With PyTorch 0.2.0_2

PYTORCH

Lab 10

NN, Xavier, Dropout

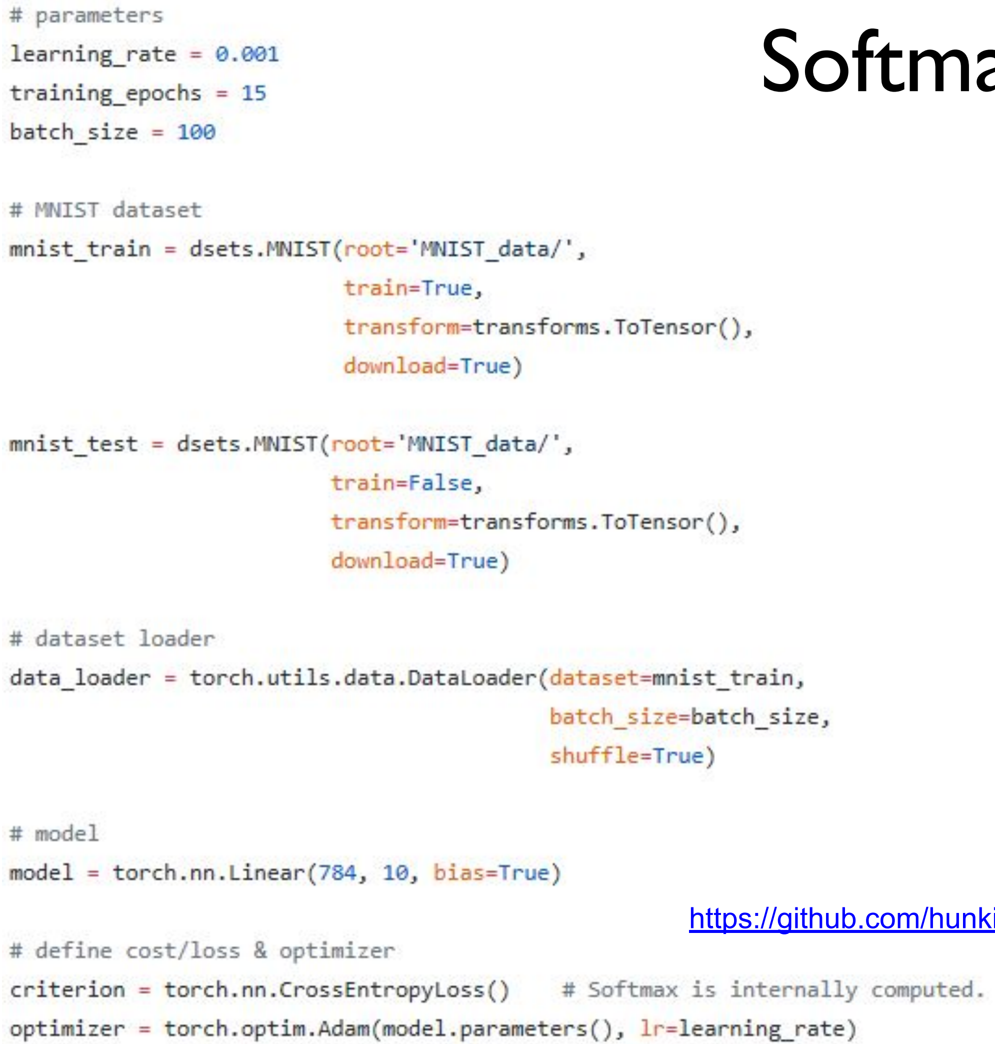
Sung Kim <hunkim+ml@gmail.com>

Code: https://github.com/hunkim/DeepLearningZeroToAll/

https://github.com/hunkim/DeepLearningZeroToAll/tree/master/pytorch

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-10-1-mnist_softmax.py

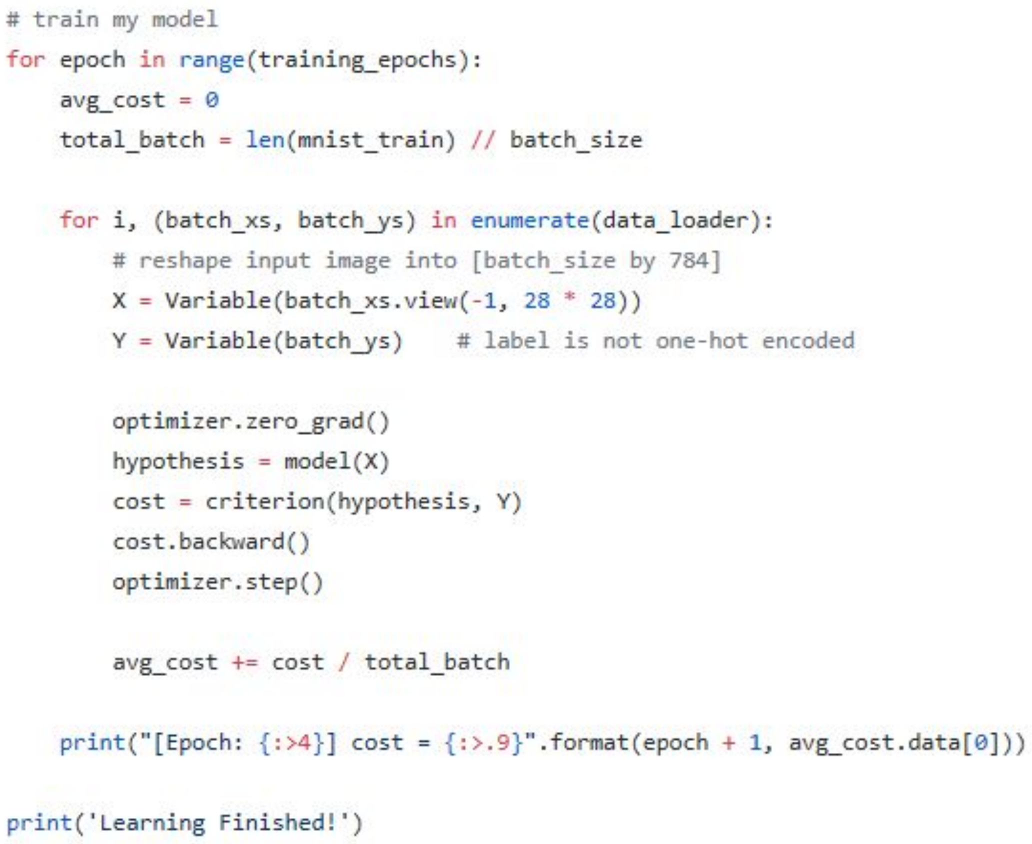
Softmax classifier for MNIST



```
# parameters
learning_rate = 0.001
training_epochs = 15
batch_size = 100

# MNIST dataset
mnist_train = dsets.MNIST(root='MNIST_data/',
                          train=True,
                          transform=transforms.ToTensor(),
                          download=True)

mnist_test = dsets.MNIST(root='MNIST_data/',
                         train=False,
                         transform=transforms.ToTensor(),
                         download=True)

# dataset loader
data_loader = torch.utils.data.DataLoader(dataset=mnist_train,
                                          batch_size=batch_size,
                                          shuffle=True)

# model
model = torch.nn.Linear(784, 10, bias=True)

# define cost/loss & optimizer
criterion = torch.nn.CrossEntropyLoss()    # Softmax is internally computed.
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-10-1-mnist_softmax.py

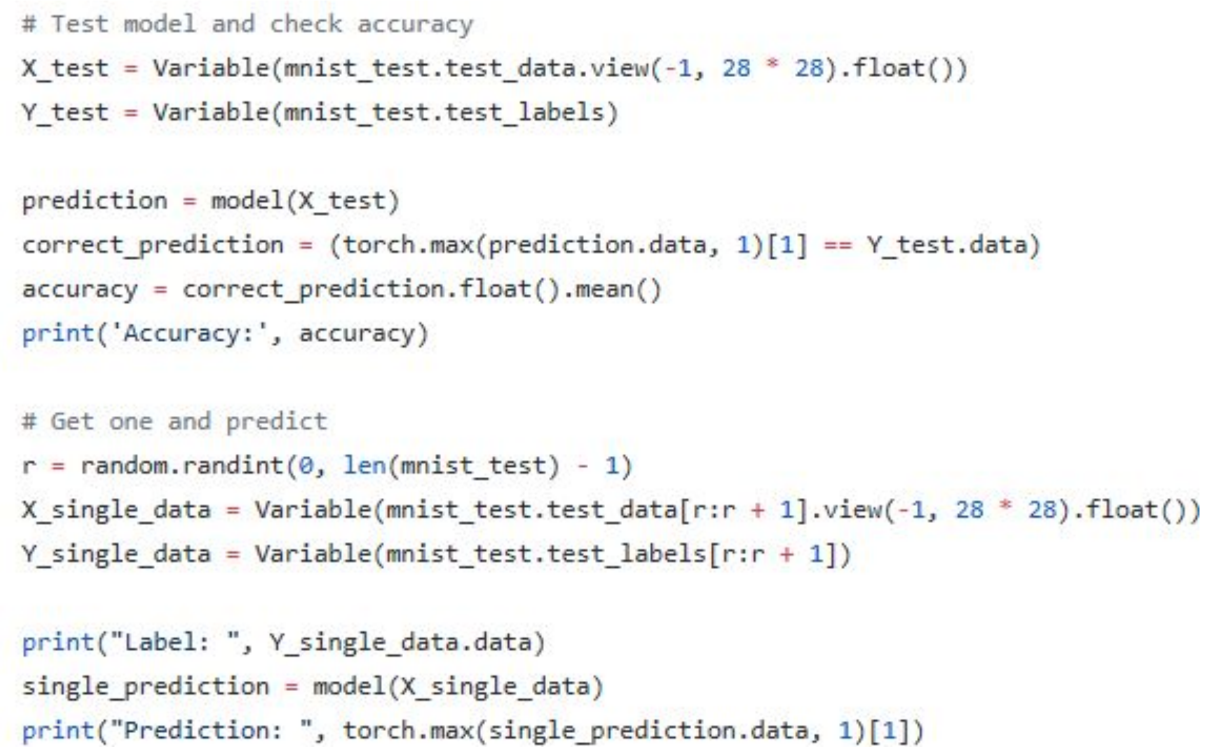
Softmax classifier for MNIST



