', 'willow_tree']:
        coarse_label_names_to_count['trees']+=cm[i,i]
    elif fine_labels[i] in ['bicycle', 'bus', 'motorcycle', 'pickup_truck', 'train']:
        coarse_label_names_to_count['vehicles 1']+=cm[i,i]
    elif fine_labels[i] in ['lawn_mower', 'rocket', 'streetcar', 'tank', 'tractor']:
        coarse_label_names_to_count['vehicles 2']+=cm[i,i]
    else:
        other+=1

#print("\nNo of examples not classified into any of the super classes are :"+str(other))

#print("\n############################################")

print("\nClassification accuracy for each Super class are as follows :\n")
for i in coarse_label_names_to_count.keys():
    print(i + " : "+str(coarse_label_names_to_count[i]/5) + " %")

print("\n############################################\n")

print(coarse_label_names_to_count)

print("\n############################################")
print("\nRank 5 Classification accuracy for each class are :\n")

for i in range(100):
    print(fine_labels[i]+" : "+str(cm1[i,i])+" %")

coarse_label_names_to_count1 = {}
coarse_label_names_to_count1['aquatic mammals'] = 0
coarse_label_names_to_count1['fish'] = 0
coarse_label_names_to_count1['flowers'] = 0
coarse_label_names_to_count1['food containers'] = 0
coarse_label_names_to_count1['fruit and vegetables'] = 0
coarse_label_names_to_count1['household electrical devices'] = 0
coarse_label_names_to_count1['household furniture'] = 0
coarse_label_names_to_count1['insects'] = 0
coarse_label_names_to_count1['large carnivores'] = 0
coarse_label_names_to_count1['large man-made outdoor things'] = 0
coarse_label_names_to_count1['large natural outdoor scenes'] = 0
coarse_label_names_to_count1['large omnivores and herbivores'] = 0
coarse_label_names_to_count1['medium-sized mammals'] = 0
coarse_label_names_to_count1['non-insect invertebrates'] = 0
coarse_label_names_to_count1['people'] = 0
coarse_label_names_to_count1['small mammals'] = 0
coarse_label_names_to_count1['trees'] = 0
coarse_label_names_to_count1['reptiles'] = 0
coarse_label_names_to_count1['vehicles 1'] = 0
coarse_label_names_to_count1['vehicles 2'] = 0
```

```
other1 = 0
overall_top_5_accuracy = 0

for i in range(100):
  if fine_labels[i] in ['beaver', 'dolphin', 'otter', 'seal','whale']:
     coarse_label_names_to_count1['aquatic mammals']+=cm1[i,i]
     overall_top_5_accuracy+=cm1[i,i]
  elif fine_labels[i] in ['aquarium_fish', 'flatfish', 'ray', 'shark', 'trout']:
     coarse_label_names_to_count1['fish']+=cm1[i,i]
     overall_top_5_accuracy+=cm1[i,i]
  elif fine_labels[i] in ['orchid', 'poppy', 'rose', 'sunflower', 'tulip']:
     coarse_label_names_to_count1['flowers']+=cm1[i,i]
     overall_top_5_accuracy+=cm1[i,i]
  elif fine_labels[i] in ['bottle', 'bowl', 'can', 'cup', 'plate']:
     coarse_label_names_to_count1['food containers']+=cm1[i,i]
     overall_top_5_accuracy+=cm1[i,i]
  elif fine_labels[i] in ['apple', 'mushroom','orange', 'pear', 'sweet_pepper']:
     coarse_label_names_to_count1['fruit and vegetables']+=cm1[i,i]
     overall_top_5_accuracy+=cm1[i,i]
  elif fine_labels[i] in ['clock', 'keyboard', 'lamp', 'telephone', 'television']:
     coarse_label_names_to_count1['household electrical devices']+=cm1[i,i]
     overall_top_5_accuracy+=cm1[i,i]
  elif fine_labels[i] in ['bed', 'chair', 'couch', 'table', 'wardrobe']:
     coarse_label_names_to_count1['household furniture']+=cm1[i,i]
     overall_top_5_accuracy+=cm1[i,i]
  elif fine_labels[i] in ['bee', 'beetle', 'butterfly', 'caterpillar', 'cockroach']:
     coarse_label_names_to_count1['insects']+=cm1[i,i]
     overall_top_5_accuracy+=cm1[i,i]
  elif fine_labels[i] in ['bear', 'leopard', 'lion', 'tiger', 'wolf']:
     coarse_label_names_to_count1['large carnivores']+=cm1[i,i]
     overall_top_5_accuracy+=cm1[i,i]
  elif fine_labels[i] in ['bridge', 'castle', 'house', 'road', 'skyscraper']:
     coarse_label_names_to_count1['large man-made outdoor things']+=cm1[i,i]
     overall_top_5_accuracy+=cm1[i,i]
  elif fine_labels[i] in ['cloud', 'forest', 'mountain', 'plain', 'sea']:
     coarse_label_names_to_count1['large natural outdoor scenes']+=cm1[i,i]
     overall_top_5_accuracy+=cm1[i,i]
  elif fine_labels[i] in ['camel', 'cattle', 'chimpanzee', 'elephant', 'kangaroo']:
     coarse_label_names_to_count1['large omnivores and herbivores']+=cm1[i,i]
     overall_top_5_accuracy+=cm1[i,i]
  elif fine_labels[i] in ['fox', 'porcupine', 'possum', 'raccoon', 'skunk']:
     coarse_label_names_to_count1['medium-sized mammals']+=cm1[i,i]
     overall_top_5_accuracy+=cm1[i,i]
  elif fine_labels[i] in ['crab', 'lobster', 'snail', 'spider', 'worm']:
     coarse_label_names_to_count1['non-insect invertebrates']+=cm1[i,i]
     overall_top_5_accuracy+=cm1[i,i]
  elif fine_labels[i] in ['baby', 'boy', 'girl', 'man', 'woman']:
     coarse_label_names_to_count1['people']+=cm1[i,i]
     overall_top_5_accuracy+=cm1[i,i]
  elif fine_labels[i] in ['crocodile', 'dinosaur', 'lizard', 'snake', 'turtle']:
```

```
      coarse_label_names_to_count1['reptiles']+=cm1[i,i]
      overall_top_5_accuracy+=cm1[i,i]
   elif fine_labels[i] in ['hamster', 'mouse', 'rabbit', 'shrew', 'squirrel']:
      coarse_label_names_to_count1['small mammals']+=cm1[i,i]
      overall_top_5_accuracy+=cm1[i,i]
   elif fine_labels[i] in ['maple_tree', 'oak_tree', 'palm_tree', 'pine_tree', 'willow_tree']:
      coarse_label_names_to_count1['trees']+=cm1[i,i]
      overall_top_5_accuracy+=cm1[i,i]
   elif fine_labels[i] in ['bicycle', 'bus', 'motorcycle', 'pickup_truck', 'train']:
      coarse_label_names_to_count1['vehicles 1']+=cm1[i,i]
      overall_top_5_accuracy+=cm1[i,i]
   elif fine_labels[i] in ['lawn_mower', 'rocket', 'streetcar', 'tank', 'tractor']:
      coarse_label_names_to_count1['vehicles 2']+=cm1[i,i]
      overall_top_5_accuracy+=cm1[i,i]
   else:
      other1+=1

print("\nExamples not classified into any of the super classes are :"+str(other1))
print("\n####################################################")
print("\nRank 5 Classification accuracy for each Super class are as follows:\n")
for i in coarse_label_names_to_count.keys():
   print(i + " : "+str(coarse_label_names_to_count1[i]/5) + " %")
print("\n####################################################")

print("\n####################################################")
print("\nOverall rank 5 accuracy is :"+str(overall_top_5_accuracy/100))
print("\n####################################################")
```

Experiment 1 (No change in architecture, training for 5 epochs)

Results of the last epoch are:

EPOCH 5 ...
Training loss is :2.8615256806512384
Validation loss = 3.271
Validation Accuracy = 0.246

Test Accuracy = 0.248

Overall rank 5 accuracy is :50.12

##

Confusion Matrix is :

```
[[41  2  0 ...  0  0  0]
 [ 0 26  0 ...  0  0  0]
 [ 0  0  9 ...  3  0  1]
 ...
 [ 0  0  0 ... 21  0  0]
 [ 0  0  2 ...  3  4  0]
 [ 0  0  0 ...  0  1 31]]
```

##

##

Classification accuracy for each class are :

apple : 41 %
aquarium_fish : 26 %
baby : 9 %
bear : 8 %
beaver : 5 %
bed : 9 %
bee : 10 %
beetle : 29 %
bicycle : 46 %
bottle : 51 %
bowl : 12 %
boy : 12 %
bridge : 31 %
bus : 25 %
butterfly : 24 %

camel : 13 %
can : 31 %
castle : 24 %
caterpillar : 11 %
cattle : 11 %
chair : 74 %
chimpanzee : 32 %
clock : 28 %
cloud : 38 %
cockroach : 64 %
couch : 15 %
crab : 32 %
crocodile : 12 %
cup : 41 %
dinosaur : 19 %
dolphin : 19 %
elephant : 17 %
flatfish : 16 %
forest : 27 %
fox : 3 %
girl : 16 %
hamster : 17 %
house : 24 %
kangaroo : 9 %
keyboard : 52 %
lamp : 25 %
lawn_mower : 51 %
leopard : 27 %
lion : 10 %
lizard : 8 %
lobster : 12 %
man : 14 %
maple_tree : 30 %
motorcycle : 71 %
mountain : 37 %
mouse : 1 %
mushroom : 18 %
oak_tree : 46 %
orange : 22 %
orchid : 35 %
otter : 1 %
palm_tree : 52 %
pear : 21 %
pickup_truck : 26 %
pine_tree : 34 %
plain : 44 %
plate : 49 %
poppy : 9 %
porcupine : 15 %
possum : 7 %
rabbit : 5 %

raccoon : 30 %
ray : 5 %
road : 58 %
rocket : 53 %
rose : 9 %
sea : 54 %
seal : 0 %
shark : 19 %
shrew : 11 %
skunk : 76 %
skyscraper : 24 %
snail : 3 %
snake : 33 %
spider : 45 %
squirrel : 0 %
streetcar : 36 %
sunflower : 37 %
sweet_pepper : 8 %
table : 10 %
tank : 33 %
telephone : 29 %
television : 38 %
tiger : 13 %
tractor : 8 %
train : 14 %
trout : 33 %
tulip : 9 %
turtle : 3 %
wardrobe : 44 %
whale : 32 %
willow_tree : 8 %
wolf : 21 %
woman : 4 %
worm : 31 %

##

Classification accuracy for each Super class are as follows :

fruit and vegetables : 22.0 %
insects : 27.6 %
vehicles 1 : 36.4 %
fish : 19.8 %
large omnivores and herbivores : 16.4 %
food containers : 36.8 %
large carnivores : 15.8 %
non-insect invertebrates : 24.6 %
people : 11.0 %
trees : 34.0 %
large man-made outdoor things : 32.2 %
flowers : 19.8 %

household furniture : 30.4 %
household electrical devices : 34.4 %
small mammals : 6.8 %
large natural outdoor scenes : 40.0 %
medium-sized mammals : 26.2 %
vehicles 2 : 36.2 %
reptiles : 15.0 %
aquatic mammals : 11.4 %

##

{'fruit and vegetables': 110, 'insects': 138, 'vehicles 1': 182, 'fish': 99, 'large omnivores and herbivores': 82, 'food containers': 184, 'large carnivores': 79, 'non-insect invertebrates': 123, 'people': 55, 'trees': 170, 'large man-made outdoor things': 161, 'flowers': 99, 'household furniture': 152, 'household electrical devices': 172, 'small mammals': 34, 'large natural outdoor scenes': 200, 'medium-sized mammals': 131, 'vehicles 2': 181, 'reptiles': 75, 'aquatic mammals': 57}

##

Rank 5 Classification accuracy for each class are :

apple : 55 %
aquarium_fish : 49 %
baby : 49 %
bear : 55 %
beaver : 47 %
bed : 52 %
bee : 54 %
beetle : 55 %
bicycle : 47 %
bottle : 51 %
bowl : 58 %
boy : 59 %
bridge : 53 %
bus : 55 %
butterfly : 44 %
camel : 46 %
can : 50 %
castle : 52 %
caterpillar : 43 %
cattle : 51 %
chair : 58 %
chimpanzee : 57 %
clock : 56 %
cloud : 49 %
cockroach : 54 %
couch : 56 %
crab : 52 %
crocodile : 54 %
cup : 46 %
dinosaur : 47 %

dolphin : 48 %
elephant : 50 %
flatfish : 48 %
forest : 50 %
fox : 53 %
girl : 48 %
hamster : 48 %
house : 57 %
kangaroo : 56 %
keyboard : 45 %
lamp : 43 %
lawn_mower : 60 %
leopard : 44 %
lion : 53 %
lizard : 47 %
lobster : 41 %
man : 57 %
maple_tree : 52 %
motorcycle : 45 %
mountain : 60 %
mouse : 49 %
mushroom : 49 %
oak_tree : 48 %
orange : 55 %
orchid : 54 %
otter : 50 %
palm_tree : 52 %
pear : 47 %
pickup_truck : 45 %
pine_tree : 50 %
plain : 45 %
plate : 45 %
poppy : 45 %
porcupine : 53 %
possum : 56 %
rabbit : 54 %
raccoon : 46 %
ray : 48 %
road : 55 %
rocket : 50 %
rose : 43 %
sea : 46 %
seal : 50 %
shark : 48 %
shrew : 36 %
skunk : 46 %
skyscraper : 52 %
snail : 49 %
snake : 52 %
spider : 56 %
squirrel : 42 %

streetcar : 49 %
sunflower : 45 %
sweet_pepper : 52 %
table : 50 %
tank : 42 %
telephone : 44 %
television : 48 %
tiger : 53 %
tractor : 47 %
train : 57 %
trout : 44 %
tulip : 49 %
turtle : 52 %
wardrobe : 50 %
whale : 48 %
willow_tree : 53 %
wolf : 48 %
woman : 54 %
worm : 52 %

Examples not classified into any of the super classes are :0

##

Rank 5 Classification accuracy for each Super class are as follows:

fruit and vegetables : 51.6 %
insects : 50.0 %
vehicles 1 : 49.8 %
fish : 47.4 %
large omnivores and herbivores : 52.0 %
food containers : 50.0 %
large carnivores : 50.6 %
non-insect invertebrates : 50.0 %
people : 53.4 %
trees : 51.0 %
large man-made outdoor things : 53.8 %
flowers : 47.2 %
household furniture : 53.2 %
household electrical devices : 47.2 %
small mammals : 45.8 %
large natural outdoor scenes : 50.0 %
medium-sized mammals : 50.8 %
vehicles 2 : 49.6 %
reptiles : 50.4 %
aquatic mammals : 48.6 %

##

##

Overall rank 5 accuracy is :50.12

Experiment 2 (No change in architecture, training for 10 epochs)

Results of the last epoch are:

EPOCH 10 ...
Training loss is :2.686200213435093
Validation loss = 3.451
Validation Accuracy = 0.237

Test Accuracy = 0.239

###

Confusion Matrix is :

