Cam Tran

Professor Xin Ye

CS 497

10 December 2018

Machine Learning Final Project

Introduction

An image classifier is used to identify an image and classify it as a class. For example, an images of a fruit can be classified as banana, apple, pear or grape. Image classification has a wide range of uses including camera software, facial recognition and visual databases. Image classifiers must be trained to become accurate and tuning the training will enhance the accuracy of the classifier.

Problem Definition

With a group of images and no data on their category or class, an image classifier can categorize the images for the user. Similar how people can recognize different vehicles or different car models, through machine learning a computer can categorize it for us. A image classifier requires images to be the input and the classifier will output the category or class. The following tutorial will be used in the project:

https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html

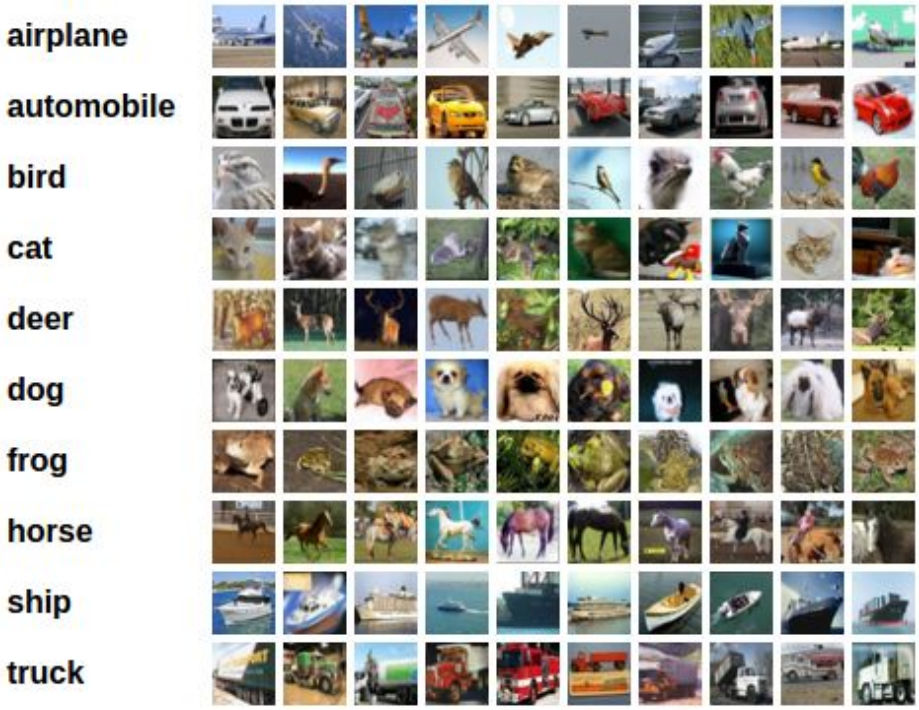
Using the tutorial, the model will be adjusted and the training will be changed to enhance the accuracy of the classifier.

Approach

To increase the accuracy of the classifier, changes to the number of epochs, number of kernels in the convolutional layers and width of the linear layers can be used to increase the accuracy of the image classifier.

Dataset

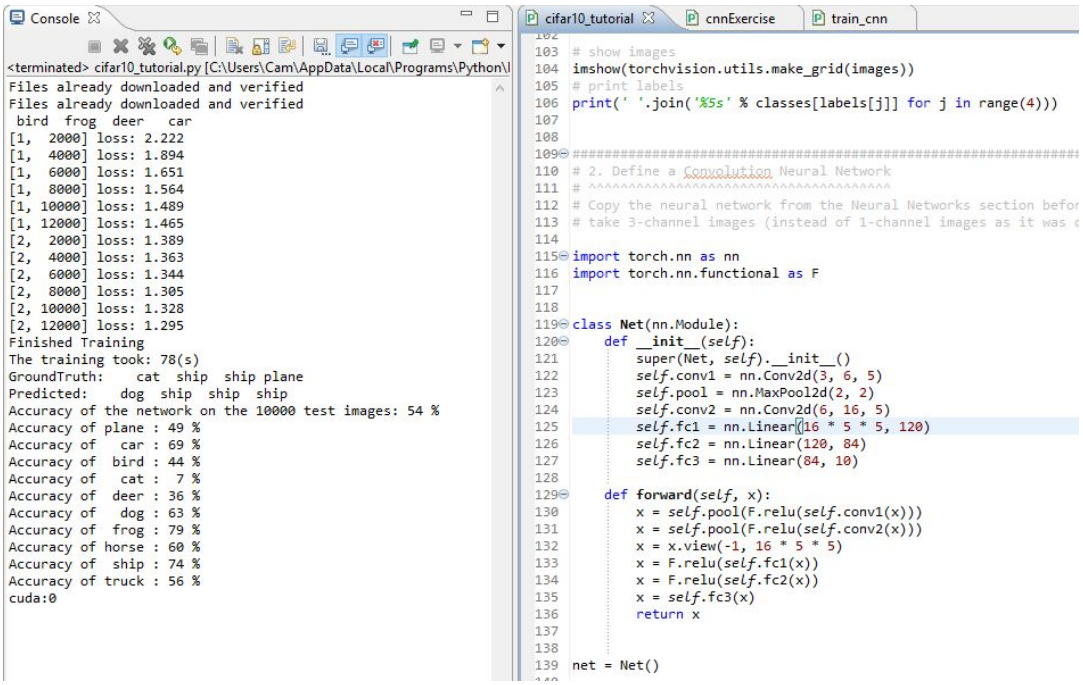
The dataset contains 10000 images, containing the categories of airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck. Each image is of the size 3x32x32 (3 channel color images of 32x32 pixels in size). Changing the dataset may happen, but the size might require changes in filter size.