```
# train my model
for epoch in range(training_epochs):
    avg_cost = 0
    total_batch = len(mnist_train) // batch_size

    for i, (batch_xs, batch_ys) in enumerate(data_loader):
        # reshape input image into [batch_size by 784]
        X = Variable(batch_xs.view(-1, 28 * 28))
        Y = Variable(batch_ys)    # label is not one-hot encoded

        optimizer.zero_grad()
        hypothesis = model(X)
        cost = criterion(hypothesis, Y)
        cost.backward()
        optimizer.step()

        avg_cost += cost / total_batch

    print("[Epoch: {:>4}] cost = {:>.9}".format(epoch + 1, avg_cost.data[0]))

print('Learning Finished!')
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-10-1-mnist_softmax.py

Softmax classifier for MNIST



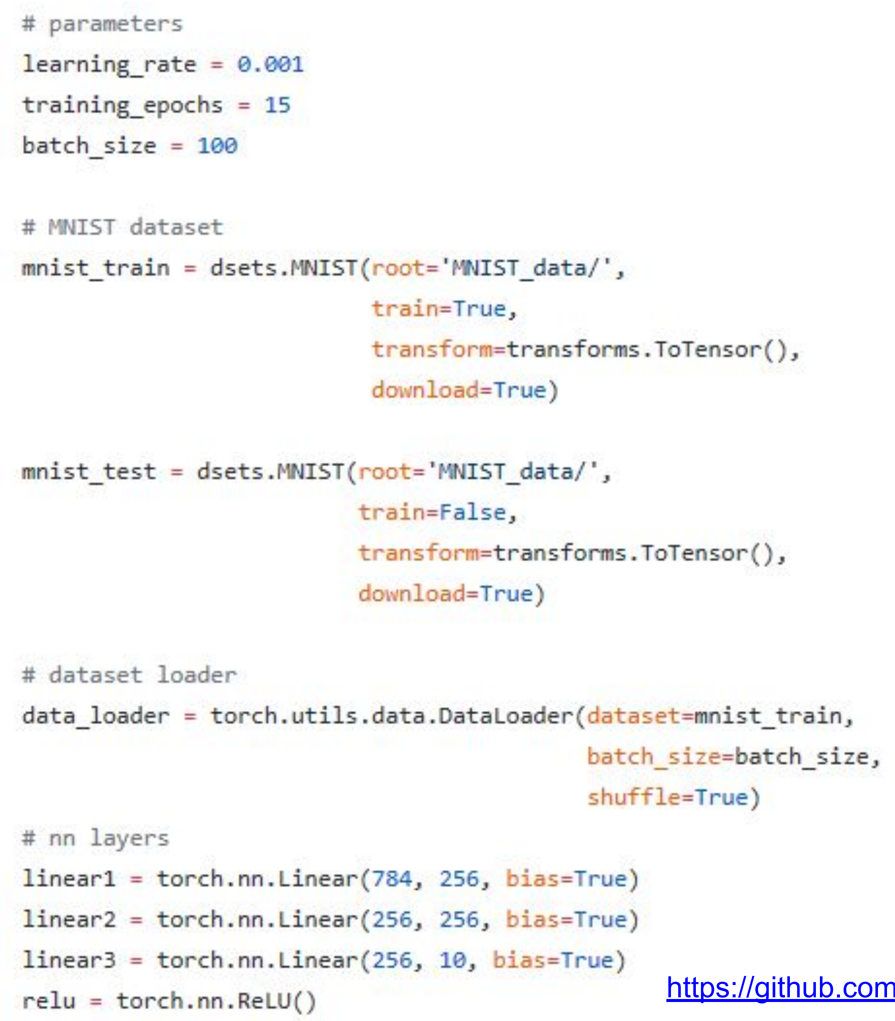
```
# Test model and check accuracy
X_test = Variable(mnist_test.test_data.view(-1, 28 * 28).float())
Y_test = Variable(mnist_test.test_labels)

prediction = model(X_test)
correct_prediction = (torch.max(prediction.data, 1)[1] == Y_test.data)
accuracy = correct_prediction.float().mean()
print('Accuracy:', accuracy)

# Get one and predict
r = random.randint(0, len(mnist_test) - 1)
X_single_data = Variable(mnist_test.test_data[r:r + 1].view(-1, 28 * 28).float())
Y_single_data = Variable(mnist_test.test_labels[r:r + 1])

print("Label: ", Y_single_data.data)
single_prediction = model(X_single_data)
print("Prediction: ", torch.max(single_prediction.data, 1)[1])
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-10-1-mnist_softmax.py

https://github.com/hunkim/DeepLearningZeroToAll/tree/master/pytorch

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-10-2-mnist_nn.py

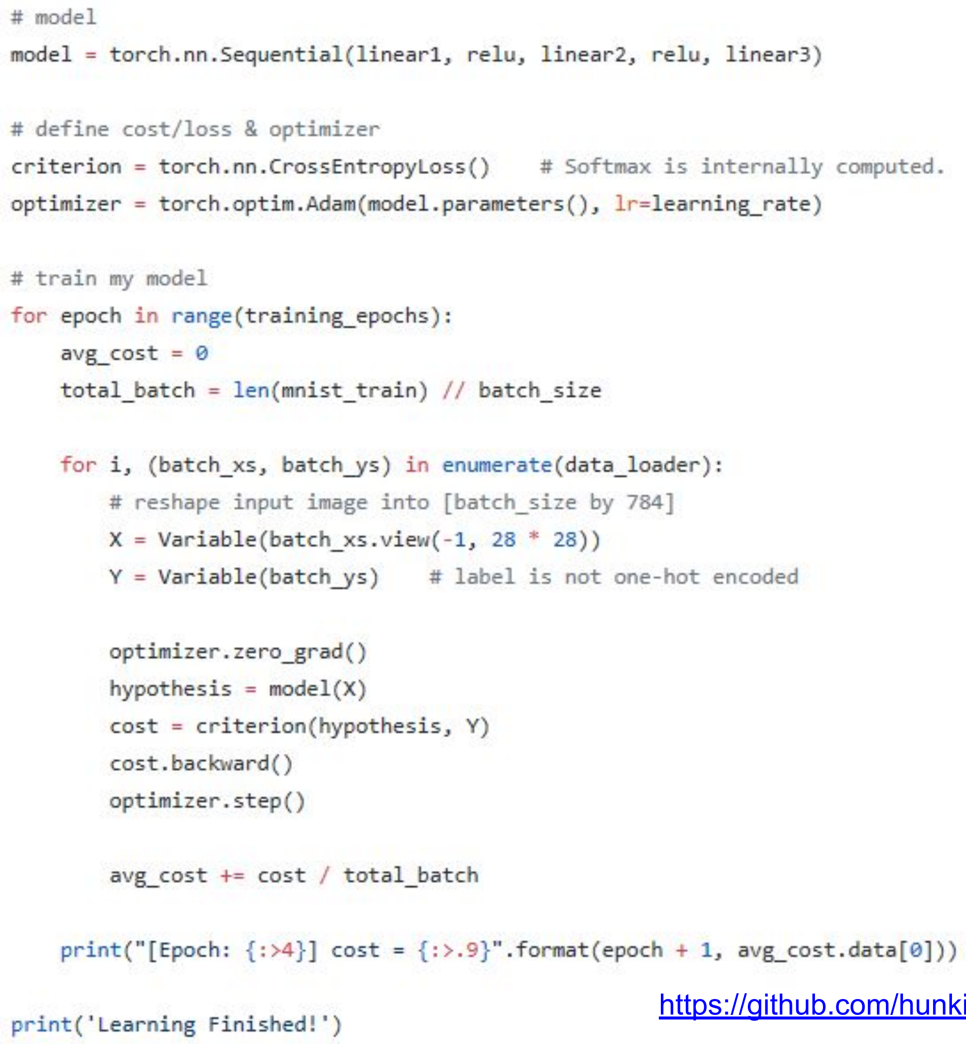
NN for MNIST