```
[[37  2  0 ...  0  0  0]
 [ 0 29  0 ...  0  2  0]
 [ 0  1  6 ...  1  3  1]
 ...
 [ 0  0  0 ... 10  0  0]
 [ 0  0  0 ...  0  6  3]
 [ 0  1  0 ...  0  1 39]]
```

###

###

Classification accuracy for each class are :

apple : 37 %
aquarium_fish : 29 %
baby : 6 %
bear : 10 %
beaver : 1 %
bed : 17 %
bee : 20 %
beetle : 44 %
bicycle : 56 %
bottle : 46 %
bowl : 5 %
boy : 10 %
bridge : 29 %
bus : 29 %
butterfly : 26 %
camel : 10 %

can : 29 %
castle : 21 %
caterpillar : 9 %
cattle : 11 %
chair : 66 %
chimpanzee : 22 %
clock : 49 %
cloud : 26 %
cockroach : 32 %
couch : 14 %
crab : 36 %
crocodile : 15 %
cup : 43 %
dinosaur : 17 %
dolphin : 28 %
elephant : 11 %
flatfish : 14 %
forest : 19 %
fox : 7 %
girl : 5 %
hamster : 7 %
house : 17 %
kangaroo : 5 %
keyboard : 58 %
lamp : 19 %
lawn_mower : 50 %
leopard : 26 %
lion : 9 %
lizard : 4 %
lobster : 18 %
man : 7 %
maple_tree : 21 %
motorcycle : 77 %
mountain : 42 %
mouse : 2 %
mushroom : 22 %
oak_tree : 44 %
orange : 13 %
orchid : 23 %
otter : 2 %
palm_tree : 39 %
pear : 24 %
pickup_truck : 34 %
pine_tree : 31 %
plain : 52 %
plate : 38 %
poppy : 1 %
porcupine : 4 %
possum : 10 %
rabbit : 5 %
raccoon : 29 %

ray : 7 %
road : 51 %
rocket : 53 %
rose : 10 %
sea : 42 %
seal : 1 %
shark : 14 %
shrew : 23 %
skunk : 79 %
skyscraper : 42 %
snail : 0 %
snake : 19 %
spider : 41 %
squirrel : 0 %
streetcar : 42 %
sunflower : 20 %
sweet_pepper : 11 %
table : 14 %
tank : 45 %
telephone : 34 %
television : 39 %
tiger : 13 %
tractor : 8 %
train : 13 %
trout : 44 %
tulip : 14 %
turtle : 2 %
wardrobe : 43 %
whale : 22 %
willow_tree : 12 %
wolf : 10 %
woman : 6 %
worm : 39 %

##

Classification accuracy for each Super class are as follows :

medium-sized mammals : 25.8 %
large natural outdoor scenes : 36.2 %
vehicles 1 : 41.8 %
flowers : 13.6 %
large man-made outdoor things : 32.0 %
small mammals : 7.4 %
vehicles 2 : 39.6 %
fish : 21.6 %
non-insect invertebrates : 26.8 %
household furniture : 30.8 %
fruit and vegetables : 21.4 %
household electrical devices : 39.8 %
large carnivores : 13.6 %

food containers : 32.2 %
reptiles : 11.4 %
aquatic mammals : 10.8 %
insects : 26.2 %
large omnivores and herbivores : 11.8 %
trees : 29.4 %
people : 6.8 %

##

{'medium-sized mammals': 129, 'large natural outdoor scenes': 181, 'vehicles 1': 209, 'flowers': 68, 'large man-made outdoor things': 160, 'small mammals': 37, 'vehicles 2': 198, 'fish': 108, 'non-insect invertebrates': 134, 'household furniture': 154, 'fruit and vegetables': 107, 'household electrical devices': 199, 'large carnivores': 68, 'food containers': 161, 'reptiles': 57, 'aquatic mammals': 54, 'insects': 131, 'large omnivores and herbivores': 59, 'trees': 147, 'people': 34}

##

Rank 5 Classification accuracy for each class are :

apple : 54 %
aquarium_fish : 49 %
baby : 49 %
bear : 55 %
beaver : 45 %
bed : 52 %
bee : 53 %
beetle : 55 %
bicycle : 46 %
bottle : 49 %
bowl : 58 %
boy : 58 %
bridge : 53 %
bus : 54 %
butterfly : 42 %
camel : 46 %
can : 49 %
castle : 52 %
caterpillar : 42 %
cattle : 51 %
chair : 58 %
chimpanzee : 57 %
clock : 55 %
cloud : 49 %
cockroach : 54 %
couch : 56 %
crab : 52 %
crocodile : 53 %
cup : 45 %
dinosaur : 47 %
dolphin : 48 %

elephant : 49 %
flatfish : 47 %
forest : 49 %
fox : 52 %
girl : 48 %
hamster : 48 %
house : 56 %
kangaroo : 56 %
keyboard : 44 %
lamp : 41 %
lawn_mower : 60 %
leopard : 42 %
lion : 53 %
lizard : 47 %
lobster : 40 %
man : 57 %
maple_tree : 51 %
motorcycle : 43 %
mountain : 58 %
mouse : 49 %
mushroom : 47 %
oak_tree : 48 %
orange : 54 %
orchid : 53 %
otter : 50 %
palm_tree : 51 %
pear : 47 %
pickup_truck : 43 %
pine_tree : 49 %
plain : 45 %
plate : 44 %
poppy : 42 %
porcupine : 52 %
possum : 56 %
rabbit : 54 %
raccoon : 46 %
ray : 48 %
road : 54 %
rocket : 50 %
rose : 42 %
sea : 46 %
seal : 50 %
shark : 48 %
shrew : 36 %
skunk : 46 %
skyscraper : 51 %
snail : 49 %
snake : 51 %
spider : 55 %
squirrel : 40 %
streetcar : 47 %

sunflower : 43 %
sweet_pepper : 52 %
table : 50 %
tank : 42 %
telephone : 44 %
television : 48 %
tiger : 53 %
tractor : 47 %
train : 56 %
trout : 43 %
tulip : 49 %
turtle : 51 %
wardrobe : 50 %
whale : 48 %
willow_tree : 53 %
wolf : 48 %
woman : 53 %
worm : 49 %

Examples not classified into any of the super classes are :0

##

Rank 5 Classification accuracy for each Super class are as follows:

medium-sized mammals : 50.4 %
large natural outdoor scenes : 49.4 %
vehicles 1 : 48.4 %
flowers : 45.8 %
large man-made outdoor things : 53.2 %
small mammals : 45.4 %
vehicles 2 : 49.2 %
fish : 47.0 %
non-insect invertebrates : 49.0 %
household furniture : 53.2 %
fruit and vegetables : 50.8 %
household electrical devices : 46.4 %
large carnivores : 50.2 %
food containers : 49.0 %
reptiles : 49.8 %
aquatic mammals : 48.2 %
insects : 49.2 %
large omnivores and herbivores : 51.8 %
trees : 50.4 %
people : 53.0 %

##

##

Overall rank 5 accuracy is :49.49

##

Experiment 3 (No change in architecture, training for 20 epochs)

Results of the last epoch are:

EPOCH 20 ...
Training loss is :2.6086030721651525
Validation loss = 3.299
Validation Accuracy = 0.262

Test Accuracy = 0.266

##

Confusion Matrix is :

```
[[50  2  1 ...  0  0  0]
 [ 0 31  0 ...  1  0  0]
 [ 1  1  9 ...  7  1  0]
 ...
 [ 0  0  0 ... 17  0  0]
 [ 0  0  0 ...  2  4  1]
 [ 0  1  1 ...  0  0 42]]
```

##

##

Classification accuracy for each class are :

apple : 50 %
aquarium_fish : 31 %
baby : 9 %
bear : 4 %
beaver : 6 %
bed : 15 %
bee : 10 %
beetle : 37 %
bicycle : 52 %
bottle : 48 %
bowl : 15 %
boy : 8 %
bridge : 16 %
bus : 31 %
butterfly : 39 %
camel : 19 %
can : 32 %
castle : 41 %

caterpillar : 12 %
cattle : 27 %
chair : 74 %
chimpanzee : 28 %
clock : 34 %
cloud : 41 %
cockroach : 41 %
couch : 10 %
crab : 17 %
crocodile : 7 %
cup : 61 %
dinosaur : 42 %
dolphin : 14 %
elephant : 16 %
flatfish : 21 %
forest : 29 %
fox : 12 %
girl : 8 %
hamster : 21 %
house : 36 %
kangaroo : 10 %
keyboard : 38 %
lamp : 29 %
lawn_mower : 52 %
leopard : 16 %
lion : 12 %
lizard : 7 %
lobster : 8 %
man : 7 %
maple_tree : 28 %
motorcycle : 70 %
mountain : 43 %
mouse : 8 %
mushroom : 14 %
oak_tree : 68 %
orange : 28 %
orchid : 32 %
otter : 3 %
palm_tree : 41 %
pear : 26 %
pickup_truck : 50 %
pine_tree : 10 %
plain : 59 %
plate : 41 %
poppy : 8 %
porcupine : 12 %
possum : 4 %
rabbit : 5 %
raccoon : 30 %
ray : 5 %
road : 50 %

rocket : 57 %
rose : 13 %
sea : 38 %
seal : 2 %
shark : 13 %
shrew : 14 %
skunk : 68 %
skyscraper : 42 %
snail : 1 %
snake : 23 %
spider : 61 %
squirrel : 4 %
streetcar : 36 %
sunflower : 33 %
sweet_pepper : 7 %
table : 10 %
tank : 43 %
telephone : 27 %
television : 51 %
tiger : 26 %
tractor : 35 %
train : 25 %
trout : 42 %
tulip : 14 %
turtle : 4 %
wardrobe : 47 %
whale : 20 %
willow_tree : 15 %
wolf : 17 %
woman : 4 %
worm : 42 %

##

Classification accuracy for each Super class are as follows :

food containers : 39.4 %
large natural outdoor scenes : 42.0 %
reptiles : 16.6 %
aquatic mammals : 9.0 %
household furniture : 31.2 %
non-insect invertebrates : 25.8 %
household electrical devices : 35.8 %
large man-made outdoor things : 37.0 %
fruit and vegetables : 25.0 %
small mammals : 10.4 %
large omnivores and herbivores : 20.0 %
insects : 27.8 %
large carnivores : 15.0 %
fish : 22.4 %
people : 7.2 %

medium-sized mammals : 25.2 %
vehicles 1 : 45.6 %
trees : 32.4 %
vehicles 2 : 44.6 %
flowers : 20.0 %

###

{'food containers': 197, 'large natural outdoor scenes': 210, 'reptiles': 83, 'aquatic mammals': 45, 'household furniture': 156, 'non-insect invertebrates': 129, 'household electrical devices': 179, 'large man-made outdoor things': 185, 'fruit and vegetables': 125, 'small mammals': 52, 'large omnivores and herbivores': 100, 'insects': 139, 'large carnivores': 75, 'fish': 112, 'people': 36, 'medium-sized mammals': 126, 'vehicles 1': 228, 'trees': 162, 'vehicles 2': 223, 'flowers': 100}

###

Rank 5 Classification accuracy for each class are :

apple : 56 %
aquarium_fish : 50 %
baby : 50 %
bear : 58 %
beaver : 48 %
bed : 57 %
bee : 56 %
beetle : 55 %
bicycle : 49 %
bottle : 53 %
bowl : 60 %
boy : 61 %
bridge : 57 %
bus : 61 %
butterfly : 48 %
camel : 53 %
can : 52 %
castle : 55 %
caterpillar : 45 %
cattle : 55 %
chair : 59 %
chimpanzee : 63 %
clock : 58 %
cloud : 51 %
cockroach : 56 %
couch : 60 %
crab : 52 %
crocodile : 56 %
cup : 50 %
dinosaur : 49 %
dolphin : 52 %
elephant : 50 %
flatfish : 50 %

forest : 52 %
fox : 54 %
girl : 53 %
hamster : 50 %
house : 59 %
kangaroo : 60 %
keyboard : 46 %
lamp : 44 %
lawn_mower : 63 %
leopard : 50 %
lion : 56 %
lizard : 51 %
lobster : 42 %
man : 58 %
maple_tree : 57 %
motorcycle : 50 %
mountain : 65 %
mouse : 50 %
mushroom : 50 %
oak_tree : 54 %
orange : 57 %
orchid : 56 %
otter : 54 %
palm_tree : 53 %
pear : 48 %
pickup_truck : 48 %
pine_tree : 52 %
plain : 49 %
plate : 49 %
poppy : 46 %
porcupine : 55 %
possum : 61 %
rabbit : 54 %
raccoon : 48 %
ray : 49 %
road : 59 %
rocket : 52 %
rose : 43 %
sea : 54 %
seal : 51 %
shark : 48 %
shrew : 42 %
skunk : 52 %
skyscraper : 56 %
snail : 49 %
snake : 56 %
spider : 56 %
squirrel : 45 %
streetcar : 52 %
sunflower : 48 %
sweet_pepper : 54 %

table : 51 %
tank : 43 %
telephone : 47 %
television : 51 %
tiger : 56 %
tractor : 51 %
train : 58 %
trout : 49 %
tulip : 49 %
turtle : 55 %
wardrobe : 52 %
whale : 48 %
willow_tree : 53 %
wolf : 50 %
woman : 58 %
worm : 55 %

Examples not classified into any of the super classes are :0

##

Rank 5 Classification accuracy for each Super class are as follows:

food containers : 52.8 %
large natural outdoor scenes : 54.2 %
reptiles : 53.4 %
aquatic mammals : 50.6 %
household furniture : 55.8 %
non-insect invertebrates : 50.8 %
household electrical devices : 49.2 %
large man-made outdoor things : 57.2 %
fruit and vegetables : 53.0 %
small mammals : 48.2 %
large omnivores and herbivores : 56.2 %
insects : 52.0 %
large carnivores : 54.0 %
fish : 49.2 %
people : 56.0 %
medium-sized mammals : 54.0 %
vehicles 1 : 53.2 %
trees : 53.8 %
vehicles 2 : 52.2 %
flowers : 48.4 %

##

##

Overall rank 5 accuracy is :52.71

##

Experiment 4 (No change in architecture and Increasing the batch size to 128)

Results of the last epoch are:

EPOCH 20 ...
Training loss is :2.5097842737686977
Validation loss = 3.332
Validation Accuracy = 0.264

Test Accuracy = 0.265

##

Confusion Matrix is :