Experimental Evaluation

The control of the tuning process, an Intel Core i7 4770 was used and GPU CUDA capabilities were not used. To evaluate the tuning process the training time limit was set to a 300 second (5 minute) and the number and type of layers used were not changed. Using the above variables, the program was tuned to have the highest accuracy while still inside the limits. The baseline tutorial had 2 convolutional layers, each having a 2x2 max pool layer, followed by 3 ReLU linear layers. The first convolutional layer had 3 channels, 6 kernels and a 5x5 filter. The second convolutional layer had 6 channels, 16 kernels and a 5x5 filter. The first linear layer had 400 neurons, the second layer had 120 and the last layer had 84. Comparably to the original, the tuned classifier had the same structure but changed the convolutional layers and the first two linear layers. The first convolutional layer changed from 6 kernels to 25. The second convolutional layer changed the channels to 25 and the kernels to 40. Lastly, the linear layers change from 400 to 1000 neurons and from 120 to 180 neurons. The classifier originally had a 54% accuracy that took 78 seconds to train. After tuning, the accuracy increased to 66%, while the training took 291 seconds. Overall it is a 12% increase in accuracy at the cost of a 373% increase in training time. Below is the neural network and output of each test:

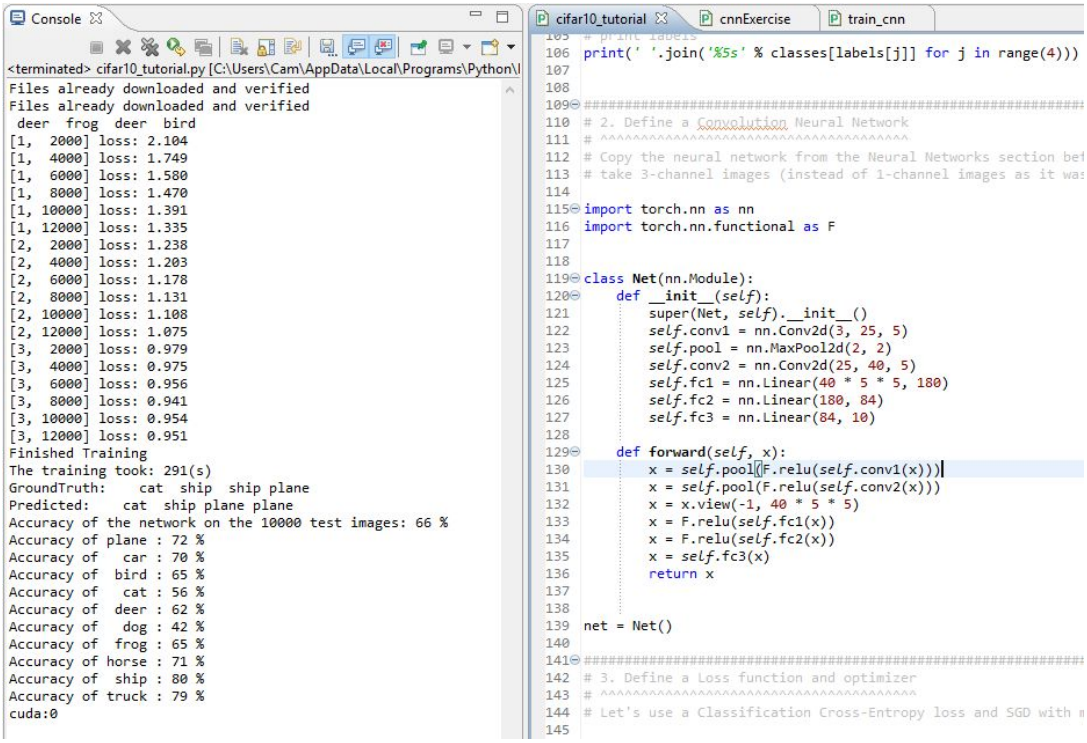
Base



```
Console
<terminated> cifar10_tutorial.py [C:\Users\Cam\AppData\Local\Programs\Python\I
Files already downloaded and verified
Files already downloaded and verified
 bird  frog  deer   car
[1,  2000] loss: 2.222
[1,  4000] loss: 1.894
[1,  6000] loss: 1.651
[1,  8000] loss: 1.564
[1, 10000] loss: 1.489
[1, 12000] loss: 1.465
[2,  2000] loss: 1.389
[2,  4000] loss: 1.363
[2,  6000] loss: 1.344
[2,  8000] loss: 1.305
[2, 10000] loss: 1.328
[2, 12000] loss: 1.295
Finished Training
The training took: 78(s)
GroundTruth:    cat  ship  ship plane
Predicted:    dog  ship  ship  ship
Accuracy of the network on the 10000 test images: 54 %
Accuracy of plane : 49 %
Accuracy of   car : 69 %
Accuracy of  bird : 44 %
Accuracy of   cat :  7 %
Accuracy of  deer : 36 %
Accuracy of   dog : 63 %
Accuracy of  frog : 79 %
Accuracy of horse : 60 %
Accuracy of  ship : 74 %
Accuracy of truck : 56 %
cuda:0
```