```
# parameters
learning_rate = 0.001
training_epochs = 15
batch_size = 100

# MNIST dataset
mnist_train = dsets.MNIST(root='MNIST_data/',
                          train=True,
                          transform=transforms.ToTensor(),
                          download=True)

mnist_test = dsets.MNIST(root='MNIST_data/',
                         train=False,
                         transform=transforms.ToTensor(),
                         download=True)

# dataset loader
data_loader = torch.utils.data.DataLoader(dataset=mnist_train,
                                          batch_size=batch_size,
                                          shuffle=True)
# nn layers
linear1 = torch.nn.Linear(784, 256, bias=True)
linear2 = torch.nn.Linear(256, 256, bias=True)
linear3 = torch.nn.Linear(256, 10, bias=True)
relu = torch.nn.ReLU()
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-10-2-mnist_nn.py

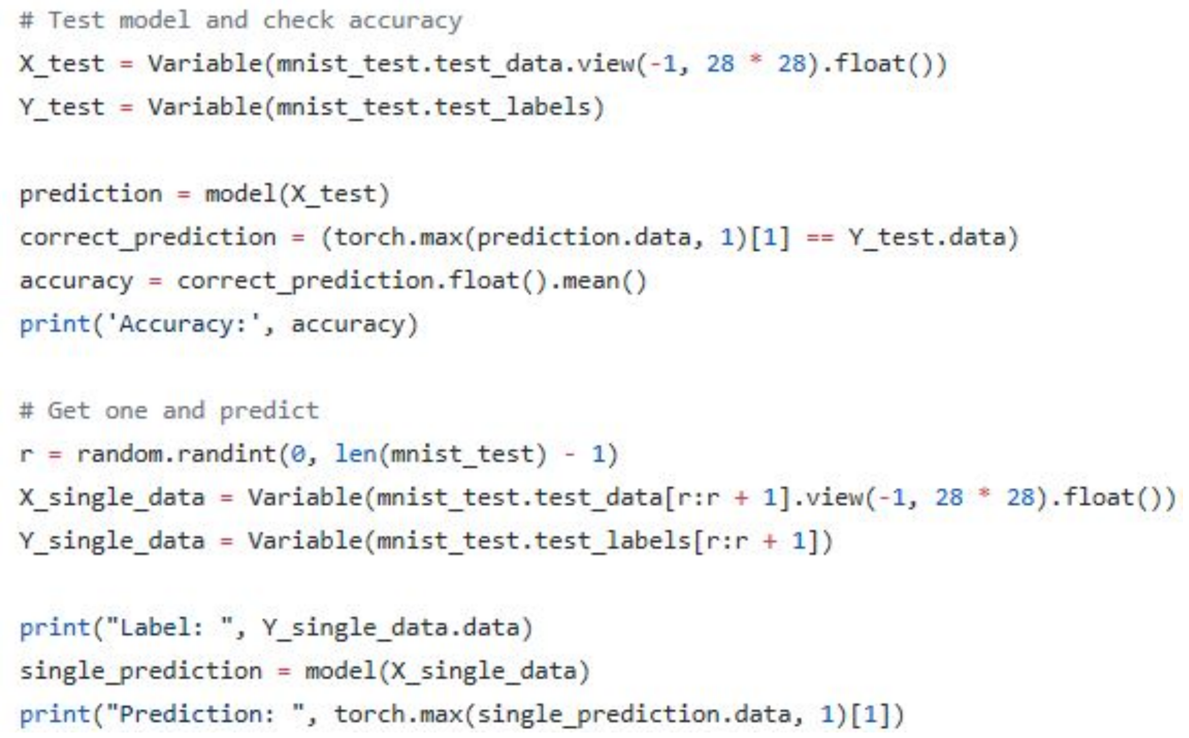
NN for MNIST



```
# model
model = torch.nn.Sequential(linear1, relu, linear2, relu, linear3)

# define cost/loss & optimizer
criterion = torch.nn.CrossEntropyLoss()    # Softmax is internally computed.
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)

# train my model
for epoch in range(training_epochs):
    avg_cost = 0
    total_batch = len(mnist_train) // batch_size

    for i, (batch_xs, batch_ys) in enumerate(data_loader):
        # reshape input image into [batch_size by 784]
        X = Variable(batch_xs.view(-1, 28 * 28))
        Y = Variable(batch_ys)    # label is not one-hot encoded

        optimizer.zero_grad()
        hypothesis = model(X)
        cost = criterion(hypothesis, Y)
        cost.backward()
        optimizer.step()

        avg_cost += cost / total_batch

    print("[Epoch: {:>4}] cost = {:>.9}".format(epoch + 1, avg_cost.data[0]))

print('Learning Finished!')
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-10-2-mnist_nn.py

NN for MNIST



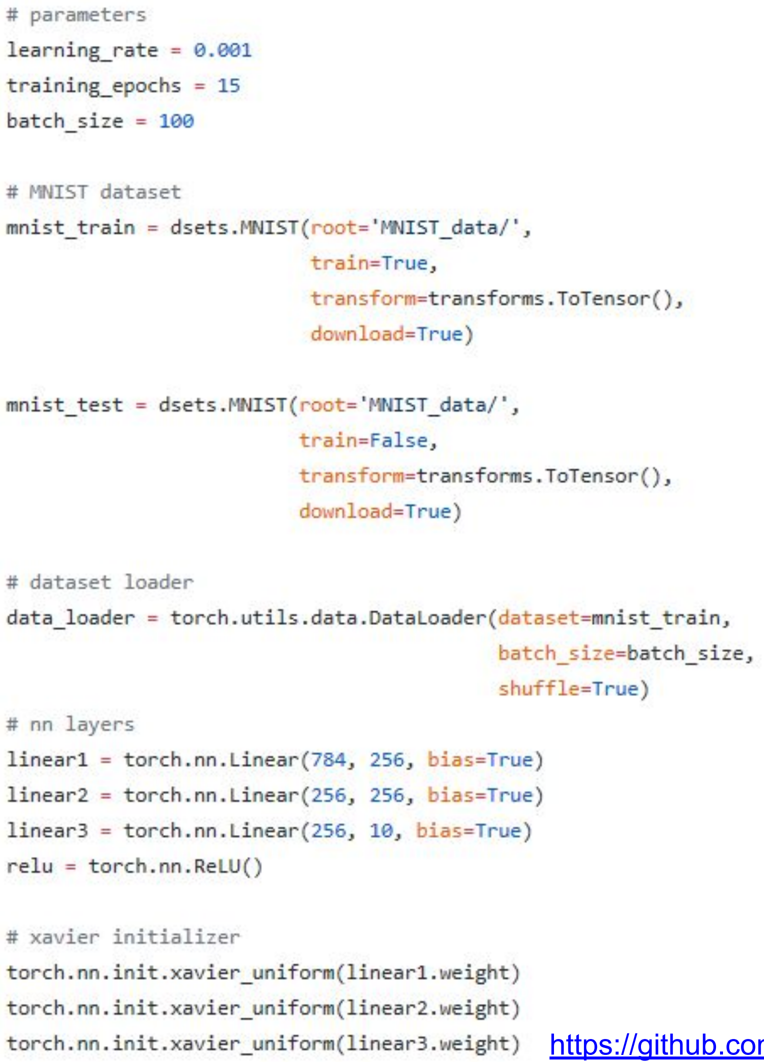
```
# Test model and check accuracy
X_test = Variable(mnist_test.test_data.view(-1, 28 * 28).float())
Y_test = Variable(mnist_test.test_labels)