```
[[51  0  0 ...  0  0  0]
 [ 0 23  0 ...  0  0  0]
 [ 0  1  5 ...  1  6  0]
 ...
 [ 0  0  0 ... 12  3  0]
 [ 0  0  1 ...  0  7  2]
 [ 0  0  0 ...  0  0 41]]
```

##

##

Classification accuracy for each class are :

apple : 51 %
aquarium_fish : 23 %
baby : 5 %
bear : 11 %
beaver : 4 %
bed : 19 %
bee : 12 %
beetle : 26 %
bicycle : 57 %
bottle : 27 %
bowl : 15 %
boy : 12 %
bridge : 39 %
bus : 26 %
butterfly : 22 %
camel : 9 %
can : 37 %
castle : 27 %

caterpillar : 12 %
cattle : 19 %
chair : 77 %
chimpanzee : 35 %
clock : 35 %
cloud : 24 %
cockroach : 67 %
couch : 17 %
crab : 26 %
crocodile : 8 %
cup : 54 %
dinosaur : 32 %
dolphin : 21 %
elephant : 30 %
flatfish : 16 %
forest : 45 %
fox : 8 %
girl : 1 %
hamster : 16 %
house : 17 %
kangaroo : 9 %
keyboard : 63 %
lamp : 28 %
lawn_mower : 51 %
leopard : 41 %
lion : 6 %
lizard : 2 %
lobster : 5 %
man : 12 %
maple_tree : 30 %
motorcycle : 81 %
mountain : 23 %
mouse : 10 %
mushroom : 19 %
oak_tree : 49 %
orange : 24 %
orchid : 26 %
otter : 1 %
palm_tree : 49 %
pear : 30 %
pickup_truck : 50 %
pine_tree : 35 %
plain : 46 %
plate : 39 %
poppy : 11 %
porcupine : 19 %
possum : 9 %
rabbit : 13 %
raccoon : 22 %
ray : 8 %
road : 56 %

rocket : 47 %
rose : 17 %
sea : 37 %
seal : 3 %
shark : 20 %
shrew : 9 %
skunk : 57 %
skyscraper : 57 %
snail : 1 %
snake : 22 %
spider : 46 %
squirrel : 0 %
streetcar : 41 %
sunflower : 28 %
sweet_pepper : 9 %
table : 19 %
tank : 41 %
telephone : 33 %
television : 44 %
tiger : 25 %
tractor : 16 %
train : 22 %
trout : 37 %
tulip : 12 %
turtle : 9 %
wardrobe : 49 %
whale : 32 %
willow_tree : 10 %
wolf : 12 %
woman : 7 %
worm : 41 %

##

Classification accuracy for each Super class are as follows :

large man-made outdoor things : 39.2 %
people : 7.4 %
non-insect invertebrates : 23.8 %
vehicles 2 : 39.2 %
large natural outdoor scenes : 35.0 %
flowers : 18.8 %
small mammals : 9.6 %
medium-sized mammals : 23.0 %
reptiles : 14.6 %
trees : 34.6 %
large omnivores and herbivores : 20.4 %
aquatic mammals : 12.2 %
household electrical devices : 40.6 %
food containers : 34.4 %
insects : 27.8 %

large carnivores : 19.0 %
fruit and vegetables : 26.6 %
fish : 20.8 %
household furniture : 36.2 %
vehicles 1 : 47.2 %

##

{'large man-made outdoor things': 196, 'people': 37, 'non-insect invertebrates': 119, 'vehicles 2': 196, 'large natural outdoor scenes': 175, 'flowers': 94, 'small mammals': 48, 'medium-sized mammals': 115, 'reptiles': 73, 'trees': 173, 'large omnivores and herbivores': 102, 'aquatic mammals': 61, 'household electrical devices': 203, 'food containers': 172, 'insects': 139, 'large carnivores': 95, 'fruit and vegetables': 133, 'fish': 104, 'household furniture': 181, 'vehicles 1': 236}

##

Rank 5 Classification accuracy for each class are :

apple : 57 %
aquarium_fish : 50 %
baby : 50 %
bear : 58 %
beaver : 48 %
bed : 57 %
bee : 58 %
beetle : 55 %
bicycle : 50 %
bottle : 53 %
bowl : 61 %
boy : 61 %
bridge : 57 %
bus : 61 %
butterfly : 49 %
camel : 55 %
can : 53 %
castle : 55 %
caterpillar : 46 %
cattle : 55 %
chair : 59 %
chimpanzee : 64 %
clock : 58 %
cloud : 51 %
cockroach : 56 %
couch : 60 %
crab : 53 %
crocodile : 56 %
cup : 50 %
dinosaur : 49 %
dolphin : 53 %
elephant : 50 %
flatfish : 51 %

forest : 53 %
fox : 54 %
girl : 53 %
hamster : 51 %
house : 59 %
kangaroo : 60 %
keyboard : 46 %
lamp : 44 %
lawn_mower : 63 %
leopard : 50 %
lion : 57 %
lizard : 51 %
lobster : 42 %
man : 58 %
maple_tree : 57 %
motorcycle : 51 %
mountain : 66 %
mouse : 50 %
mushroom : 50 %
oak_tree : 54 %
orange : 57 %
orchid : 57 %
otter : 54 %
palm_tree : 53 %
pear : 49 %
pickup_truck : 48 %
pine_tree : 52 %
plain : 50 %
plate : 49 %
poppy : 46 %
porcupine : 55 %
possum : 62 %
rabbit : 54 %
raccoon : 49 %
ray : 51 %
road : 59 %
rocket : 52 %
rose : 43 %
sea : 55 %
seal : 51 %
shark : 48 %
shrew : 42 %
skunk : 53 %
skyscraper : 56 %
snail : 49 %
snake : 58 %
spider : 57 %
squirrel : 45 %
streetcar : 53 %
sunflower : 48 %
sweet_pepper : 54 %

table : 51 %
tank : 44 %
telephone : 47 %
television : 51 %
tiger : 58 %
tractor : 52 %
train : 59 %
trout : 49 %
tulip : 51 %
turtle : 55 %
wardrobe : 52 %
whale : 49 %
willow_tree : 55 %
wolf : 51 %
woman : 58 %
worm : 55 %

Examples not classified into any of the super classes are :0

##

Rank 5 Classification accuracy for each Super class are as follows:

large man-made outdoor things : 57.2 %
people : 56.0 %
non-insect invertebrates : 51.2 %
vehicles 2 : 52.8 %
large natural outdoor scenes : 55.0 %
flowers : 49.0 %
small mammals : 48.4 %
medium-sized mammals : 54.6 %
reptiles : 53.8 %
trees : 54.2 %
large omnivores and herbivores : 56.8 %
aquatic mammals : 51.0 %
household electrical devices : 49.2 %
food containers : 53.2 %
insects : 52.8 %
large carnivores : 54.8 %
fruit and vegetables : 53.4 %
fish : 49.8 %
household furniture : 55.8 %
vehicles 1 : 53.8 %

##

##

Overall rank 5 accuracy is :53.14

##

Experiment 5 (Increasing the size of last but one fully connected layer to 100 from 84)

Results of the last epoch are:

EPOCH 25 ...
Training loss is :2.445217664142784
Validation loss = 3.796
Validation Accuracy = 0.230

Test Accuracy = 0.231

##

Confusion Matrix is :

```
[[39  1  0 ...  0  0  1]
 [ 0 25  0 ...  1  0  0]
 [ 1  1  6 ...  3  2  0]
 ...
 [ 0  2  0 ... 20  0  0]
 [ 0  0  3 ...  2  1  0]
 [ 0  1  0 ...  0  0 31]]
```

##

##

Classification accuracy for each class are :

apple : 39 %
aquarium_fish : 25 %
baby : 6 %
bear : 6 %
beaver : 3 %
bed : 16 %
bee : 13 %
beetle : 22 %
bicycle : 61 %
bottle : 45 %
bowl : 15 %
boy : 3 %
bridge : 32 %
bus : 30 %
butterfly : 45 %
camel : 11 %
can : 42 %
castle : 32 %

caterpillar : 23 %
cattle : 31 %
chair : 69 %
chimpanzee : 9 %
clock : 35 %
cloud : 30 %
cockroach : 46 %
couch : 12 %
crab : 31 %
crocodile : 12 %
cup : 54 %
dinosaur : 18 %
dolphin : 6 %
elephant : 12 %
flatfish : 10 %
forest : 37 %
fox : 6 %
girl : 4 %
hamster : 9 %
house : 22 %
kangaroo : 6 %
keyboard : 57 %
lamp : 28 %
lawn_mower : 45 %
leopard : 39 %
lion : 8 %
lizard : 4 %
lobster : 10 %
man : 6 %
maple_tree : 29 %
motorcycle : 66 %
mountain : 31 %
mouse : 1 %
mushroom : 12 %
oak_tree : 33 %
orange : 20 %
orchid : 14 %
otter : 2 %
palm_tree : 23 %
pear : 20 %
pickup_truck : 30 %
pine_tree : 22 %
plain : 45 %
plate : 37 %
poppy : 1 %
porcupine : 13 %
possum : 3 %
rabbit : 10 %
raccoon : 25 %
ray : 3 %
road : 38 %

rocket : 47 %
rose : 14 %
sea : 46 %
seal : 0 %
shark : 15 %
shrew : 6 %
skunk : 52 %
skyscraper : 39 %
snail : 1 %
snake : 15 %
spider : 27 %
squirrel : 1 %
streetcar : 52 %
sunflower : 17 %
sweet_pepper : 7 %
table : 16 %
tank : 27 %
telephone : 39 %
television : 44 %
tiger : 20 %
tractor : 16 %
train : 22 %
trout : 42 %
tulip : 4 %
turtle : 2 %
wardrobe : 46 %
whale : 26 %
willow_tree : 13 %
wolf : 20 %
woman : 1 %
worm : 31 %

##

Classification accuracy for each Super class are as follows :

vehicles 1 : 41.8 %
fruit and vegetables : 19.6 %
insects : 29.8 %
medium-sized mammals : 19.8 %
aquatic mammals : 7.4 %
trees : 24.0 %
large natural outdoor scenes : 37.8 %
non-insect invertebrates : 20.0 %
vehicles 2 : 37.4 %
food containers : 38.6 %
fish : 19.0 %
large man-made outdoor things : 32.6 %
large carnivores : 18.6 %
reptiles : 10.2 %
household furniture : 31.8 %

small mammals : 5.4 %
flowers : 10.0 %
large omnivores and herbivores : 13.8 %
household electrical devices : 40.6 %
people : 4.0 %

##

{'vehicles 1': 209, 'fruit and vegetables': 98, 'insects': 149, 'medium-sized mammals': 99, 'aquatic mammals': 37, 'trees': 120, 'large natural outdoor scenes': 189, 'non-insect invertebrates': 100, 'vehicles 2': 187, 'food containers': 193, 'fish': 95, 'large man-made outdoor things': 163, 'large carnivores': 93, 'reptiles': 51, 'household furniture': 159, 'small mammals': 27, 'flowers': 50, 'large omnivores and herbivores': 69, 'household electrical devices': 203, 'people': 20}

##

Rank 5 Classification accuracy for each class are :

apple : 53 %
aquarium_fish : 49 %
baby : 48 %
bear : 54 %
beaver : 44 %
bed : 51 %
bee : 52 %
beetle : 53 %
bicycle : 44 %
bottle : 49 %
bowl : 57 %
boy : 57 %
bridge : 52 %
bus : 54 %
butterfly : 41 %
camel : 45 %
can : 47 %
castle : 49 %
caterpillar : 42 %
cattle : 50 %
chair : 57 %
chimpanzee : 57 %
clock : 54 %
cloud : 47 %
cockroach : 54 %
couch : 53 %
crab : 51 %
crocodile : 52 %
cup : 43 %
dinosaur : 46 %
dolphin : 47 %
elephant : 47 %
flatfish : 47 %

forest : 48 %
fox : 50 %
girl : 47 %
hamster : 48 %
house : 56 %
kangaroo : 56 %
keyboard : 44 %
lamp : 39 %
lawn_mower : 59 %
leopard : 41 %
lion : 53 %
lizard : 46 %
lobster : 40 %
man : 56 %
maple_tree : 51 %
motorcycle : 41 %
mountain : 56 %
mouse : 49 %
mushroom : 47 %
oak_tree : 47 %
orange : 53 %
orchid : 52 %
otter : 47 %
palm_tree : 50 %
pear : 46 %
pickup_truck : 41 %
pine_tree : 47 %
plain : 45 %
plate : 44 %
poppy : 42 %
porcupine : 52 %
possum : 55 %
rabbit : 54 %
raccoon : 44 %
ray : 48 %
road : 52 %
rocket : 50 %
rose : 42 %
sea : 46 %
seal : 49 %
shark : 48 %
shrew : 36 %
skunk : 44 %
skyscraper : 51 %
snail : 48 %
snake : 50 %
spider : 54 %
squirrel : 40 %
streetcar : 47 %
sunflower : 43 %
sweet_pepper : 52 %

table : 50 %
tank : 39 %
telephone : 42 %
television : 46 %
tiger : 52 %
tractor : 47 %
train : 56 %
trout : 40 %
tulip : 49 %
turtle : 50 %
wardrobe : 47 %
whale : 47 %
willow_tree : 53 %
wolf : 48 %
woman : 52 %
worm : 48 %

Examples not classified into any of the super classes are :0

###

Rank 5 Classification accuracy for each Super class are as follows:

vehicles 1 : 47.2 %
fruit and vegetables : 50.2 %
insects : 48.4 %
medium-sized mammals : 49.0 %
aquatic mammals : 46.8 %
trees : 49.6 %
large natural outdoor scenes : 48.4 %
non-insect invertebrates : 48.2 %
vehicles 2 : 48.4 %
food containers : 48.0 %
fish : 46.4 %
large man-made outdoor things : 52.0 %
large carnivores : 49.6 %
reptiles : 48.8 %
household furniture : 51.6 %
small mammals : 45.4 %
flowers : 45.6 %
large omnivores and herbivores : 51.0 %
household electrical devices : 45.0 %
people : 52.0 %

###

###

Overall rank 5 accuracy is :48.58

###

Experiment 6 (Increasing the size of fc layer (last but one) to 200)

Results of the last epoch are:

EPOCH 25 ...
Training loss is :2.2481757302985184
Validation loss = 3.644
Validation Accuracy = 0.247

##

Confusion Matrix is :

