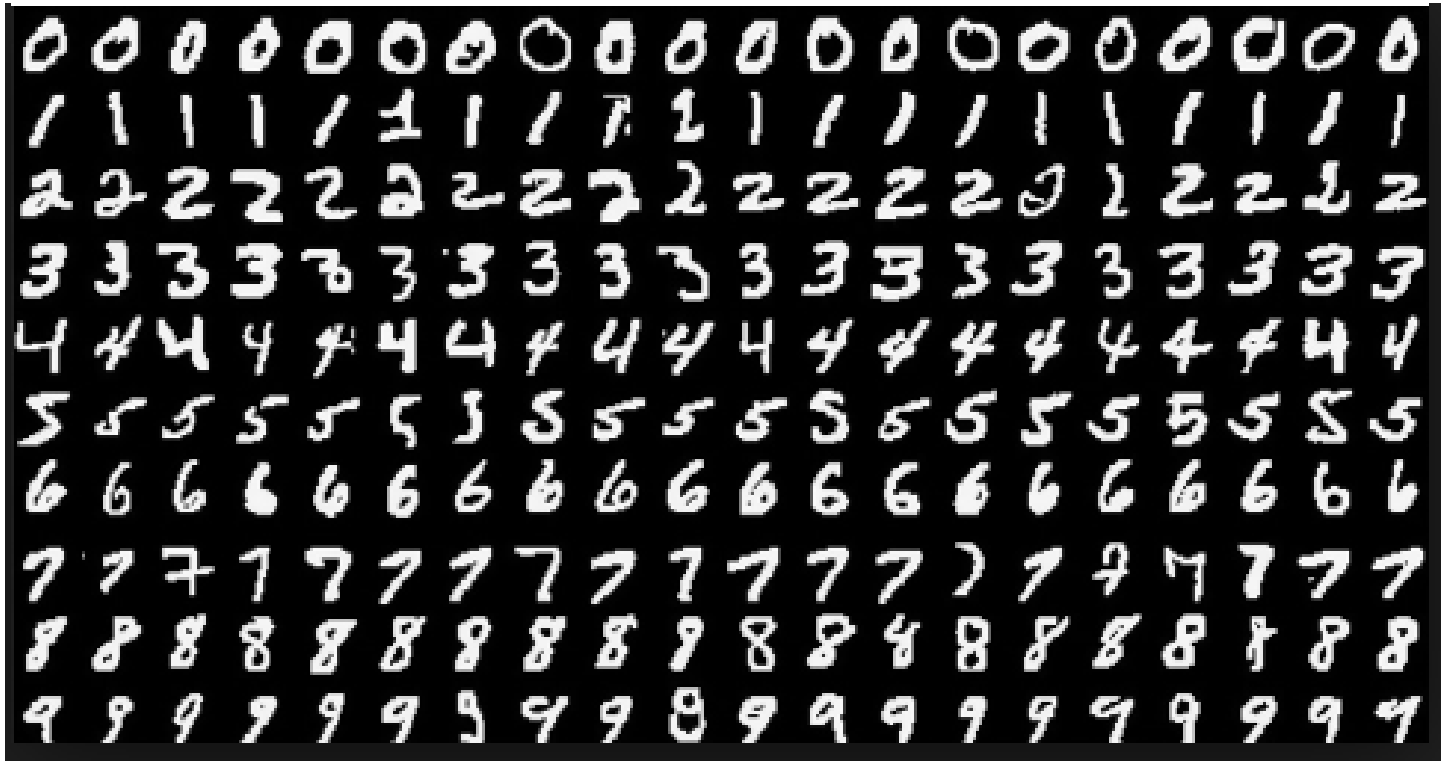
Kaggle Digit Recognizer

sample_submission	2018/4/26 下午 0...	Microsoft Excel ...	236 KB
test	2018/4/26 下午 0...	Microsoft Excel ...	49,921 KB
train	2018/4/26 下午 0...	Microsoft Excel ...	74,976 KB



Dataset Details

- Goal
  - Recognition handwritten image of digit
- Task
  - image recognition(10 classes of digits)
- Training / Testing
  - around 42k / 28k
- Reference
  - https://www.kaggle.com/c/digit-recognizer/data

Train.csv

- Each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels in total.
- This pixel-value is an integer between 0 and 255, inclusive.

label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	pixel10	pixel11	pixel12	pix
1	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	
7	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	0	0	0	
5	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	0	0	0	
8	0	0	0	0	0	0	0	0	0	0	0	0	0	
9	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	