prediction = model(X_test)
correct_prediction = (torch.max(prediction.data, 1)[1] == Y_test.data)
accuracy = correct_prediction.float().mean()
print('Accuracy:', accuracy)

# Get one and predict
r = random.randint(0, len(mnist_test) - 1)
X_single_data = Variable(mnist_test.test_data[r:r + 1].view(-1, 28 * 28).float())
Y_single_data = Variable(mnist_test.test_labels[r:r + 1])

print("Label: ", Y_single_data.data)
single_prediction = model(X_single_data)
print("Prediction: ", torch.max(single_prediction.data, 1)[1])
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-10-2-mnist_nn.py

https://github.com/hunkim/DeepLearningZeroToAll/tree/master/pytorch

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-10-3-mnist_nn_xavier.py

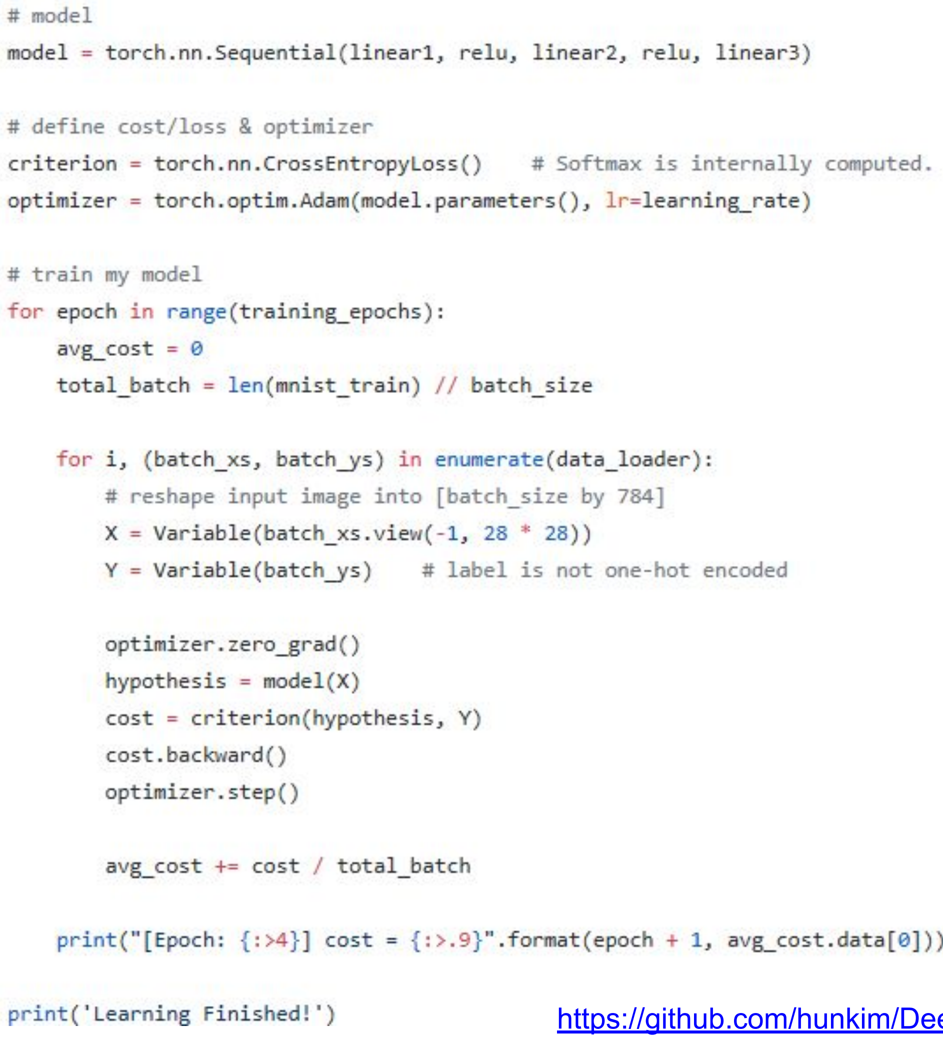
Xavier for MNIST



```
# parameters
learning_rate = 0.001
training_epochs = 15
batch_size = 100

# MNIST dataset
mnist_train = dsets.MNIST(root='MNIST_data/',
                          train=True,
                          transform=transforms.ToTensor(),
                          download=True)

mnist_test = dsets.MNIST(root='MNIST_data/',
                         train=False,
                         transform=transforms.ToTensor(),
                         download=True)

# dataset loader
data_loader = torch.utils.data.DataLoader(dataset=mnist_train,
                                          batch_size=batch_size,
                                          shuffle=True)
# nn layers
linear1 = torch.nn.Linear(784, 256, bias=True)
linear2 = torch.nn.Linear(256, 256, bias=True)
linear3 = torch.nn.Linear(256, 10, bias=True)
relu = torch.nn.ReLU()

# xavier initializer
torch.nn.init.xavier_uniform(linear1.weight)
torch.nn.init.xavier_uniform(linear2.weight)
torch.nn.init.xavier_uniform(linear3.weight)
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-10-3-mnist_nn_xavier.py

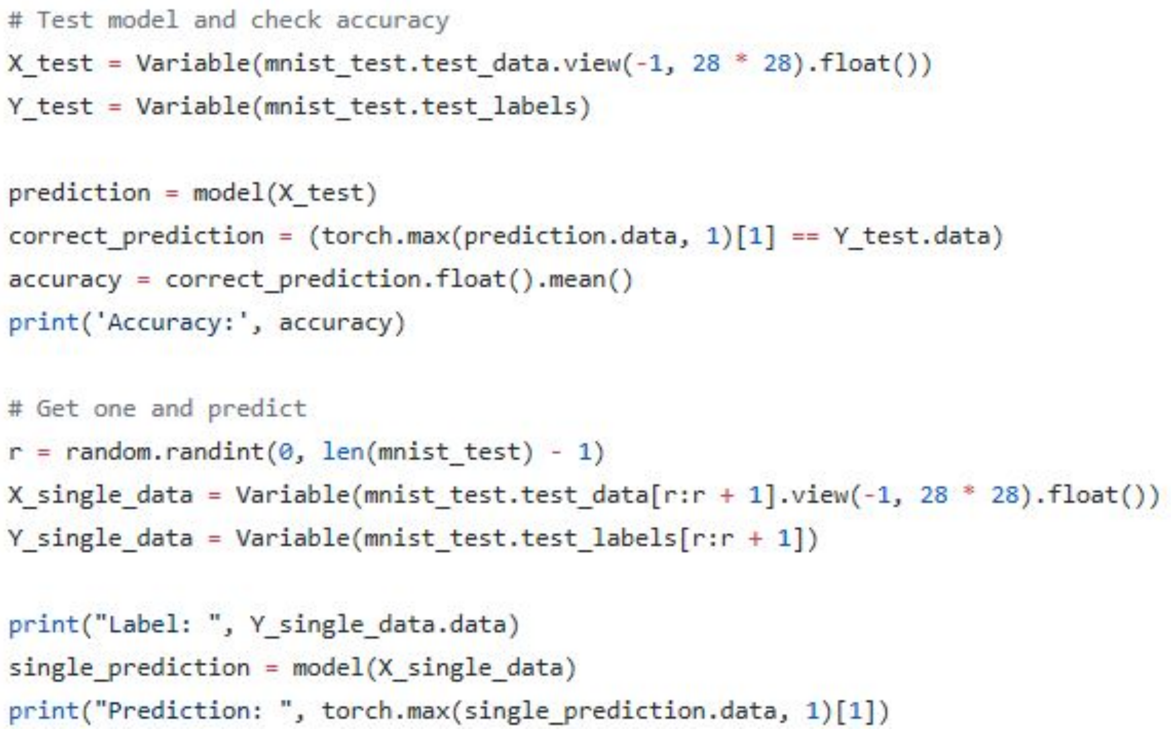
Xavier for MNIST