```
[[36  1  0 ...  0  0  0]
 [ 0 25  1 ...  2  1  1]
 [ 0  1  5 ...  0  4  0]
 ...
 [ 0  0  0 ...  6  1  0]
 [ 0  0  0 ...  0  8  2]
 [ 0  0  0 ...  0  0 35]]
```

##

##

Classification accuracy for each class are :

apple : 36 %
aquarium_fish : 25 %
baby : 5 %
bear : 2 %
beaver : 6 %
bed : 21 %
bee : 7 %
beetle : 36 %
bicycle : 40 %
bottle : 48 %
bowl : 13 %
boy : 14 %
bridge : 39 %
bus : 26 %
butterfly : 23 %
camel : 15 %
can : 40 %
castle : 26 %
caterpillar : 9 %
cattle : 23 %

chair : 58 %
chimpanzee : 24 %
clock : 47 %
cloud : 34 %
cockroach : 40 %
couch : 14 %
crab : 33 %
crocodile : 12 %
cup : 61 %
dinosaur : 25 %
dolphin : 24 %
elephant : 29 %
flatfish : 20 %
forest : 21 %
fox : 10 %
girl : 6 %
hamster : 10 %
house : 26 %
kangaroo : 13 %
keyboard : 55 %
lamp : 31 %
lawn_mower : 59 %
leopard : 41 %
lion : 11 %
lizard : 6 %
lobster : 11 %
man : 6 %
maple_tree : 36 %
motorcycle : 69 %
mountain : 27 %
mouse : 3 %
mushroom : 25 %
oak_tree : 46 %
orange : 21 %
orchid : 20 %
otter : 4 %
palm_tree : 36 %
pear : 21 %
pickup_truck : 42 %
pine_tree : 30 %
plain : 54 %
plate : 41 %
poppy : 7 %
porcupine : 27 %
possum : 7 %
rabbit : 7 %
raccoon : 22 %
ray : 4 %
road : 44 %
rocket : 52 %
rose : 14 %

sea : 46 %
seal : 5 %
shark : 15 %
shrew : 7 %
skunk : 67 %
skyscraper : 37 %
snail : 4 %
snake : 24 %
spider : 32 %
squirrel : 2 %
streetcar : 45 %
sunflower : 36 %
sweet_pepper : 10 %
table : 25 %
tank : 30 %
telephone : 35 %
television : 43 %
tiger : 19 %
tractor : 19 %
train : 19 %
trout : 41 %
tulip : 9 %
turtle : 9 %
wardrobe : 42 %
whale : 36 %
willow_tree : 4 %
wolf : 6 %
woman : 8 %
worm : 35 %

##

Classification accuracy for each Super class are as follows :

vehicles 1 : 39.2 %
insects : 23.0 %
medium-sized mammals : 26.6 %
large man-made outdoor things : 34.4 %
food containers : 40.6 %
large carnivores : 15.8 %
aquatic mammals : 15.0 %
vehicles 2 : 41.0 %
fish : 21.0 %
non-insect invertebrates : 23.0 %
people : 7.8 %
flowers : 17.2 %
large omnivores and herbivores : 20.8 %
household electrical devices : 42.2 %
household furniture : 32.0 %
small mammals : 5.8 %
large natural outdoor scenes : 36.4 %

trees : 30.4 %
reptiles : 15.2 %
fruit and vegetables : 22.6 %

##

{'vehicles 1': 196, 'insects': 115, 'medium-sized mammals': 133, 'large man-made outdoor things': 172, 'food containers': 203, 'large carnivores': 79, 'aquatic mammals': 75, 'vehicles 2': 205, 'fish': 105, 'non-insect invertebrates': 115, 'people': 39, 'flowers': 86, 'large omnivores and herbivores': 104, 'household electrical devices': 211, 'household furniture': 160, 'small mammals': 29, 'large natural outdoor scenes': 182, 'trees': 152, 'reptiles': 76, 'fruit and vegetables': 113}

##

Rank 5 Classification accuracy for each class are :

apple : 55 %
aquarium_fish : 50 %
baby : 49 %
bear : 55 %
beaver : 48 %
bed : 54 %
bee : 55 %
beetle : 55 %
bicycle : 48 %
bottle : 53 %
bowl : 59 %
boy : 60 %
bridge : 54 %
bus : 58 %
butterfly : 45 %
camel : 49 %
can : 50 %
castle : 54 %
caterpillar : 44 %
cattle : 52 %
chair : 59 %
chimpanzee : 60 %
clock : 56 %
cloud : 50 %
cockroach : 54 %
couch : 56 %
crab : 52 %
crocodile : 54 %
cup : 47 %
dinosaur : 48 %
dolphin : 48 %
elephant : 50 %
flatfish : 50 %
forest : 52 %
fox : 54 %

girl : 52 %
hamster : 49 %
house : 58 %
kangaroo : 58 %
keyboard : 45 %
lamp : 43 %
lawn_mower : 62 %
leopard : 46 %
lion : 55 %
lizard : 49 %
lobster : 41 %
man : 57 %
maple_tree : 53 %
motorcycle : 47 %
mountain : 63 %
mouse : 49 %
mushroom : 49 %
oak_tree : 50 %
orange : 57 %
orchid : 55 %
otter : 52 %
palm_tree : 52 %
pear : 47 %
pickup_truck : 46 %
pine_tree : 50 %
plain : 45 %
plate : 46 %
poppy : 46 %
porcupine : 55 %
possum : 58 %
rabbit : 54 %
raccoon : 47 %
ray : 48 %
road : 58 %
rocket : 51 %
rose : 43 %
sea : 47 %
seal : 50 %
shark : 48 %
shrew : 38 %
skunk : 49 %
skyscraper : 55 %
snail : 49 %
snake : 56 %
spider : 56 %
squirrel : 44 %
streetcar : 50 %
sunflower : 45 %
sweet_pepper : 53 %
table : 51 %
tank : 42 %

telephone : 45 %
television : 49 %
tiger : 55 %
tractor : 48 %
train : 57 %
trout : 46 %
tulip : 49 %
turtle : 53 %
wardrobe : 51 %
whale : 48 %
willow_tree : 53 %
wolf : 50 %
woman : 56 %
worm : 53 %

Examples not classified into any of the super classes are :0

##

Rank 5 Classification accuracy for each Super class are as follows:

vehicles 1 : 51.2 %
insects : 50.6 %
medium-sized mammals : 52.6 %
large man-made outdoor things : 55.8 %
food containers : 51.0 %
large carnivores : 52.2 %
aquatic mammals : 49.2 %
vehicles 2 : 50.6 %
fish : 48.4 %
non-insect invertebrates : 50.2 %
people : 54.8 %
flowers : 47.6 %
large omnivores and herbivores : 53.8 %
household electrical devices : 47.6 %
household furniture : 54.2 %
small mammals : 46.8 %
large natural outdoor scenes : 51.4 %
trees : 51.6 %
reptiles : 52.0 %
fruit and vegetables : 52.2 %

##

##

Overall rank 5 accuracy is :51.19

##

Experiment 7 (No of filters in Conv layers increased to 10, 25 from 6 and 10 while keeping 200 in last but one fc layer)

Results of the last epoch are:

EPOCH 25 ...
Training loss is :2.091996416267163
Validation loss = 3.852
Validation Accuracy = 0.256

Test Accuracy = 0.268

##

Confusion Matrix is :

```
[[45  2  0 ...  0  0  0]
 [ 0 31  0 ...  2  0  1]
 [ 0  2  2 ...  0  2  1]
 ...
 [ 0  2  1 ... 16  0  0]
 [ 0  1  0 ...  0  4  0]
 [ 0  0  0 ...  0  0 40]]
```

##

##

Classification accuracy for each class are :

apple : 45 %
aquarium_fish : 31 %
baby : 2 %
bear : 11 %
beaver : 10 %
bed : 20 %
bee : 26 %
beetle : 43 %
bicycle : 44 %
bottle : 44 %
bowl : 16 %
boy : 22 %
bridge : 39 %
bus : 26 %
butterfly : 45 %
camel : 21 %
can : 41 %
castle : 34 %

caterpillar : 16 %
cattle : 33 %
chair : 70 %
chimpanzee : 20 %
clock : 41 %
cloud : 26 %
cockroach : 38 %
couch : 14 %
crab : 21 %
crocodile : 14 %
cup : 54 %
dinosaur : 31 %
dolphin : 9 %
elephant : 23 %
flatfish : 21 %
forest : 31 %
fox : 11 %
girl : 3 %
hamster : 14 %
house : 30 %
kangaroo : 5 %
keyboard : 45 %
lamp : 37 %
lawn_mower : 51 %
leopard : 24 %
lion : 9 %
lizard : 8 %
lobster : 6 %
man : 6 %
maple_tree : 37 %
motorcycle : 64 %
mountain : 33 %
mouse : 11 %
mushroom : 23 %
oak_tree : 42 %
orange : 36 %
orchid : 34 %
otter : 1 %
palm_tree : 36 %
pear : 25 %
pickup_truck : 42 %
pine_tree : 31 %
plain : 45 %
plate : 37 %
poppy : 8 %
porcupine : 31 %
possum : 8 %
rabbit : 7 %
raccoon : 13 %
ray : 6 %
road : 51 %

rocket : 50 %
rose : 12 %
sea : 45 %
seal : 0 %
shark : 19 %
shrew : 18 %
skunk : 66 %
skyscraper : 48 %
snail : 5 %
snake : 17 %
spider : 37 %
squirrel : 4 %
streetcar : 46 %
sunflower : 34 %
sweet_pepper : 7 %
table : 13 %
tank : 30 %
telephone : 45 %
television : 43 %
tiger : 37 %
tractor : 30 %
train : 21 %
trout : 43 %
tulip : 7 %
turtle : 2 %
wardrobe : 42 %
whale : 29 %
willow_tree : 17 %
wolf : 16 %
woman : 4 %
worm : 40 %

##

Classification accuracy for each Super class are as follows :

fish : 24.0 %
large carnivores : 19.4 %
large natural outdoor scenes : 36.0 %
food containers : 38.4 %
small mammals : 10.8 %
flowers : 19.0 %
household furniture : 31.8 %
vehicles 2 : 41.4 %
non-insect invertebrates : 21.8 %
people : 7.4 %
fruit and vegetables : 27.2 %
reptiles : 14.4 %
trees : 32.6 %
household electrical devices : 42.2 %
vehicles 1 : 39.4 %

large man-made outdoor things : 40.4 %
large omnivores and herbivores : 20.4 %
aquatic mammals : 9.8 %
insects : 33.6 %
medium-sized mammals : 25.8 %

##

{'fish': 120, 'large carnivores': 97, 'large natural outdoor scenes': 180, 'food containers': 192, 'small mammals': 54, 'flowers': 95, 'household furniture': 159, 'vehicles 2': 207, 'non-insect invertebrates': 109, 'people': 37, 'fruit and vegetables': 136, 'reptiles': 72, 'trees': 163, 'household electrical devices': 211, 'vehicles 1': 197, 'large man-made outdoor things': 202, 'large omnivores and herbivores': 102, 'aquatic mammals': 49, 'insects': 168, 'medium-sized mammals': 129}

##

Rank 5 Classification accuracy for each class are :

apple : 56 %
aquarium_fish : 50 %
baby : 50 %
bear : 58 %
beaver : 48 %
bed : 57 %
bee : 56 %
beetle : 55 %
bicycle : 49 %
bottle : 53 %
bowl : 60 %
boy : 61 %
bridge : 57 %
bus : 61 %
butterfly : 48 %
camel : 53 %
can : 52 %
castle : 55 %
caterpillar : 45 %
cattle : 55 %
chair : 59 %
chimpanzee : 63 %
clock : 58 %
cloud : 51 %
cockroach : 56 %
couch : 60 %
crab : 52 %
crocodile : 56 %
cup : 50 %
dinosaur : 49 %
dolphin : 52 %
elephant : 50 %
flatfish : 50 %

forest : 52 %
fox : 54 %
girl : 53 %
hamster : 51 %
house : 59 %
kangaroo : 60 %
keyboard : 46 %
lamp : 44 %
lawn_mower : 63 %
leopard : 50 %
lion : 56 %
lizard : 51 %
lobster : 42 %
man : 58 %
maple_tree : 57 %
motorcycle : 50 %
mountain : 65 %
mouse : 50 %
mushroom : 50 %
oak_tree : 54 %
orange : 57 %
orchid : 56 %
otter : 54 %
palm_tree : 53 %
pear : 48 %
pickup_truck : 48 %
pine_tree : 52 %
plain : 49 %
plate : 49 %
poppy : 46 %
porcupine : 55 %
possum : 61 %
rabbit : 54 %
raccoon : 48 %
ray : 49 %
road : 59 %
rocket : 52 %
rose : 43 %
sea : 54 %
seal : 51 %
shark : 48 %
shrew : 42 %
skunk : 52 %
skyscraper : 56 %
snail : 49 %
snake : 56 %
spider : 56 %
squirrel : 45 %
streetcar : 52 %
sunflower : 48 %
sweet_pepper : 54 %

table : 51 %
tank : 43 %
telephone : 47 %
television : 51 %
tiger : 56 %
tractor : 51 %
train : 59 %
trout : 49 %
tulip : 49 %
turtle : 55 %
wardrobe : 52 %
whale : 48 %
willow_tree : 53 %
wolf : 50 %
woman : 58 %
worm : 55 %

Examples not classified into any of the super classes are :0

##

Rank 5 Classification accuracy for each Super class are as follows:

fish : 49.2 %
large carnivores : 54.0 %
large natural outdoor scenes : 54.2 %
food containers : 52.8 %
small mammals : 48.4 %
flowers : 48.4 %
household furniture : 55.8 %
vehicles 2 : 52.2 %
non-insect invertebrates : 50.8 %
people : 56.0 %
fruit and vegetables : 53.0 %
reptiles : 53.4 %
trees : 53.8 %
household electrical devices : 49.2 %
vehicles 1 : 53.4 %
large man-made outdoor things : 57.2 %
large omnivores and herbivores : 56.2 %
aquatic mammals : 50.6 %
insects : 52.0 %
medium-sized mammals : 54.0 %

##

##

Overall rank 5 accuracy is :52.73

##

Experiment 8 (Increasing no of filters in conv layers to 25, 50 from 10,25 while keeping 200 in last but one fc layer)

Results of the last epoch are:

EPOCH 25 ...
Training loss is :1.7129645128477224
Validation loss = 3.975
Validation Accuracy = 0.277

Test Accuracy = 0.290
###

Confusion Matrix is :

