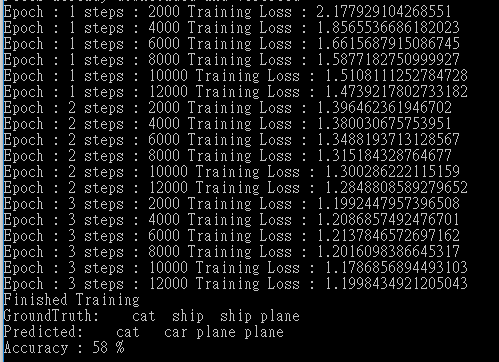
Exercise 2-6 CIFAR-10

Jirayu Petchhan, D10907801

Example

**Default; Epochs=3, learning rate=0.001**



Result: Accuracy at default setting is 58%

Exercise

Changing to Epochs=4, learning rate=0.005

Coding

########################################################################

# 3. Define a Loss function and optimizer

# ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

# Let's use a Classification Cross-Entropy loss and SGD with momentum.

criterion = nn.CrossEntropyLoss()

optimizer = optim.SGD(net.parameters(), lr=0.005, momentum=0.9)

########################################################################

# 4. Train the network

# ^^^^^^^^^^^^^^^^^^^^

#

# This is when things start to get interesting.

# We simply have to loop over our data iterator, and feed the inputs to the

# network and optimize.

for epoch in range(4): # loop over the dataset multiple times

running\_loss = 0.0

for i, data in enumerate(trainloader, 0):

# get the inputs; data is a list of [inputs, labels]

inputs, labels = data

# zero the parameter gradients

optimizer.zero\_grad()

# forward + backward + optimize

outputs = net(inputs)

loss = criterion(outputs, labels)

loss.backward()

optimizer.step()

# print statistics

running\_loss += loss.item()

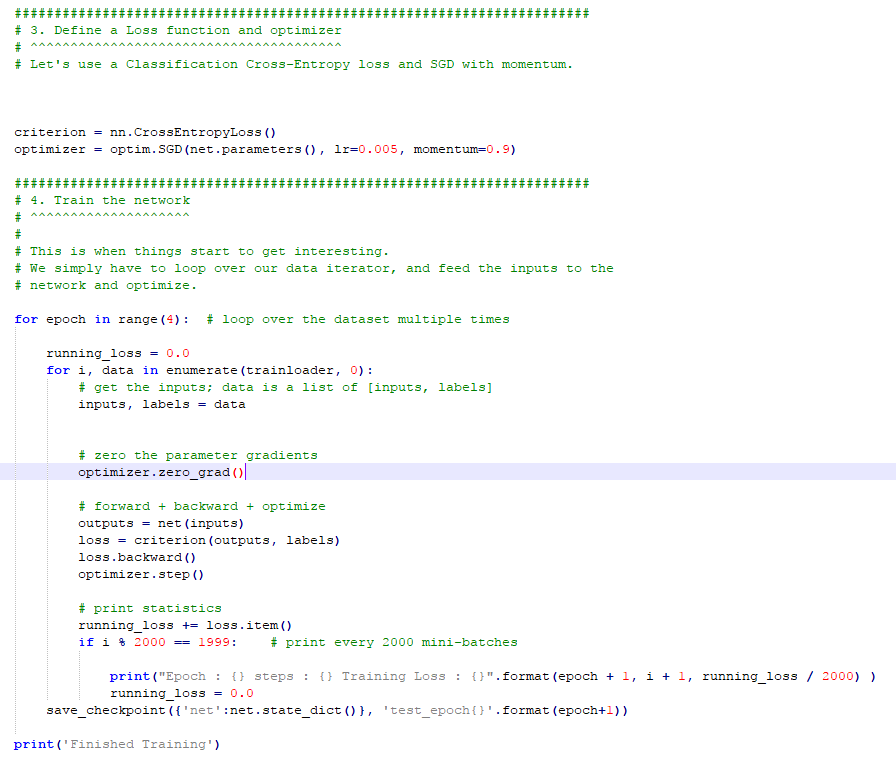
if i % 2000 == 1999: # print every 2000 mini-batches

print("Epoch : {} steps : {} Training Loss : {}".format(epoch + 1, i + 1, running\_loss / 2000) )

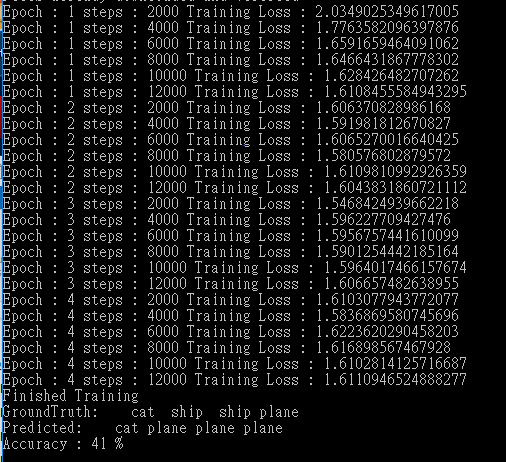
running\_loss = 0.0

save\_checkpoint({'net':net.state\_dict()}, 'test\_epoch{}'.format(epoch+1))

print('Finished Training')

Code displaying

Training model



Result: Accuracy at default setting is 41%

Conclusion

When comparing with default model, the accuracy is less than old once but Training Loss have been raised from 1.200 to 1.611