```
# model
model = torch.nn.Sequential(linear1, relu, linear2, relu, linear3)

# define cost/loss & optimizer
criterion = torch.nn.CrossEntropyLoss()    # Softmax is internally computed.
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)

# train my model
for epoch in range(training_epochs):
    avg_cost = 0
    total_batch = len(mnist_train) // batch_size

    for i, (batch_xs, batch_ys) in enumerate(data_loader):
        # reshape input image into [batch_size by 784]
        X = Variable(batch_xs.view(-1, 28 * 28))
        Y = Variable(batch_ys)    # label is not one-hot encoded

        optimizer.zero_grad()
        hypothesis = model(X)
        cost = criterion(hypothesis, Y)
        cost.backward()
        optimizer.step()

        avg_cost += cost / total_batch

    print("[Epoch: {:>4}] cost = {:>.9}".format(epoch + 1, avg_cost.data[0]))

print('Learning Finished!')
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-10-3-mnist_nn_xavier.py

Xavier for MNIST



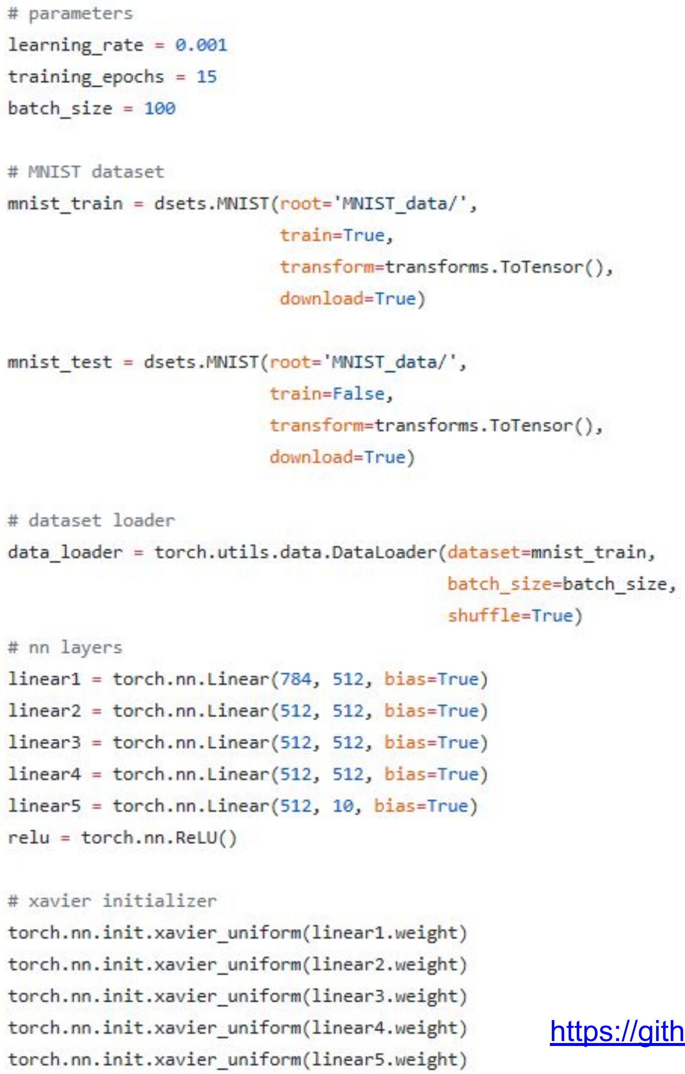
```
# Test model and check accuracy
X_test = Variable(mnist_test.test_data.view(-1, 28 * 28).float())
Y_test = Variable(mnist_test.test_labels)

prediction = model(X_test)
correct_prediction = (torch.max(prediction.data, 1)[1] == Y_test.data)
accuracy = correct_prediction.float().mean()
print('Accuracy:', accuracy)

# Get one and predict
r = random.randint(0, len(mnist_test) - 1)
X_single_data = Variable(mnist_test.test_data[r:r + 1].view(-1, 28 * 28).float())
Y_single_data = Variable(mnist_test.test_labels[r:r + 1])

print("Label: ", Y_single_data.data)
single_prediction = model(X_single_data)
print("Prediction: ", torch.max(single_prediction.data, 1)[1])
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-10-3-mnist_nn_xavier.py

https://github.com/hunkim/DeepLearningZeroToAll/tree/master/pytorch

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-10-4-mnist_nn_deep.py

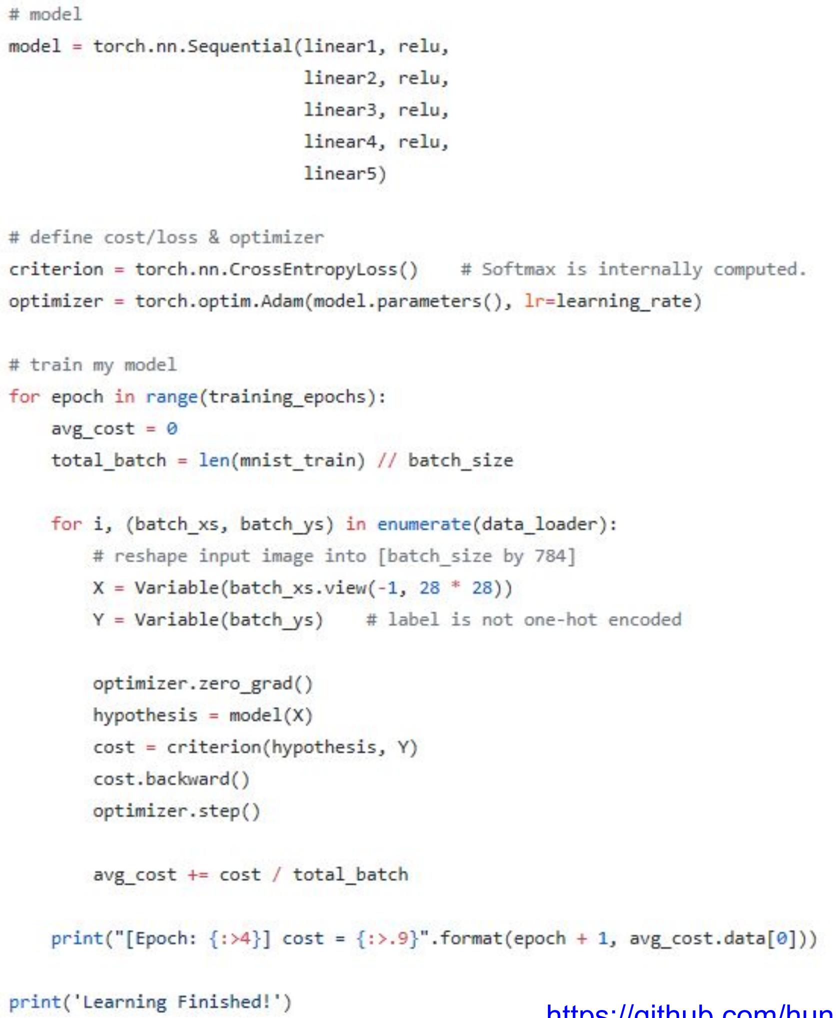
Deep NN for MNIST