```
[[32  3  0 ...  0  0  0]
 [ 0 30  0 ...  1  0  0]
 [ 0  0 10 ...  2  1  0]
 ...
 [ 0  1  1 ... 19  0  0]
 [ 0  1  1 ...  2  8  0]
 [ 0  0  0 ...  0  0 56]]
```

###

###

Classification accuracy for each class are :

apple : 32 %
aquarium_fish : 30 %
baby : 10 %
bear : 9 %
beaver : 7 %
bed : 24 %
bee : 20 %
beetle : 33 %
bicycle : 62 %
bottle : 51 %
bowl : 16 %
boy : 16 %
bridge : 37 %
bus : 36 %
butterfly : 37 %
camel : 24 %
can : 39 %
castle : 49 %
caterpillar : 21 %

cattle : 20 %
chair : 72 %
chimpanzee : 19 %
clock : 40 %
cloud : 34 %
cockroach : 42 %
couch : 27 %
crab : 32 %
crocodile : 15 %
cup : 50 %
dinosaur : 32 %
dolphin : 17 %
elephant : 19 %
flatfish : 30 %
forest : 43 %
fox : 11 %
girl : 10 %
hamster : 24 %
house : 28 %
kangaroo : 16 %
keyboard : 60 %
lamp : 38 %
lawn_mower : 55 %
leopard : 36 %
lion : 4 %
lizard : 11 %
lobster : 8 %
man : 15 %
maple_tree : 29 %
motorcycle : 76 %
mountain : 36 %
mouse : 7 %
mushroom : 17 %
oak_tree : 45 %
orange : 34 %
orchid : 32 %
otter : 3 %
palm_tree : 36 %
pear : 31 %
pickup_truck : 29 %
pine_tree : 38 %
plain : 71 %
plate : 37 %
poppy : 3 %
porcupine : 9 %
possum : 18 %
rabbit : 6 %
raccoon : 19 %
ray : 4 %
road : 56 %
rocket : 55 %

rose : 15 %
sea : 26 %
seal : 4 %
shark : 19 %
shrew : 17 %
skunk : 61 %
skyscraper : 53 %
snail : 10 %
snake : 28 %
spider : 38 %
squirrel : 1 %
streetcar : 43 %
sunflower : 40 %
sweet_pepper : 16 %
table : 18 %
tank : 42 %
telephone : 45 %
television : 43 %
tiger : 35 %
tractor : 31 %
train : 31 %
trout : 44 %
tulip : 9 %
turtle : 5 %
wardrobe : 38 %
whale : 35 %
willow_tree : 18 %
wolf : 19 %
woman : 8 %
worm : 56 %

###

Classification accuracy for each Super class are as follows :

household electrical devices : 45.2 %
reptiles : 18.2 %
food containers : 38.6 %
vehicles 1 : 46.8 %
household furniture : 35.8 %
vehicles 2 : 45.2 %
aquatic mammals : 13.2 %
trees : 33.2 %
large carnivores : 20.6 %
large omnivores and herbivores : 19.6 %
medium-sized mammals : 23.6 %
large natural outdoor scenes : 42.0 %
flowers : 19.8 %
fruit and vegetables : 26.0 %
non-insect invertebrates : 28.8 %
insects : 30.6 %

people : 11.8 %
large man-made outdoor things : 44.6 %
small mammals : 11.0 %
fish : 25.4 %

##

{'household electrical devices': 226, 'reptiles': 91, 'food containers': 193, 'vehicles 1': 234, 'household furniture': 179, 'vehicles 2': 226, 'aquatic mammals': 66, 'trees': 166, 'large carnivores': 103, 'large omnivores and herbivores': 98, 'medium-sized mammals': 118, 'large natural outdoor scenes': 210, 'flowers': 99, 'fruit and vegetables': 130, 'non-insect invertebrates': 144, 'insects': 153, 'people': 59, 'large man-made outdoor things': 223, 'small mammals': 55, 'fish': 127}

##

Rank 5 Classification accuracy for each class are :

apple : 59 %
aquarium_fish : 53 %
baby : 53 %
bear : 60 %
beaver : 50 %
bed : 60 %
bee : 61 %
beetle : 60 %
bicycle : 52 %
bottle : 55 %
bowl : 63 %
boy : 62 %
bridge : 59 %
bus : 62 %
butterfly : 52 %
camel : 60 %
can : 59 %
castle : 58 %
caterpillar : 48 %
cattle : 57 %
chair : 61 %
chimpanzee : 66 %
clock : 61 %
cloud : 53 %
cockroach : 57 %
couch : 64 %
crab : 57 %
crocodile : 61 %
cup : 56 %
dinosaur : 53 %
dolphin : 56 %
elephant : 53 %
flatfish : 53 %
forest : 55 %

fox : 54 %
girl : 55 %
hamster : 54 %
house : 62 %
kangaroo : 61 %
keyboard : 49 %
lamp : 48 %
lawn_mower : 64 %
leopard : 57 %
lion : 61 %
lizard : 59 %
lobster : 47 %
man : 59 %
maple_tree : 58 %
motorcycle : 58 %
mountain : 67 %
mouse : 52 %
mushroom : 54 %
oak_tree : 57 %
orange : 58 %
orchid : 59 %
otter : 57 %
palm_tree : 56 %
pear : 52 %
pickup_truck : 51 %
pine_tree : 56 %
plain : 56 %
plate : 54 %
poppy : 51 %
porcupine : 57 %
possum : 65 %
rabbit : 56 %
raccoon : 54 %
ray : 59 %
road : 61 %
rocket : 55 %
rose : 47 %
sea : 56 %
seal : 52 %
shark : 50 %
shrew : 45 %
skunk : 55 %
skyscraper : 61 %
snail : 50 %
snake : 65 %
spider : 58 %
squirrel : 49 %
streetcar : 54 %
sunflower : 49 %
sweet_pepper : 57 %
table : 53 %

tank : 49 %
telephone : 49 %
television : 55 %
tiger : 61 %
tractor : 57 %
train : 61 %
trout : 49 %
tulip : 51 %
turtle : 57 %
wardrobe : 57 %
whale : 57 %
willow_tree : 55 %
wolf : 53 %
woman : 62 %
worm : 56 %

Examples not classified into any of the super classes are :0

##

Rank 5 Classification accuracy for each Super class are as follows:

household electrical devices : 52.4 %
reptiles : 59.0 %
food containers : 57.4 %
vehicles 1 : 56.8 %
household furniture : 59.0 %
vehicles 2 : 55.8 %
aquatic mammals : 54.4 %
trees : 56.4 %
large carnivores : 58.4 %
large omnivores and herbivores : 59.4 %
medium-sized mammals : 57.0 %
large natural outdoor scenes : 57.4 %
flowers : 51.4 %
fruit and vegetables : 56.0 %
non-insect invertebrates : 53.6 %
insects : 55.6 %
people : 58.2 %
large man-made outdoor things : 60.2 %
small mammals : 51.2 %
fish : 52.8 %

##

##

Overall rank 5 accuracy is :56.12

##

Experiment 9 (Having no of filters in conv layers as 6,16 and no of node in last but one FC as 200)

Results of the last epoch are:

EPOCH 25 ...
Training loss is :2.1661688278390745
Validation loss = 3.721
Validation Accuracy = 0.252

Test Accuracy = 0.261

##

Confusion Matrix is :

```
[[41  2  1 ...  0  0  3]
 [ 0 23  0 ...  3  1  1]
 [ 0  0  6 ...  1  1  0]
 ...
 [ 0  1  0 ... 16  0  0]
 [ 0  1  2 ...  2  7  1]
 [ 0  0  0 ...  0  0 36]]
```

##

##

Classification accuracy for each class are :

apple : 41 %
aquarium_fish : 23 %
baby : 6 %
bear : 11 %
beaver : 4 %
bed : 25 %
bee : 10 %
beetle : 36 %
bicycle : 58 %
bottle : 41 %
bowl : 17 %
boy : 22 %
bridge : 36 %
bus : 29 %
butterfly : 21 %
camel : 10 %
can : 45 %
castle : 35 %
caterpillar : 11 %
cattle : 23 %

chair : 71 %
chimpanzee : 19 %
clock : 37 %
cloud : 41 %
cockroach : 41 %
couch : 18 %
crab : 27 %
crocodile : 14 %
cup : 50 %
dinosaur : 17 %
dolphin : 14 %
elephant : 18 %
flatfish : 22 %
forest : 42 %
fox : 7 %
girl : 7 %
hamster : 18 %
house : 17 %
kangaroo : 6 %
keyboard : 54 %
lamp : 35 %
lawn_mower : 52 %
leopard : 27 %
lion : 14 %
lizard : 9 %
lobster : 11 %
man : 13 %
maple_tree : 33 %
motorcycle : 72 %
mountain : 28 %
mouse : 8 %
mushroom : 26 %
oak_tree : 49 %
orange : 29 %
orchid : 24 %
otter : 8 %
palm_tree : 35 %
pear : 22 %
pickup_truck : 40 %
pine_tree : 42 %
plain : 59 %
plate : 40 %
poppy : 7 %
porcupine : 23 %
possum : 5 %
rabbit : 7 %
raccoon : 26 %
ray : 7 %
road : 42 %
rocket : 49 %
rose : 20 %

sea : 37 %
seal : 3 %
shark : 22 %
shrew : 9 %
skunk : 60 %
skyscraper : 37 %
snail : 3 %
snake : 18 %
spider : 38 %
squirrel : 2 %
streetcar : 47 %
sunflower : 26 %
sweet_pepper : 12 %
table : 19 %
tank : 24 %
telephone : 40 %
television : 34 %
tiger : 28 %
tractor : 20 %
train : 26 %
trout : 39 %
tulip : 5 %
turtle : 4 %
wardrobe : 49 %
whale : 30 %
willow_tree : 15 %
wolf : 16 %
woman : 7 %
worm : 36 %

##

Classification accuracy for each Super class are as follows :

medium-sized mammals : 24.2 %
vehicles 1 : 45.0 %
non-insect invertebrates : 23.0 %
vehicles 2 : 38.4 %
small mammals : 8.8 %
large natural outdoor scenes : 41.4 %
insects : 23.8 %
trees : 34.8 %
large carnivores : 19.2 %
household electrical devices : 40.0 %
reptiles : 12.4 %
fish : 22.6 %
large omnivores and herbivores : 15.2 %
aquatic mammals : 11.8 %
food containers : 38.6 %
household furniture : 36.4 %
fruit and vegetables : 26.0 %

large man-made outdoor things : 33.4 %
flowers : 16.4 %
people : 11.0 %

##

{'medium-sized mammals': 121, 'vehicles 1': 225, 'non-insect invertebrates': 115, 'vehicles 2': 192, 'small mammals': 44, 'large natural outdoor scenes': 207, 'insects': 119, 'trees': 174, 'large carnivores': 96, 'household electrical devices': 200, 'reptiles': 62, 'fish': 113, 'large omnivores and herbivores': 76, 'aquatic mammals': 59, 'food containers': 193, 'household furniture': 182, 'fruit and vegetables': 130, 'large man-made outdoor things': 167, 'flowers': 82, 'people': 55}

##

Rank 5 Classification accuracy for each class are :

apple : 55 %
aquarium_fish : 50 %
baby : 50 %
bear : 56 %
beaver : 48 %
bed : 55 %
bee : 56 %
beetle : 55 %
bicycle : 48 %
bottle : 53 %
bowl : 60 %
boy : 60 %
bridge : 54 %
bus : 58 %
butterfly : 46 %
camel : 50 %
can : 50 %
castle : 54 %
caterpillar : 44 %
cattle : 53 %
chair : 59 %
chimpanzee : 61 %
clock : 57 %
cloud : 51 %
cockroach : 54 %
couch : 57 %
crab : 52 %
crocodile : 54 %
cup : 48 %
dinosaur : 48 %
dolphin : 49 %
elephant : 50 %
flatfish : 50 %
forest : 52 %
fox : 54 %

girl : 52 %
hamster : 49 %
house : 59 %
kangaroo : 59 %
keyboard : 45 %
lamp : 43 %
lawn_mower : 62 %
leopard : 47 %
lion : 56 %
lizard : 49 %
lobster : 42 %
man : 57 %
maple_tree : 55 %
motorcycle : 47 %
mountain : 63 %
mouse : 49 %
mushroom : 49 %
oak_tree : 50 %
orange : 57 %
orchid : 55 %
otter : 52 %
palm_tree : 52 %
pear : 47 %
pickup_truck : 48 %
pine_tree : 51 %
plain : 45 %
plate : 48 %
poppy : 46 %
porcupine : 55 %
possum : 58 %
rabbit : 54 %
raccoon : 48 %
ray : 48 %
road : 59 %
rocket : 51 %
rose : 43 %
sea : 49 %
seal : 50 %
shark : 48 %
shrew : 40 %
skunk : 50 %
skyscraper : 55 %
snail : 49 %
snake : 56 %
spider : 56 %
squirrel : 45 %
streetcar : 51 %
sunflower : 46 %
sweet_pepper : 53 %
table : 51 %
tank : 42 %

telephone : 45 %
television : 50 %
tiger : 55 %
tractor : 49 %
train : 58 %
trout : 49 %
tulip : 49 %
turtle : 53 %
wardrobe : 51 %
whale : 48 %
willow_tree : 53 %
wolf : 50 %
woman : 56 %
worm : 53 %

Examples not classified into any of the super classes are :0

##

Rank 5 Classification accuracy for each Super class are as follows:

medium-sized mammals : 53.0 %
vehicles 1 : 51.8 %
non-insect invertebrates : 50.4 %
vehicles 2 : 51.0 %
small mammals : 47.4 %
large natural outdoor scenes : 52.0 %
insects : 51.0 %
trees : 52.2 %
large carnivores : 52.8 %
household electrical devices : 48.0 %
reptiles : 52.0 %
fish : 49.0 %
large omnivores and herbivores : 54.6 %
aquatic mammals : 49.4 %
food containers : 51.8 %
household furniture : 54.6 %
fruit and vegetables : 52.2 %
large man-made outdoor things : 56.2 %
flowers : 47.8 %
people : 55.0 %

##

##

Overall rank 5 accuracy is :51.61

##

Experiment 10 (Having the no of filters as 6,16 and no of nodes in last but one FC as 84)

Results of the last epoch are:

EPOCH 25 ...
Training loss is :2.5241929866243136
Validation loss = 3.649
Validation Accuracy = 0.224

Test Accuracy = 0.224

##

Confusion Matrix is :