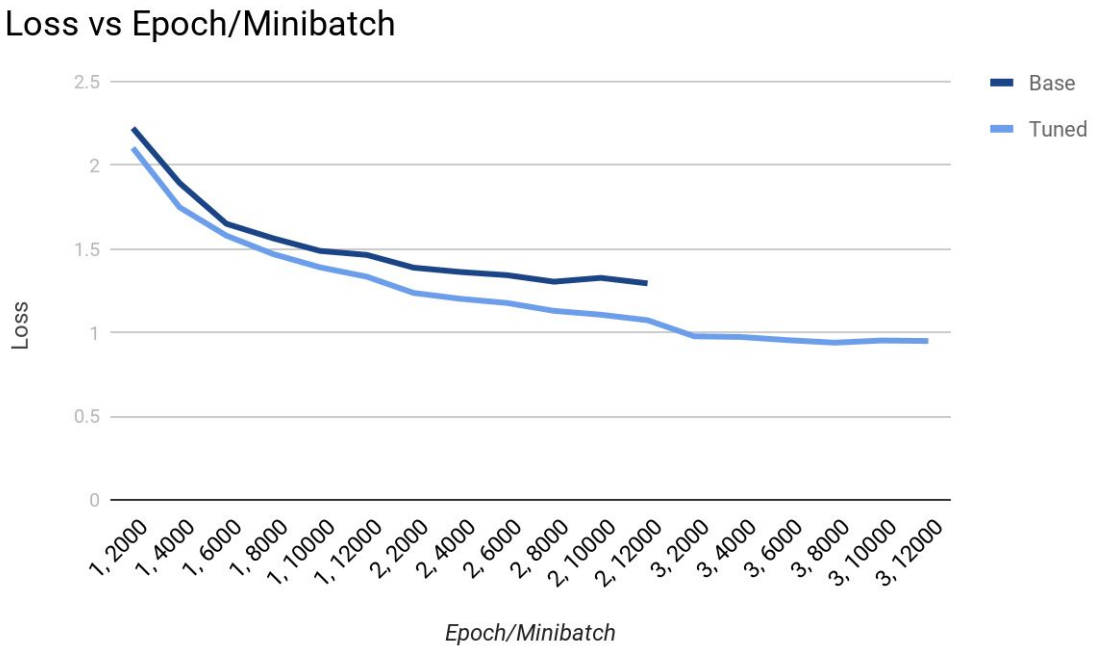
```
cifar10_tutorial | cnnExercise | train_cnn
103 # show images
104 imshow(torchvision.utils.make_grid(images))
105 # print labels
106 print(' '.join('%5s' % classes[labels[j]] for j in range(4)))
107
108
109 ########################################################################
110 # 2. Define a Convolution Neural Network
111 # ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
112 # Copy the neural network from the Neural Networks section befor
113 # take 3-channel images (instead of 1-channel images as it was d
114
115 import torch.nn as nn
116 import torch.nn.functional as F
117
118
119 class Net(nn.Module):
120     def __init__(self):
121         super(Net, self).__init__()
122         self.conv1 = nn.Conv2d(3, 6, 5)
123         self.pool = nn.MaxPool2d(2, 2)
124         self.conv2 = nn.Conv2d(6, 16, 5)
125         self.fc1 = nn.Linear(16 * 5 * 5, 120)
126         self.fc2 = nn.Linear(120, 84)
127         self.fc3 = nn.Linear(84, 10)
128
129     def forward(self, x):
130         x = self.pool(F.relu(self.conv1(x)))
131         x = self.pool(F.relu(self.conv2(x)))
132         x = x.view(-1, 16 * 5 * 5)
133         x = F.relu(self.fc1(x))
134         x = F.relu(self.fc2(x))
135         x = self.fc3(x)
136         return x
137
138
139 net = Net()
```