```
# parameters
learning_rate = 0.001
training_epochs = 15
batch_size = 100

# MNIST dataset
mnist_train = dsets.MNIST(root='MNIST_data/',
                          train=True,
                          transform=transforms.ToTensor(),
                          download=True)

mnist_test = dsets.MNIST(root='MNIST_data/',
                         train=False,
                         transform=transforms.ToTensor(),
                         download=True)

# dataset loader
data_loader = torch.utils.data.DataLoader(dataset=mnist_train,
                                          batch_size=batch_size,
                                          shuffle=True)
# nn layers
linear1 = torch.nn.Linear(784, 512, bias=True)
linear2 = torch.nn.Linear(512, 512, bias=True)
linear3 = torch.nn.Linear(512, 512, bias=True)
linear4 = torch.nn.Linear(512, 512, bias=True)
linear5 = torch.nn.Linear(512, 10, bias=True)
relu = torch.nn.ReLU()

# xavier initializer
torch.nn.init.xavier_uniform(linear1.weight)
torch.nn.init.xavier_uniform(linear2.weight)
torch.nn.init.xavier_uniform(linear3.weight)
torch.nn.init.xavier_uniform(linear4.weight)
torch.nn.init.xavier_uniform(linear5.weight)
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-10-4-mnist_nn_deep.py

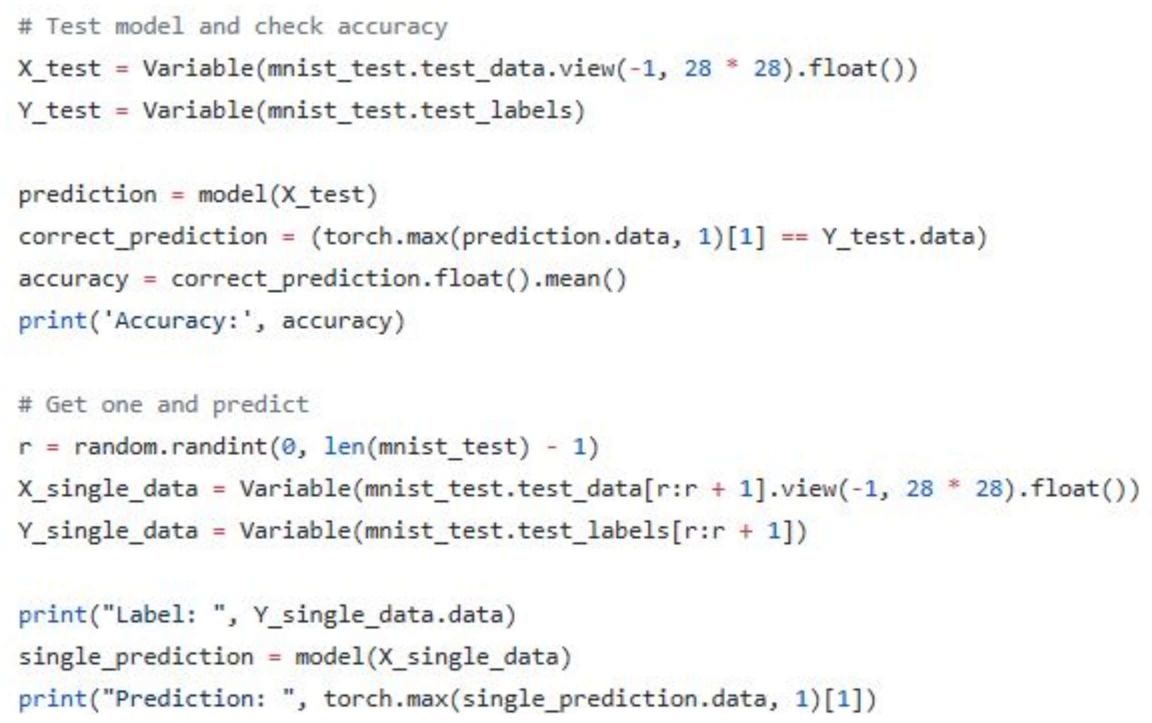
Deep NN for MNIST



```
# model
model = torch.nn.Sequential(linear1, relu,
                            linear2, relu,
                            linear3, relu,
                            linear4, relu,
                            linear5)

# define cost/loss & optimizer
criterion = torch.nn.CrossEntropyLoss()    # Softmax is internally computed.
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)

# train my model
for epoch in range(training_epochs):
    avg_cost = 0
    total_batch = len(mnist_train) // batch_size

    for i, (batch_xs, batch_ys) in enumerate(data_loader):
        # reshape input image into [batch_size by 784]
        X = Variable(batch_xs.view(-1, 28 * 28))
        Y = Variable(batch_ys)    # label is not one-hot encoded

        optimizer.zero_grad()
        hypothesis = model(X)
        cost = criterion(hypothesis, Y)
        cost.backward()
        optimizer.step()

        avg_cost += cost / total_batch

    print("[Epoch: {:>4}] cost = {:>.9}".format(epoch + 1, avg_cost.data[0]))

print('Learning Finished!')
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-10-4-mnist_nn_deep.py

Deep NN for MNIST



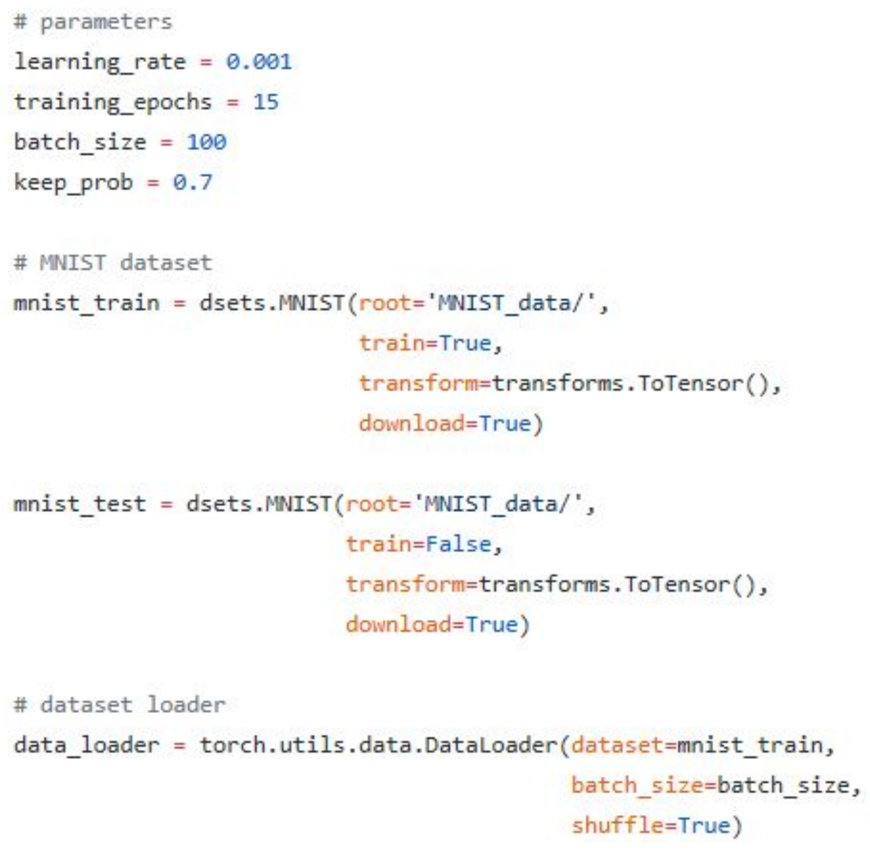
```
# Test model and check accuracy
X_test = Variable(mnist_test.test_data.view(-1, 28 * 28).float())
Y_test = Variable(mnist_test.test_labels)

prediction = model(X_test)
correct_prediction = (torch.max(prediction.data, 1)[1] == Y_test.data)
accuracy = correct_prediction.float().mean()
print('Accuracy:', accuracy)