```
[[36  0  0 ...  0  0  1]
 [ 0 14  2 ...  0  2  1]
 [ 0  0  9 ...  0  6  2]
 ...
 [ 0  0  0 ...  6  1  0]
 [ 0  0  0 ...  0 10  0]
 [ 0  0  0 ...  0  2 47]]
```

##

##

Classification accuracy for each class are :

apple : 36 %
aquarium_fish : 14 %
baby : 9 %
bear : 2 %
beaver : 2 %
bed : 20 %
bee : 7 %
beetle : 25 %
bicycle : 48 %
bottle : 43 %
bowl : 16 %
boy : 5 %
bridge : 36 %
bus : 15 %
butterfly : 40 %
camel : 7 %
can : 43 %
castle : 21 %
caterpillar : 4 %
cattle : 12 %
chair : 68 %

chimpanzee : 14 %
clock : 36 %
cloud : 14 %
cockroach : 53 %
couch : 14 %
crab : 15 %
crocodile : 9 %
cup : 61 %
dinosaur : 15 %
dolphin : 17 %
elephant : 10 %
flatfish : 13 %
forest : 26 %
fox : 3 %
girl : 10 %
hamster : 12 %
house : 15 %
kangaroo : 3 %
keyboard : 42 %
lamp : 36 %
lawn_mower : 53 %
leopard : 46 %
lion : 9 %
lizard : 4 %
lobster : 9 %
man : 6 %
maple_tree : 36 %
motorcycle : 59 %
mountain : 12 %
mouse : 6 %
mushroom : 14 %
oak_tree : 16 %
orange : 20 %
orchid : 22 %
otter : 0 %
palm_tree : 40 %
pear : 24 %
pickup_truck : 22 %
pine_tree : 42 %
plain : 41 %
plate : 30 %
poppy : 6 %
porcupine : 31 %
possum : 3 %
rabbit : 5 %
raccoon : 35 %
ray : 4 %
road : 47 %
rocket : 50 %
rose : 18 %
sea : 49 %

seal : 5 %
shark : 13 %
shrew : 7 %
skunk : 59 %
skyscraper : 48 %
snail : 4 %
snake : 13 %
spider : 37 %
squirrel : 1 %
streetcar : 33 %
sunflower : 30 %
sweet_pepper : 9 %
table : 12 %
tank : 12 %
telephone : 42 %
television : 44 %
tiger : 26 %
tractor : 7 %
train : 10 %
trout : 34 %
tulip : 7 %
turtle : 0 %
wardrobe : 40 %
whale : 31 %
willow_tree : 6 %
wolf : 6 %
woman : 10 %
worm : 47 %

##

Classification accuracy for each Super class are as follows :

fish : 15.6 %
large omnivores and herbivores : 9.2 %
vehicles 1 : 30.8 %
household furniture : 30.8 %
vehicles 2 : 31.0 %
medium-sized mammals : 26.2 %
fruit and vegetables : 20.6 %
household electrical devices : 40.0 %
flowers : 16.6 %
food containers : 38.6 %
trees : 28.0 %
large natural outdoor scenes : 28.4 %
insects : 25.8 %
large carnivores : 17.8 %
large man-made outdoor things : 33.4 %
aquatic mammals : 11.0 %
reptiles : 8.2 %
non-insect invertebrates : 22.4 %

people : 8.0 %
small mammals : 6.2 %

##

{'fish': 78, 'large omnivores and herbivores': 46, 'vehicles 1': 154, 'household furniture': 154, 'vehicles 2': 155, 'medium-sized mammals': 131, 'fruit and vegetables': 103, 'household electrical devices': 200, 'flowers': 83, 'food containers': 193, 'trees': 140, 'large natural outdoor scenes': 142, 'insects': 129, 'large carnivores': 89, 'large man-made outdoor things': 167, 'aquatic mammals': 55, 'reptiles': 41, 'non-insect invertebrates': 112, 'people': 40, 'small mammals': 31}

##

Rank 5 Classification accuracy for each class are :

apple : 53 %
aquarium_fish : 48 %
baby : 48 %
bear : 54 %
beaver : 42 %
bed : 50 %
bee : 51 %
beetle : 53 %
bicycle : 44 %
bottle : 47 %
bowl : 56 %
boy : 57 %
bridge : 52 %
bus : 53 %
butterfly : 40 %
camel : 45 %
can : 46 %
castle : 48 %
caterpillar : 42 %
cattle : 50 %
chair : 56 %
chimpanzee : 57 %
clock : 54 %
cloud : 47 %
cockroach : 53 %
couch : 53 %
crab : 51 %
crocodile : 52 %
cup : 43 %
dinosaur : 46 %
dolphin : 46 %
elephant : 46 %
flatfish : 47 %
forest : 46 %
fox : 50 %
girl : 45 %

hamster : 47 %
house : 54 %
kangaroo : 56 %
keyboard : 41 %
lamp : 39 %
lawn_mower : 59 %
leopard : 41 %
lion : 53 %
lizard : 46 %
lobster : 40 %
man : 52 %
maple_tree : 51 %
motorcycle : 41 %
mountain : 56 %
mouse : 49 %
mushroom : 47 %
oak_tree : 47 %
orange : 53 %
orchid : 50 %
otter : 47 %
palm_tree : 50 %
pear : 46 %
pickup_truck : 40 %
pine_tree : 47 %
plain : 45 %
plate : 42 %
poppy : 42 %
porcupine : 51 %
possum : 54 %
rabbit : 53 %
raccoon : 44 %
ray : 48 %
road : 52 %
rocket : 48 %
rose : 42 %
sea : 45 %
seal : 47 %
shark : 48 %
shrew : 35 %
skunk : 44 %
skyscraper : 50 %
snail : 46 %
snake : 50 %
spider : 53 %
squirrel : 40 %
streetcar : 47 %
sunflower : 42 %
sweet_pepper : 51 %
table : 49 %
tank : 39 %
telephone : 41 %

television : 45 %
tiger : 50 %
tractor : 47 %
train : 56 %
trout : 40 %
tulip : 49 %
turtle : 50 %
wardrobe : 47 %
whale : 47 %
willow_tree : 52 %
wolf : 48 %
woman : 51 %
worm : 48 %

Examples not classified into any of the super classes are :0

##

Rank 5 Classification accuracy for each Super class are as follows:

fish : 46.2 %
large omnivores and herbivores : 50.8 %
vehicles 1 : 46.8 %
household furniture : 51.0 %
vehicles 2 : 48.0 %
medium-sized mammals : 48.6 %
fruit and vegetables : 50.0 %
household electrical devices : 44.0 %
flowers : 45.0 %
food containers : 46.8 %
trees : 49.4 %
large natural outdoor scenes : 47.8 %
insects : 47.8 %
large carnivores : 49.2 %
large man-made outdoor things : 51.2 %
aquatic mammals : 45.8 %
reptiles : 48.8 %
non-insect invertebrates : 47.6 %
people : 50.6 %
small mammals : 44.8 %

##

##

Overall rank 5 accuracy is :48.01

##

Experiment 11 (Adding one more conv layer to original architecture and having the sizes of filters in them as 5, 3, 1 respectively. No of filters in layers are 6, 16, 25 respectively):

Results of the last epoch are:

EPOCH 25 ...
Training loss is :2.529280208085486
Validation loss = 3.290
Validation Accuracy = 0.266

Test Accuracy = 0.275

##

Confusion Matrix is :

```
[[41  1  0 ...  1  0  0]
 [ 0 29  1 ...  0  1  0]
 [ 0  1 12 ...  1  4  0]
 ...
 [ 0  0  1 ... 19  0  0]
 [ 0  0  2 ...  2  7  0]
 [ 0  0  0 ...  0  0 43]]
```

##

##

Classification accuracy for each class are :

apple : 41 %
aquarium_fish : 29 %
baby : 12 %
bear : 3 %
beaver : 3 %
bed : 13 %
bee : 25 %
beetle : 42 %
bicycle : 59 %
bottle : 45 %
bowl : 14 %
boy : 19 %
bridge : 34 %
bus : 26 %
butterfly : 31 %
camel : 11 %
can : 38 %
castle : 31 %
caterpillar : 12 %

cattle : 15 %
chair : 66 %
chimpanzee : 36 %
clock : 48 %
cloud : 32 %
cockroach : 54 %
couch : 11 %
crab : 23 %
crocodile : 11 %
cup : 59 %
dinosaur : 30 %
dolphin : 16 %
elephant : 25 %
flatfish : 18 %
forest : 37 %
fox : 8 %
girl : 10 %
hamster : 16 %
house : 17 %
kangaroo : 9 %
keyboard : 51 %
lamp : 33 %
lawn_mower : 53 %
leopard : 31 %
lion : 14 %
lizard : 2 %
lobster : 9 %
man : 8 %
maple_tree : 50 %
motorcycle : 74 %
mountain : 33 %
mouse : 6 %
mushroom : 13 %
oak_tree : 62 %
orange : 31 %
orchid : 29 %
otter : 5 %
palm_tree : 53 %
pear : 33 %
pickup_truck : 37 %
pine_tree : 30 %
plain : 60 %
plate : 30 %
poppy : 7 %
porcupine : 19 %
possum : 9 %
rabbit : 6 %
raccoon : 34 %
ray : 13 %
road : 56 %
rocket : 61 %

rose : 24 %
sea : 43 %
seal : 0 %
shark : 13 %
shrew : 11 %
skunk : 67 %
skyscraper : 44 %
snail : 3 %
snake : 21 %
spider : 36 %
squirrel : 0 %
streetcar : 57 %
sunflower : 33 %
sweet_pepper : 9 %
table : 17 %
tank : 36 %
telephone : 37 %
television : 41 %
tiger : 21 %
tractor : 11 %
train : 21 %
trout : 41 %
tulip : 17 %
turtle : 3 %
wardrobe : 56 %
whale : 19 %
willow_tree : 15 %
wolf : 19 %
woman : 7 %
worm : 43 %

###

Classification accuracy for each Super class are as follows :

aquatic mammals : 8.6 %
fruit and vegetables : 25.4 %
medium-sized mammals : 27.4 %
vehicles 2 : 43.6 %
household furniture : 32.6 %
insects : 32.8 %
reptiles : 13.4 %
vehicles 1 : 43.4 %
household electrical devices : 42.0 %
large carnivores : 17.6 %
food containers : 37.2 %
large man-made outdoor things : 36.4 %
fish : 22.8 %
non-insect invertebrates : 22.8 %
flowers : 22.0 %
small mammals : 7.8 %

people : 11.2 %
trees : 42.0 %
large natural outdoor scenes : 41.0 %
large omnivores and herbivores : 19.2 %

##

{'aquatic mammals': 43, 'fruit and vegetables': 127, 'medium-sized mammals': 137, 'vehicles 2': 218, 'household furniture': 163, 'insects': 164, 'reptiles': 67, 'vehicles 1': 217, 'household electrical devices': 210, 'large carnivores': 88, 'food containers': 186, 'large man-made outdoor things': 182, 'fish': 114, 'non-insect invertebrates': 114, 'flowers': 110, 'small mammals': 39, 'people': 56, 'trees': 210, 'large natural outdoor scenes': 205, 'large omnivores and herbivores': 96}

##

Rank 5 Classification accuracy for each class are :

apple : 58 %
aquarium_fish : 50 %
baby : 51 %
bear : 60 %
beaver : 49 %
bed : 58 %
bee : 59 %
beetle : 55 %
bicycle : 52 %
bottle : 54 %
bowl : 61 %
boy : 61 %
bridge : 57 %
bus : 61 %
butterfly : 49 %
camel : 55 %
can : 54 %
castle : 56 %
caterpillar : 46 %
cattle : 55 %
chair : 59 %
chimpanzee : 64 %
clock : 61 %
cloud : 51 %
cockroach : 56 %
couch : 62 %
crab : 56 %
crocodile : 59 %
cup : 53 %
dinosaur : 51 %
dolphin : 54 %
elephant : 51 %
flatfish : 52 %
forest : 54 %

fox : 54 %
girl : 55 %
hamster : 53 %
house : 61 %
kangaroo : 60 %
keyboard : 47 %
lamp : 45 %
lawn_mower : 63 %
leopard : 53 %
lion : 58 %
lizard : 55 %
lobster : 45 %
man : 58 %
maple_tree : 57 %
motorcycle : 53 %
mountain : 67 %
mouse : 52 %
mushroom : 53 %
oak_tree : 55 %
orange : 57 %
orchid : 57 %
otter : 55 %
palm_tree : 54 %
pear : 49 %
pickup_truck : 49 %
pine_tree : 53 %
plain : 52 %
plate : 54 %
poppy : 48 %
porcupine : 55 %
possum : 64 %
rabbit : 54 %
raccoon : 51 %
ray : 53 %
road : 59 %
rocket : 52 %
rose : 43 %
sea : 55 %
seal : 51 %
shark : 48 %
shrew : 43 %
skunk : 54 %
skyscraper : 58 %
snail : 50 %
snake : 60 %
spider : 57 %
squirrel : 46 %
streetcar : 53 %
sunflower : 48 %
sweet_pepper : 56 %
table : 51 %

tank : 45 %
telephone : 48 %
television : 52 %
tiger : 60 %
tractor : 54 %
train : 59 %
trout : 49 %
tulip : 51 %
turtle : 55 %
wardrobe : 54 %
whale : 53 %
willow_tree : 55 %
wolf : 52 %
woman : 60 %
worm : 56 %

Examples not classified into any of the super classes are :0

##

Rank 5 Classification accuracy for each Super class are as follows:

aquatic mammals : 52.4 %
fruit and vegetables : 54.6 %
medium-sized mammals : 55.6 %
vehicles 2 : 53.4 %
household furniture : 56.8 %
insects : 53.0 %
reptiles : 56.0 %
vehicles 1 : 54.8 %
household electrical devices : 50.6 %
large carnivores : 56.6 %
food containers : 55.2 %
large man-made outdoor things : 58.2 %
fish : 50.4 %
non-insect invertebrates : 52.8 %
flowers : 49.4 %
small mammals : 49.6 %
people : 57.0 %
trees : 54.8 %
large natural outdoor scenes : 55.8 %
large omnivores and herbivores : 57.0 %

##

##

Overall rank 5 accuracy is :54.2

##

Experiment 12 (Adding one more conv layer to original architecture and having the sizes of filters in them as 1,5,3 respectively. No of filters in layers are 6, 16, 25 respectively):

Results of the last epoch are:

EPOCH 25 ...
Training loss is :2.8379729450162214
Validation loss = 3.386
Validation Accuracy = 0.232

Test Accuracy = 0.237

###

Confusion Matrix is :