Tuned



```
Console
<terminated> cifar10_tutorial.py [C:\Users\Cam\AppData\Local\Programs\Python\I
Files already downloaded and verified
Files already downloaded and verified
 deer  frog  deer  bird
[1,  2000] loss: 2.104
[1,  4000] loss: 1.749
[1,  6000] loss: 1.580
[1,  8000] loss: 1.470
[1, 10000] loss: 1.391
[1, 12000] loss: 1.335
[2,  2000] loss: 1.238
[2,  4000] loss: 1.203
[2,  6000] loss: 1.178
[2,  8000] loss: 1.131
[2, 10000] loss: 1.108
[2, 12000] loss: 1.075
[3,  2000] loss: 0.979
[3,  4000] loss: 0.975
[3,  6000] loss: 0.956
[3,  8000] loss: 0.941
[3, 10000] loss: 0.954
[3, 12000] loss: 0.951
Finished Training
The training took: 291(s)
GroundTruth:    cat  ship  ship plane
Predicted:    cat  ship plane plane
Accuracy of the network on the 10000 test images: 66 %
Accuracy of plane : 72 %
Accuracy of   car : 70 %
Accuracy of  bird : 65 %
Accuracy of   cat : 56 %
Accuracy of  deer : 62 %
Accuracy of   dog : 42 %
Accuracy of  frog : 65 %
Accuracy of horse : 71 %
Accuracy of  ship : 80 %
Accuracy of truck : 79 %
cuda:0
```

```
cifar10_tutorial | cnnExercise | train_cnn
106 print(' '.join('%5s' % classes[labels[j]] for j in range(4)))
107
108
109 ########################################################################
110 # 2. Define a Convolution Neural Network
111 # ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
112 # Copy the neural network from the Neural Networks section bef
113 # take 3-channel images (instead of 1-channel images as it was
114
115 import torch.nn as nn
116 import torch.nn.functional as F
117
118
119 class Net(nn.Module):
120     def __init__(self):
121         super(Net, self).__init__()
122         self.conv1 = nn.Conv2d(3, 25, 5)
123         self.pool = nn.MaxPool2d(2, 2)
124         self.conv2 = nn.Conv2d(25, 40, 5)
125         self.fc1 = nn.Linear(40 * 5 * 5, 180)
126         self.fc2 = nn.Linear(180, 84)
127         self.fc3 = nn.Linear(84, 10)
128
129     def forward(self, x):
130         x = self.pool(F.relu(self.conv1(x)))
131         x = self.pool(F.relu(self.conv2(x)))
132         x = x.view(-1, 40 * 5 * 5)
133         x = F.relu(self.fc1(x))
134         x = F.relu(self.fc2(x))
135         x = self.fc3(x)
136         return x
137
138
139 net = Net()
140
141 ########################################################################
142 # 3. Define a Loss function and optimizer
143 # ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
144 # Let's use a Classification Cross-Entropy loss and SGD with m
145
```



Contribution

Changes to the width of the layers and number of epochs were made to tune the neural network in order to increase accuracy.

Future Work

Originally it was not expected to only receive a 12% increase in accuracy. Additional linear layers could have possibly increased the accuracy further, but the width of the convolutional layers would have to be reduced to meet the 300 second training time limit. Lastly, further increasing the accuracy to 90% through tuning would be an interesting task.

Conclusion

In conclusion, a 12% increase in accuracy is a good amount for the limited training time and such a large data set. The accuracy of the neural network could be higher, but would require a significant increase in the size of the network and time spent training.