# Get one and predict
r = random.randint(0, len(mnist_test) - 1)
X_single_data = Variable(mnist_test.test_data[r:r + 1].view(-1, 28 * 28).float())
Y_single_data = Variable(mnist_test.test_labels[r:r + 1])

print("Label: ", Y_single_data.data)
single_prediction = model(X_single_data)
print("Prediction: ", torch.max(single_prediction.data, 1)[1])
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-10-4-mnist_nn_deep.py

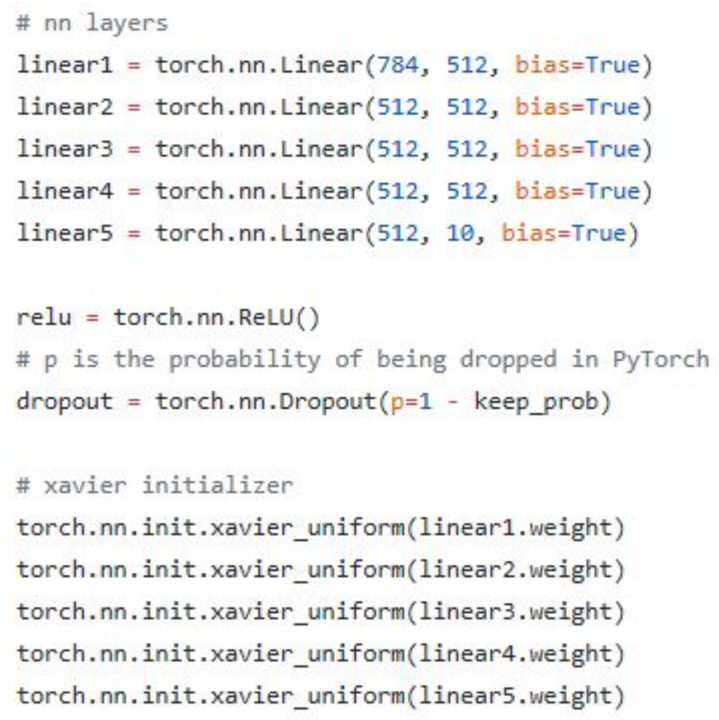
https://github.com/hunkim/DeepLearningZeroToAll/tree/master/pytorch

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-10-5-mnist_nn_dropout.py

Deep NN with Dropout for MNIST



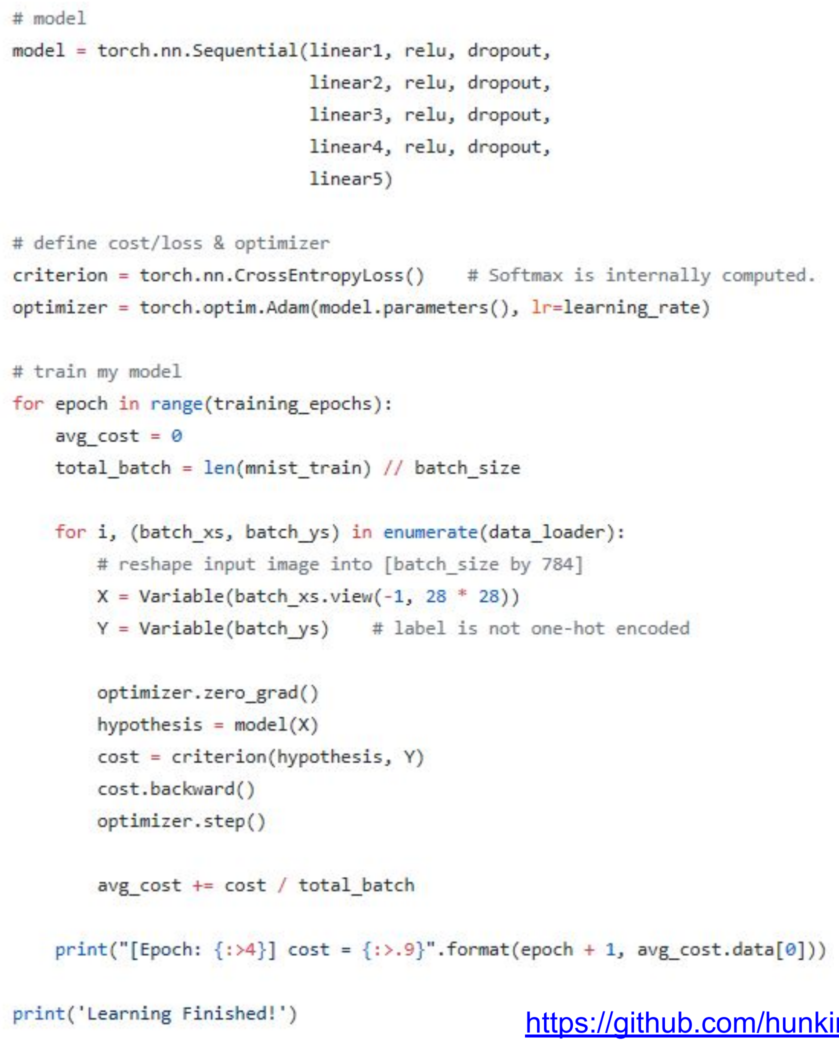
```
# parameters
learning_rate = 0.001
training_epochs = 15
batch_size = 100
keep_prob = 0.7

# MNIST dataset
mnist_train = dsets.MNIST(root='MNIST_data/',
                          train=True,
                          transform=transforms.ToTensor(),
                          download=True)

mnist_test = dsets.MNIST(root='MNIST_data/',
                         train=False,
                         transform=transforms.ToTensor(),
                         download=True)

# dataset loader
data_loader = torch.utils.data.DataLoader(dataset=mnist_train,
                                          batch_size=batch_size,
                                          shuffle=True)
```



```
# nn layers
linear1 = torch.nn.Linear(784, 512, bias=True)
linear2 = torch.nn.Linear(512, 512, bias=True)
linear3 = torch.nn.Linear(512, 512, bias=True)
linear4 = torch.nn.Linear(512, 512, bias=True)
linear5 = torch.nn.Linear(512, 10, bias=True)

relu = torch.nn.ReLU()
# p is the probability of being dropped in PyTorch
dropout = torch.nn.Dropout(p=1 - keep_prob)

# xavier initializer
torch.nn.init.xavier_uniform(linear1.weight)
torch.nn.init.xavier_uniform(linear2.weight)
torch.nn.init.xavier_uniform(linear3.weight)
torch.nn.init.xavier_uniform(linear4.weight)
torch.nn.init.xavier_uniform(linear5.weight)
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-10-5-mnist_nn_dropout.py

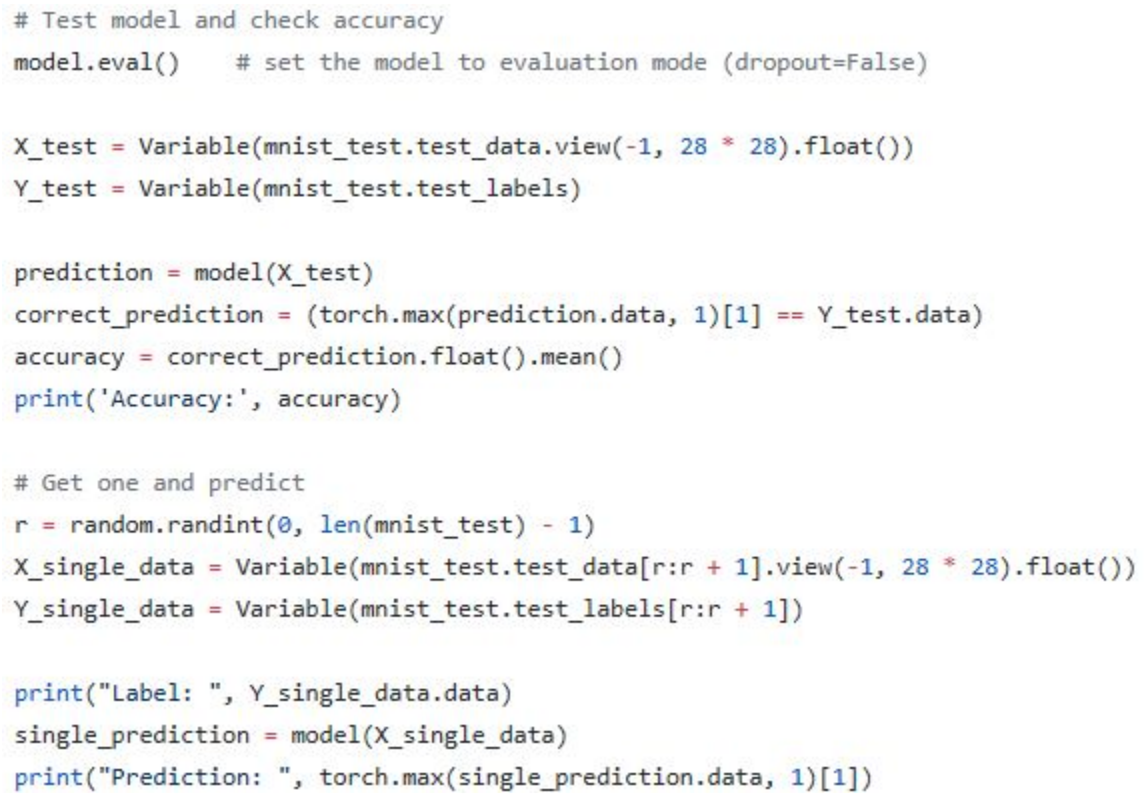
Deep NN with Dropout for MNIST