Model Architecture

use “LeNet” model

```
class LeNet(nn.Module):
  def __init__(self):
    super(LeNet, self).__init__()
    self.conv1 = nn.Conv2d(1, 6, (5,5), padding=2)
    self.conv2 = nn.Conv2d(6, 16, (5,5))
    self.fc1   = nn.Linear(16*5*5, 120)
    self.fc2   = nn.Linear(120, 84)
    self.fc3   = nn.Linear(84, 10)
  def forward(self, x):
    x = F.max_pool2d(F.relu(self.conv1(x)), (2,2))
    x = F.max_pool2d(F.relu(self.conv2(x)), (2,2))
    x = x.view(-1, self.num_flat_features(x))
    x = F.relu(self.fc1(x))
    x = F.relu(self.fc2(x))
    x = self.fc3(x)
    return x
  def num_flat_features(self, x):
    size = x.size()[1:]
    num_features = 1
    for s in size:
      num_features *= s
    return num_features

net = LeNet()
#print (net)

use_gpu = torch.cuda.is_available()
if use_gpu:
  net = net.cuda()
  print ('USE GPU')
else:
  print ('USE CPU')

criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(net.parameters(), lr=0.001)
```

```
#print ("1. Loading data")
train = pd.read_csv('/content/drive/My Drive/train.csv').values
train = shuffle(train)
test  = pd.read_csv('/content/drive/My Drive/test.csv').values

#print ("2. Converting data")
X_data  = train[:, 1:].reshape(train.shape[0], 1, 28, 28)
X_data  = X_data.astype(float)
X_data /= 255.0
X_data  = torch.from_numpy(X_data);
X_label = train[:,0];
X_label = X_label.astype(int);
X_label = torch.from_numpy(X_label);
X_label = X_label.view(train.shape[0],-1);
#print (X_data.size(), X_label.size())

#print ("3. Training phase")
nb_train = train.shape[0]
nb_epoch = 30000
nb_index = 0
nb_batch = 16

for epoch in range(nb_epoch):
  if nb_index + nb_batch >= nb_train:
    nb_index = 0
  else:
    nb_index = nb_index + nb_batch

  mini_data  = Variable(X_data[nb_index:(nb_index+nb_batch)].clone())
  mini_label = Variable(X_label[nb_index:(nb_index+nb_batch)].clone(), requires_grad = False)
  mini_data  = mini_data.type(torch.FloatTensor)
  mini_label = mini_label.type(torch.LongTensor)
  if use_gpu:
    mini_data  = mini_data.cuda()
    mini_label = mini_label.cuda()
  optimizer.zero_grad()
  mini_out   = net(mini_data)
  mini_label = mini_label.view(nb_batch)
  mini_loss  = criterion(mini_out, mini_label)
  mini_loss.backward()
  optimizer.step()

  if (epoch + 1) % 10 == 0:
    print("Epoch = %d, Loss = %f" %(epoch+1, mini_loss.data))
```

```
#print ("4. Testing phase")

Y_data  = test.reshape(test.shape[0], 1, 28, 28)
Y_data  = Y_data.astype(float)
Y_data /= 255.0
Y_data  = torch.from_numpy(Y_data);
#print (Y_data.size())
nb_test = test.shape[0]

net.eval()

final_prediction = np.ndarray(shape = (nb_test, 2), dtype=int)
for each_sample in range(nb_test):
  sample_data = Variable(Y_data[each_sample:each_sample+1].clone())
  sample_data = sample_data.type(torch.FloatTensor)
  if use_gpu:
    sample_data = sample_data.cuda()
  sample_out = net(sample_data)
  _, pred = torch.max(sample_out, 1)
  final_prediction[each_sample][0] = 1 + each_sample
  final_prediction[each_sample][1] = pred.data
  if (each_sample + 1) % 1000 == 0:
    print("Total tested = %d" %(each_sample + 1))

#print ('5. Generating submission file')

submission = pd.DataFrame(final_prediction, dtype=int, columns=['ImageId', 'Label'])
submission.to_csv('/content/drive/My Drive/pytorch_LeNet.csv', index=False, header=True)

# end
```

Assignment #2 – CNN Modeling

- Requirement
  1. Setup programming environment
     - Pytorch、Python、Cuda……anything you need for deep learning modeling.
  2. Given the datatset and the CNN sample code, run training and testing. Report your final testing accuracy
  3. For the given CNN model, add one additional CNN layer and one fully connected layer. Compare the testing accuracy of the modified model to the original model.

Assignment #2 – CNN Modeling

- You need to hand in your source code and report
- The report should cover:
  - Method description – what are your reference codes? How to run your test?
  - Experimental results
    - The comparison of classification accuracy from validation
    - The comparison of ranking from Kaggle for each feature
  - Discussion
  - Problem and difficulties
- Deadline : 11:59 pm, 5/06(Mon).
- File format – zip all your files into a single file: studentID_hw1_version, ex: 602410143_hw1_v1

Assignment Rules

- **Late policy**
  - You will get 20% deduction of your scores per day.
  - It means if the assignment is delayed one day for 80%, two days for 60%,…, five days for 0% .
- **No-copy policy**
  - Copying is strictly forbidden in our class.
  - Once the assignment is confirmed by TA as COPY, the score will be 0%.