```
[[39  4  0 ...  0  1  0]
 [ 0 29  0 ...  1  1  1]
 [ 0  3  5 ...  5  4  0]
 ...
 [ 0  1  0 ... 27  1  2]
 [ 0  0  0 ...  1  8  3]
 [ 1  2  0 ...  1  0 33]]
```

###

###

Classification accuracy for each class are :

apple : 39 %
aquarium_fish : 29 %
baby : 5 %
bear : 5 %
beaver : 5 %
bed : 12 %
bee : 6 %
beetle : 32 %
bicycle : 50 %
bottle : 45 %
bowl : 10 %
boy : 7 %
bridge : 31 %
bus : 21 %
butterfly : 21 %
camel : 20 %
can : 29 %
castle : 23 %
caterpillar : 14 %
cattle : 17 %

chair : 74 %
chimpanzee : 31 %
clock : 27 %
cloud : 38 %
cockroach : 58 %
couch : 6 %
crab : 28 %
crocodile : 8 %
cup : 44 %
dinosaur : 33 %
dolphin : 12 %
elephant : 24 %
flatfish : 16 %
forest : 32 %
fox : 2 %
girl : 9 %
hamster : 16 %
house : 16 %
kangaroo : 10 %
keyboard : 51 %
lamp : 30 %
lawn_mower : 52 %
leopard : 14 %
lion : 10 %
lizard : 9 %
lobster : 8 %
man : 10 %
maple_tree : 30 %
motorcycle : 66 %
mountain : 21 %
mouse : 11 %
mushroom : 14 %
oak_tree : 64 %
orange : 26 %
orchid : 25 %
otter : 1 %
palm_tree : 33 %
pear : 24 %
pickup_truck : 32 %
pine_tree : 16 %
plain : 47 %
plate : 31 %
poppy : 7 %
porcupine : 22 %
possum : 16 %
rabbit : 7 %
raccoon : 21 %
ray : 4 %
road : 55 %
rocket : 48 %
rose : 13 %

sea : 31 %
seal : 2 %
shark : 16 %
shrew : 11 %
skunk : 65 %
skyscraper : 34 %
snail : 3 %
snake : 25 %
spider : 39 %
squirrel : 1 %
streetcar : 31 %
sunflower : 23 %
sweet_pepper : 8 %
table : 9 %
tank : 26 %
telephone : 27 %
television : 37 %
tiger : 10 %
tractor : 23 %
train : 24 %
trout : 43 %
tulip : 10 %
turtle : 2 %
wardrobe : 34 %
whale : 30 %
willow_tree : 17 %
wolf : 27 %
woman : 8 %
worm : 33 %

##

Classification accuracy for each Super class are as follows :

small mammals : 9.2 %
fruit and vegetables : 22.2 %
large carnivores : 13.2 %
vehicles 2 : 36.0 %
flowers : 15.6 %
insects : 26.2 %
reptiles : 15.4 %
household furniture : 27.0 %
large omnivores and herbivores : 20.4 %
medium-sized mammals : 25.2 %
fish : 21.6 %
aquatic mammals : 10.0 %
people : 7.8 %
trees : 32.0 %
non-insect invertebrates : 22.2 %
vehicles 1 : 38.6 %
large natural outdoor scenes : 33.8 %

household electrical devices : 34.4 %
large man-made outdoor things : 31.8 %
food containers : 31.8 %

##

{'small mammals': 46, 'fruit and vegetables': 111, 'large carnivores': 66, 'vehicles 2': 180, 'flowers': 78, 'insects': 131, 'reptiles': 77, 'household furniture': 135, 'large omnivores and herbivores': 102, 'medium-sized mammals': 126, 'fish': 108, 'aquatic mammals': 50, 'people': 39, 'trees': 160, 'non-insect invertebrates': 111, 'vehicles 1': 193, 'large natural outdoor scenes': 169, 'household electrical devices': 172, 'large man-made outdoor things': 159, 'food containers': 159}

##

Rank 5 Classification accuracy for each class are :

apple : 53 %
aquarium_fish : 49 %
baby : 48 %
bear : 55 %
beaver : 44 %
bed : 51 %
bee : 53 %
beetle : 54 %
bicycle : 46 %
bottle : 49 %
bowl : 58 %
boy : 57 %
bridge : 53 %
bus : 54 %
butterfly : 41 %
camel : 46 %
can : 48 %
castle : 51 %
caterpillar : 42 %
cattle : 51 %
chair : 57 %
chimpanzee : 57 %
clock : 55 %
cloud : 48 %
cockroach : 54 %
couch : 55 %
crab : 52 %
crocodile : 53 %
cup : 44 %
dinosaur : 47 %
dolphin : 47 %
elephant : 48 %
flatfish : 47 %
forest : 48 %
fox : 52 %

girl : 48 %
hamster : 48 %
house : 56 %
kangaroo : 56 %
keyboard : 44 %
lamp : 40 %
lawn_mower : 60 %
leopard : 41 %
lion : 53 %
lizard : 47 %
lobster : 40 %
man : 57 %
maple_tree : 51 %
motorcycle : 42 %
mountain : 57 %
mouse : 49 %
mushroom : 47 %
oak_tree : 47 %
orange : 54 %
orchid : 52 %
otter : 50 %
palm_tree : 50 %
pear : 46 %
pickup_truck : 43 %
pine_tree : 48 %
plain : 45 %
plate : 44 %
poppy : 42 %
porcupine : 52 %
possum : 56 %
rabbit : 54 %
raccoon : 45 %
ray : 48 %
road : 53 %
rocket : 50 %
rose : 42 %
sea : 46 %
seal : 49 %
shark : 48 %
shrew : 36 %
skunk : 45 %
skyscraper : 51 %
snail : 48 %
snake : 51 %
spider : 54 %
squirrel : 40 %
streetcar : 47 %
sunflower : 43 %
sweet_pepper : 52 %
table : 50 %
tank : 42 %

telephone : 43 %
television : 47 %
tiger : 53 %
tractor : 47 %
train : 56 %
trout : 42 %
tulip : 49 %
turtle : 50 %
wardrobe : 48 %
whale : 48 %
willow_tree : 53 %
wolf : 48 %
woman : 53 %
worm : 49 %

Examples not classified into any of the super classes are :0

##

Rank 5 Classification accuracy for each Super class are as follows:

small mammals : 45.4 %
fruit and vegetables : 50.4 %
large carnivores : 50.0 %
vehicles 2 : 49.2 %
flowers : 45.6 %
insects : 48.8 %
reptiles : 49.6 %
household furniture : 52.2 %
large omnivores and herbivores : 51.6 %
medium-sized mammals : 50.0 %
fish : 46.8 %
aquatic mammals : 47.6 %
people : 52.6 %
trees : 49.8 %
non-insect invertebrates : 48.6 %
vehicles 1 : 48.2 %
large natural outdoor scenes : 48.8 %
household electrical devices : 45.8 %
large man-made outdoor things : 52.8 %
food containers : 48.6 %

##

##

Overall rank 5 accuracy is :49.12

##

Experiment 13 (Without batch normalization i.e no change in architecture)

Results of the last epoch are:

EPOCH 25 ...
Training loss is :2.509759411135611
Validation loss = 3.559
Validation Accuracy = 0.247

Test Accuracy = 0.248

##

Confusion Matrix is :

```
[[48  1  0 ...  0  0  1]
 [ 0 32  1 ...  2  0  0]
 [ 2  2  5 ...  2  3  0]
 ...
 [ 0  1  0 ... 18  0  0]
 [ 0  0  1 ...  2  4  0]
 [ 0  1  1 ...  0  0 46]]
```

##

##

Classification accuracy for each class are :

apple : 48 %
aquarium_fish : 32 %
baby : 5 %
bear : 3 %
beaver : 2 %
bed : 20 %
bee : 21 %
beetle : 28 %
bicycle : 55 %
bottle : 54 %
bowl : 11 %
boy : 6 %
bridge : 28 %
bus : 40 %
butterfly : 23 %
camel : 13 %
can : 42 %
castle : 33 %
caterpillar : 12 %
cattle : 27 %
chair : 75 %
chimpanzee : 12 %

clock : 36 %
cloud : 19 %
cockroach : 49 %
couch : 14 %
crab : 19 %
crocodile : 3 %
cup : 51 %
dinosaur : 32 %
dolphin : 14 %
elephant : 21 %
flatfish : 16 %
forest : 30 %
fox : 5 %
girl : 4 %
hamster : 22 %
house : 22 %
kangaroo : 3 %
keyboard : 39 %
lamp : 28 %
lawn_mower : 55 %
leopard : 32 %
lion : 10 %
lizard : 4 %
lobster : 9 %
man : 6 %
maple_tree : 35 %
motorcycle : 74 %
mountain : 24 %
mouse : 3 %
mushroom : 20 %
oak_tree : 46 %
orange : 25 %
orchid : 18 %
otter : 1 %
palm_tree : 32 %
pear : 21 %
pickup_truck : 38 %
pine_tree : 27 %
plain : 56 %
plate : 46 %
poppy : 7 %
porcupine : 3 %
possum : 6 %
rabbit : 7 %
raccoon : 23 %
ray : 8 %
road : 51 %
rocket : 58 %
rose : 12 %
sea : 33 %
seal : 1 %

shark : 9 %
shrew : 7 %
skunk : 63 %
skyscraper : 36 %
snail : 1 %
snake : 17 %
spider : 37 %
squirrel : 1 %
streetcar : 54 %
sunflower : 27 %
sweet_pepper : 10 %
table : 25 %
tank : 25 %
telephone : 35 %
television : 46 %
tiger : 34 %
tractor : 23 %
train : 10 %
trout : 43 %
tulip : 12 %
turtle : 2 %
wardrobe : 44 %
whale : 29 %
willow_tree : 15 %
wolf : 18 %
woman : 4 %
worm : 46 %

###

Classification accuracy for each Super class are as follows :

aquatic mammals : 9.4 %
fruit and vegetables : 24.8 %
large omnivores and herbivores : 15.2 %
insects : 26.6 %
fish : 21.6 %
vehicles 2 : 43.0 %
flowers : 15.2 %
large natural outdoor scenes : 32.4 %
people : 5.0 %
vehicles 1 : 43.4 %
non-insect invertebrates : 22.4 %
small mammals : 8.0 %
large carnivores : 19.4 %
reptiles : 11.6 %
household furniture : 35.6 %
household electrical devices : 36.8 %
trees : 31.0 %
large man-made outdoor things : 34.0 %
food containers : 40.8 %

medium-sized mammals : 20.0 %

##

{'aquatic mammals': 47, 'fruit and vegetables': 124, 'large omnivores and herbivores': 76, 'insects': 133, 'fish': 108, 'vehicles 2': 215, 'flowers': 76, 'large natural outdoor scenes': 162, 'people': 25, 'vehicles 1': 217, 'non-insect invertebrates': 112, 'small mammals': 40, 'large carnivores': 97, 'reptiles': 58, 'household furniture': 178, 'household electrical devices': 184, 'trees': 155, 'large man-made outdoor things': 170, 'food containers': 204, 'medium-sized mammals': 100}

##

Rank 5 Classification accuracy for each class are :

apple : 54 %
aquarium_fish : 49 %
baby : 49 %
bear : 55 %
beaver : 46 %
bed : 52 %
bee : 54 %
beetle : 55 %
bicycle : 46 %
bottle : 50 %
bowl : 58 %
boy : 59 %
bridge : 53 %
bus : 54 %
butterfly : 42 %
camel : 46 %
can : 49 %
castle : 52 %
caterpillar : 42 %
cattle : 51 %
chair : 58 %
chimpanzee : 57 %
clock : 55 %
cloud : 49 %
cockroach : 54 %
couch : 56 %
crab : 52 %
crocodile : 53 %
cup : 45 %
dinosaur : 47 %
dolphin : 48 %
elephant : 49 %
flatfish : 48 %
forest : 49 %
fox : 52 %
girl : 48 %
hamster : 48 %

house : 56 %
kangaroo : 56 %
keyboard : 44 %
lamp : 42 %
lawn_mower : 60 %
leopard : 42 %
lion : 53 %
lizard : 47 %
lobster : 41 %
man : 57 %
maple_tree : 51 %
motorcycle : 44 %
mountain : 58 %
mouse : 49 %
mushroom : 47 %
oak_tree : 48 %
orange : 54 %
orchid : 53 %
otter : 50 %
palm_tree : 51 %
pear : 47 %
pickup_truck : 43 %
pine_tree : 50 %
plain : 45 %
plate : 44 %
poppy : 42 %
porcupine : 53 %
possum : 56 %
rabbit : 54 %
raccoon : 46 %
ray : 48 %
road : 54 %
rocket : 50 %
rose : 42 %
sea : 46 %
seal : 50 %
shark : 48 %
shrew : 36 %
skunk : 46 %
skyscraper : 51 %
snail : 49 %
snake : 51 %
spider : 55 %
squirrel : 40 %
streetcar : 49 %
sunflower : 43 %
sweet_pepper : 52 %
table : 50 %
tank : 42 %
telephone : 44 %
television : 48 %

tiger : 53 %
tractor : 47 %
train : 56 %
trout : 43 %
tulip : 49 %
turtle : 51 %
wardrobe : 50 %
whale : 48 %
willow_tree : 53 %
wolf : 48 %
woman : 54 %
worm : 50 %

Examples not classified into any of the super classes are :0

##

Rank 5 Classification accuracy for each Super class are as follows:

aquatic mammals : 48.4 %
fruit and vegetables : 50.8 %
large omnivores and herbivores : 51.8 %
insects : 49.4 %
fish : 47.2 %
vehicles 2 : 49.6 %
flowers : 45.8 %
large natural outdoor scenes : 49.4 %
people : 53.4 %
vehicles 1 : 48.6 %
non-insect invertebrates : 49.4 %
small mammals : 45.4 %
large carnivores : 50.2 %
reptiles : 49.8 %
household furniture : 53.2 %
household electrical devices : 46.6 %
trees : 50.6 %
large man-made outdoor things : 53.2 %
food containers : 49.2 %
medium-sized mammals : 50.6 %

##

##

Overall rank 5 accuracy is :49.63

##

Experiment 14 (With batch normalization added without changing any other in the architecture)

$ python assign5_lenet_modified.py --epochs 25 --batch_size 64 --plot plots/plot_lenet_with_batch_normalization

EPOCH 25 ...
Training loss is :2.364512693824676
Validation loss = 3.443
Validation Accuracy = 0.261

Test Accuracy = 0.261

##

Confusion Matrix is :