```
# model
model = torch.nn.Sequential(linear1, relu, dropout,
                            linear2, relu, dropout,
                            linear3, relu, dropout,
                            linear4, relu, dropout,
                            linear5)

# define cost/loss & optimizer
criterion = torch.nn.CrossEntropyLoss()    # Softmax is internally computed.
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)

# train my model
for epoch in range(training_epochs):
    avg_cost = 0
    total_batch = len(mnist_train) // batch_size

    for i, (batch_xs, batch_ys) in enumerate(data_loader):
        # reshape input image into [batch_size by 784]
        X = Variable(batch_xs.view(-1, 28 * 28))
        Y = Variable(batch_ys)    # label is not one-hot encoded

        optimizer.zero_grad()
        hypothesis = model(X)
        cost = criterion(hypothesis, Y)
        cost.backward()
        optimizer.step()

        avg_cost += cost / total_batch

    print("[Epoch: {:>4}] cost = {:>.9}".format(epoch + 1, avg_cost.data[0]))

print('Learning Finished!')
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-10-5-mnist_nn_dropout.py

Deep NN with Dropout for MNIST



```
# Test model and check accuracy
model.eval()    # set the model to evaluation mode (dropout=False)

X_test = Variable(mnist_test.test_data.view(-1, 28 * 28).float())
Y_test = Variable(mnist_test.test_labels)

prediction = model(X_test)
correct_prediction = (torch.max(prediction.data, 1)[1] == Y_test.data)
accuracy = correct_prediction.float().mean()
print('Accuracy:', accuracy)

# Get one and predict
r = random.randint(0, len(mnist_test) - 1)
X_single_data = Variable(mnist_test.test_data[r:r + 1].view(-1, 28 * 28).float())
Y_single_data = Variable(mnist_test.test_labels[r:r + 1])

print("Label: ", Y_single_data.data)
single_prediction = model(X_single_data)
print("Prediction: ", torch.max(single_prediction.data, 1)[1])
```

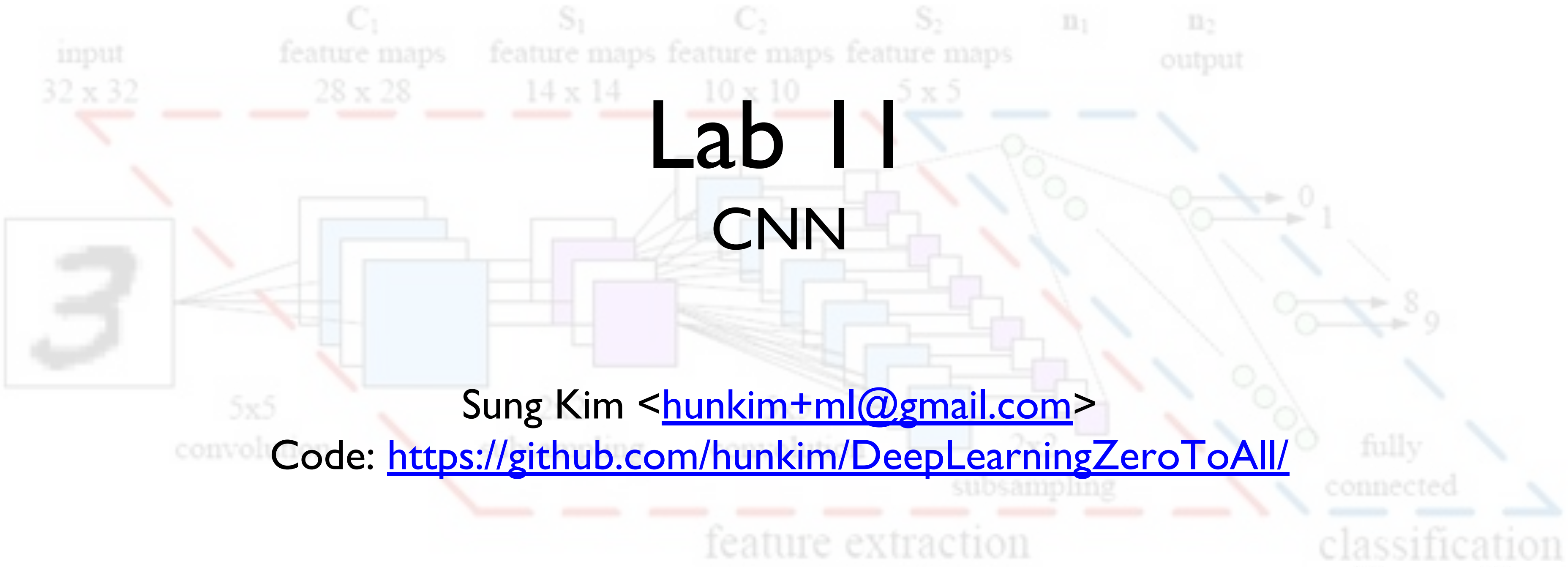
https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-10-5-mnist_nn_dropout.py

Summary

- Softmax VS Neural Nets for MNIST, 90% and 94.5%
- Xavier initialization: 97.8%
- Deep Neural Nets with Dropout: **98%**

With PyTorch 0.2.0_2

PYTORCH



Lab 11

CNN

Sung Kim <hunkim+ml@gmail.com>

Code: https://github.com/hunkim/DeepLearningZeroToAll/