```
[[44  2  0 ...  0  0  0]
 [ 0 16  0 ...  0  2  0]
 [ 1  1  9 ...  2  8  0]
 ...
 [ 0  0  1 ... 13  0  0]
 [ 0  0  3 ...  1 10  3]
 [ 0  0  0 ...  0  0 45]]
```

##

##

Classification accuracy for each class are :

apple : 44 %
aquarium_fish : 16 %
baby : 9 %
bear : 2 %
beaver : 8 %
bed : 16 %
bee : 14 %
beetle : 34 %
bicycle : 44 %
bottle : 51 %
bowl : 13 %
boy : 19 %
bridge : 40 %
bus : 28 %
butterfly : 28 %
camel : 12 %
can : 30 %
castle : 32 %
caterpillar : 6 %
cattle : 22 %

chair : 65 %
chimpanzee : 19 %
clock : 38 %
cloud : 30 %
cockroach : 48 %
couch : 19 %
crab : 28 %
crocodile : 13 %
cup : 49 %
dinosaur : 36 %
dolphin : 18 %
elephant : 12 %
flatfish : 21 %
forest : 34 %
fox : 6 %
girl : 5 %
hamster : 24 %
house : 12 %
kangaroo : 15 %
keyboard : 54 %
lamp : 31 %
lawn_mower : 55 %
leopard : 27 %
lion : 14 %
lizard : 14 %
lobster : 13 %
man : 8 %
maple_tree : 36 %
motorcycle : 63 %
mountain : 19 %
mouse : 9 %
mushroom : 17 %
oak_tree : 58 %
orange : 22 %
orchid : 26 %
otter : 2 %
palm_tree : 42 %
pear : 33 %
pickup_truck : 31 %
pine_tree : 28 %
plain : 64 %
plate : 41 %
poppy : 6 %
porcupine : 13 %
possum : 6 %
rabbit : 8 %
raccoon : 25 %
ray : 6 %
road : 35 %
rocket : 51 %
rose : 21 %

sea : 44 %
seal : 4 %
shark : 13 %
shrew : 13 %
skunk : 66 %
skyscraper : 34 %
snail : 2 %
snake : 26 %
spider : 38 %
squirrel : 2 %
streetcar : 51 %
sunflower : 28 %
sweet_pepper : 11 %
table : 17 %
tank : 37 %
telephone : 52 %
television : 42 %
tiger : 39 %
tractor : 16 %
train : 24 %
trout : 54 %
tulip : 8 %
turtle : 5 %
wardrobe : 38 %
whale : 27 %
willow_tree : 11 %
wolf : 13 %
woman : 10 %
worm : 45 %

##

Classification accuracy for each Super class are as follows :

trees : 35.0 %
reptiles : 18.8 %
large man-made outdoor things : 30.6 %
non-insect invertebrates : 25.2 %
large carnivores : 19.0 %
people : 10.2 %
large omnivores and herbivores : 16.0 %
insects : 26.0 %
aquatic mammals : 11.8 %
fish : 22.0 %
small mammals : 11.2 %
flowers : 17.8 %
vehicles 1 : 38.0 %
household furniture : 31.0 %
vehicles 2 : 42.0 %
food containers : 36.8 %
large natural outdoor scenes : 38.2 %

fruit and vegetables : 25.4 %
medium-sized mammals : 23.2 %
household electrical devices : 43.4 %

##

{'trees': 175, 'reptiles': 94, 'large man-made outdoor things': 153, 'non-insect invertebrates': 126, 'large carnivores': 95, 'people': 51, 'large omnivores and herbivores': 80, 'insects': 130, 'aquatic mammals': 59, 'fish': 110, 'small mammals': 56, 'flowers': 89, 'vehicles 1': 190, 'household furniture': 155, 'vehicles 2': 210, 'food containers': 184, 'large natural outdoor scenes': 191, 'fruit and vegetables': 127, 'medium-sized mammals': 116, 'household electrical devices': 217}

##

Rank 5 Classification accuracy for each class are :

apple : 56 %
aquarium_fish : 50 %
baby : 50 %
bear : 58 %
beaver : 48 %
bed : 57 %
bee : 56 %
beetle : 55 %
bicycle : 49 %
bottle : 53 %
bowl : 60 %
boy : 61 %
bridge : 57 %
bus : 61 %
butterfly : 48 %
camel : 53 %
can : 51 %
castle : 55 %
caterpillar : 45 %
cattle : 55 %
chair : 59 %
chimpanzee : 63 %
clock : 58 %
cloud : 51 %
cockroach : 56 %
couch : 60 %
crab : 52 %
crocodile : 56 %
cup : 50 %
dinosaur : 49 %
dolphin : 52 %
elephant : 50 %
flatfish : 50 %
forest : 52 %
fox : 54 %

girl : 53 %
hamster : 50 %
house : 59 %
kangaroo : 60 %
keyboard : 46 %
lamp : 44 %
lawn_mower : 63 %
leopard : 50 %
lion : 56 %
lizard : 51 %
lobster : 42 %
man : 58 %
maple_tree : 55 %
motorcycle : 50 %
mountain : 65 %
mouse : 50 %
mushroom : 50 %
oak_tree : 53 %
orange : 57 %
orchid : 56 %
otter : 53 %
palm_tree : 53 %
pear : 48 %
pickup_truck : 48 %
pine_tree : 52 %
plain : 48 %
plate : 48 %
poppy : 46 %
porcupine : 55 %
possum : 61 %
rabbit : 54 %
raccoon : 48 %
ray : 48 %
road : 59 %
rocket : 52 %
rose : 43 %
sea : 53 %
seal : 51 %
shark : 48 %
shrew : 42 %
skunk : 52 %
skyscraper : 56 %
snail : 49 %
snake : 56 %
spider : 56 %
squirrel : 45 %
streetcar : 52 %
sunflower : 48 %
sweet_pepper : 54 %
table : 51 %
tank : 43 %

telephone : 47 %
television : 51 %
tiger : 56 %
tractor : 51 %
train : 58 %
trout : 49 %
tulip : 49 %
turtle : 55 %
wardrobe : 52 %
whale : 48 %
willow_tree : 53 %
wolf : 50 %
woman : 58 %
worm : 55 %

Examples not classified into any of the super classes are :0

###

Rank 5 Classification accuracy for each Super class are as follows:

trees : 53.2 %
reptiles : 53.4 %
large man-made outdoor things : 57.2 %
non-insect invertebrates : 50.8 %
large carnivores : 54.0 %
people : 56.0 %
large omnivores and herbivores : 56.2 %
insects : 52.0 %
aquatic mammals : 50.4 %
fish : 49.0 %
small mammals : 48.2 %
flowers : 48.4 %
vehicles 1 : 53.2 %
household furniture : 55.8 %
vehicles 2 : 52.2 %
food containers : 52.4 %
large natural outdoor scenes : 53.8 %
fruit and vegetables : 53.0 %
medium-sized mammals : 54.0 %
household electrical devices : 49.2 %

###

###

Overall rank 5 accuracy is :52.62

###

Experiment 15 (Combing best of all)

Results of the last epoch are:

EPOCH 25 ...
Training loss is :0.8084015914867912
Validation loss = 5.621
Validation Accuracy = 0.297

Test Accuracy = 30.9

##

Confusion Matrix is :

```
[[43  1  1 ...  0  0  1]
 [ 0 22  0 ...  1  1  2]
 [ 0  0 14 ...  0  4  0]
 ...
 [ 0  0  1 ... 18  0  0]
 [ 0  0  2 ...  0 13  1]
 [ 0  0  0 ...  0  0 50]]
```

##

##

Classification accuracy for each class are :

apple : 43 %
aquarium_fish : 22 %
baby : 14 %
bear : 10 %
beaver : 7 %
bed : 27 %
bee : 23 %
beetle : 29 %
bicycle : 62 %
bottle : 46 %
bowl : 24 %
boy : 16 %
bridge : 46 %
bus : 32 %
butterfly : 40 %
camel : 14 %
can : 59 %
castle : 38 %
caterpillar : 20 %
cattle : 27 %

chair : 70 %
chimpanzee : 37 %
clock : 50 %
cloud : 34 %
cockroach : 49 %
couch : 16 %
crab : 35 %
crocodile : 12 %
cup : 55 %
dinosaur : 36 %
dolphin : 21 %
elephant : 18 %
flatfish : 22 %
forest : 29 %
fox : 16 %
girl : 13 %
hamster : 12 %
house : 30 %
kangaroo : 13 %
keyboard : 66 %
lamp : 26 %
lawn_mower : 51 %
leopard : 49 %
lion : 8 %
lizard : 10 %
lobster : 16 %
man : 13 %
maple_tree : 31 %
motorcycle : 75 %
mountain : 31 %
mouse : 25 %
mushroom : 19 %
oak_tree : 55 %
orange : 47 %
orchid : 24 %
otter : 4 %
palm_tree : 44 %
pear : 31 %
pickup_truck : 54 %
pine_tree : 44 %
plain : 56 %
plate : 34 %
poppy : 12 %
porcupine : 33 %
possum : 13 %
rabbit : 10 %
raccoon : 35 %
ray : 7 %
road : 56 %
rocket : 59 %
rose : 26 %

sea : 36 %
seal : 9 %
shark : 16 %
shrew : 21 %
skunk : 67 %
skyscraper : 40 %
snail : 8 %
snake : 30 %
spider : 44 %
squirrel : 6 %
streetcar : 38 %
sunflower : 41 %
sweet_pepper : 16 %
table : 21 %
tank : 27 %
telephone : 44 %
television : 47 %
tiger : 37 %
tractor : 26 %
train : 35 %
trout : 49 %
tulip : 7 %
turtle : 13 %
wardrobe : 53 %
whale : 41 %
willow_tree : 11 %
wolf : 18 %
woman : 13 %
worm : 50 %

##

Classification accuracy for each Super class are as follows :

aquatic mammals : 16.4 %
non-insect invertebrates : 30.6 %
food containers : 43.6 %
large natural outdoor scenes : 37.2 %
fruit and vegetables : 31.2 %
medium-sized mammals : 32.8 %
reptiles : 20.2 %
vehicles 1 : 51.6 %
household electrical devices : 46.6 %
vehicles 2 : 40.2 %
trees : 37.0 %
small mammals : 14.8 %
people : 13.8 %
flowers : 22.0 %
insects : 32.2 %
household furniture : 37.4 %
fish : 23.2 %

large man-made outdoor things : 42.0 %
large omnivores and herbivores : 21.8 %
large carnivores : 24.4 %

##

{'aquatic mammals': 82, 'non-insect invertebrates': 153, 'food containers': 218, 'large natural outdoor scenes': 186, 'fruit and vegetables': 156, 'medium-sized mammals': 164, 'reptiles': 101, 'vehicles 1': 258, 'household electrical devices': 233, 'vehicles 2': 201, 'trees': 185, 'small mammals': 74, 'people': 69, 'flowers': 110, 'insects': 161, 'household furniture': 187, 'fish': 116, 'large man-made outdoor things': 210, 'large omnivores and herbivores': 109, 'large carnivores': 122}

##

Rank 5 Classification accuracy for each class are :

apple : 61 %
aquarium_fish : 54 %
baby : 55 %
bear : 62 %
beaver : 51 %
bed : 61 %
bee : 62 %
beetle : 61 %
bicycle : 52 %
bottle : 56 %
bowl : 64 %
boy : 63 %
bridge : 59 %
bus : 64 %
butterfly : 53 %
camel : 61 %
can : 59 %
castle : 58 %
caterpillar : 50 %
cattle : 57 %
chair : 63 %
chimpanzee : 67 %
clock : 62 %
cloud : 55 %
cockroach : 58 %
couch : 64 %
crab : 57 %
crocodile : 62 %
cup : 57 %
dinosaur : 53 %
dolphin : 56 %
elephant : 55 %
flatfish : 55 %
forest : 57 %
fox : 55 %

girl : 57 %
hamster : 57 %
house : 64 %
kangaroo : 61 %
keyboard : 51 %
lamp : 48 %
lawn_mower : 64 %
leopard : 58 %
lion : 61 %
lizard : 59 %
lobster : 47 %
man : 59 %
maple_tree : 60 %
motorcycle : 61 %
mountain : 69 %
mouse : 52 %
mushroom : 57 %
oak_tree : 58 %
orange : 58 %
orchid : 64 %
otter : 59 %
palm_tree : 58 %
pear : 52 %
pickup_truck : 52 %
pine_tree : 58 %
plain : 57 %
plate : 55 %
poppy : 53 %
porcupine : 59 %
possum : 66 %
rabbit : 57 %
raccoon : 54 %
ray : 59 %
road : 63 %
rocket : 55 %
rose : 48 %
sea : 58 %
seal : 55 %
shark : 51 %
shrew : 45 %
skunk : 55 %
skyscraper : 62 %
snail : 53 %
snake : 66 %
spider : 59 %
squirrel : 49 %
streetcar : 57 %
sunflower : 50 %
sweet_pepper : 57 %
table : 54 %
tank : 49 %

telephone : 51 %
television : 56 %
tiger : 62 %
tractor : 58 %
train : 62 %
trout : 49 %
tulip : 52 %
turtle : 60 %
wardrobe : 59 %
whale : 58 %
willow_tree : 56 %
wolf : 53 %
woman : 63 %
worm : 58 %

Examples not classified into any of the super classes are :0

##

Rank 5 Classification accuracy for each Super class are as follows:

aquatic mammals : 55.8 %
non-insect invertebrates : 54.8 %
food containers : 58.2 %
large natural outdoor scenes : 59.2 %
fruit and vegetables : 57.0 %
medium-sized mammals : 57.8 %
reptiles : 60.0 %
vehicles 1 : 58.2 %
household electrical devices : 53.6 %
vehicles 2 : 56.6 %
trees : 58.0 %
small mammals : 52.0 %
people : 59.4 %
flowers : 53.4 %
insects : 56.8 %
household furniture : 60.2 %
fish : 53.6 %
large man-made outdoor things : 61.2 %
large omnivores and herbivores : 60.2 %
large carnivores : 59.2 %

##

##

Overall rank 5 accuracy is :57.26

##