SML-PROJECT3.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share S

+ Code + Text

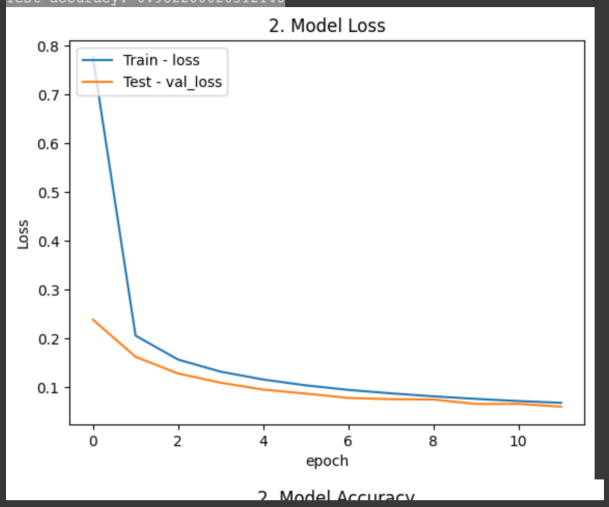
RAM Disk

```python
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K

batch_size = 128
num_classes = 10
epochs = 12

# input image dimensions
img_rows, img_cols = 28, 28

# the data, split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

if K.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
    x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
    input_shape = (1, img_rows, img_cols)
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

model = Sequential()
model.add(Conv2D(6, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(16, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(120, activation='relu'))
model.add(Dense(84, activation='relu'))

model.add(Dense(num_classes, activation='softmax'))

# https://keras.io/optimizers/
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.legacy.Adadelta(learning_rate=0.1, rho=0.95, decay=0.0),
              metrics=['accuracy'])

model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])

```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 0s 0us/step
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
Epoch 1/12
469/469 [==============================] - 16s 33ms/step - loss: 0.7592 - accuracy: 0.7829 - val_loss: 0.2414 - val_accuracy: 0.9266
Epoch 2/12
469/469 [==============================] - 12s 25ms/step - loss: 0.2020 - accuracy: 0.9398 - val_loss: 0.1526 - val_accuracy: 0.9562
Epoch 3/12
469/469 [==============================] - 12s 26ms/step - loss: 0.1517 - accuracy: 0.9550 - val_loss: 0.1213 - val_accuracy: 0.9650
Epoch 4/12
469/469 [==============================] - 12s 26ms/step - loss: 0.1259 - accuracy: 0.9627 - val_loss: 0.1041 - val_accuracy: 0.9710
Epoch 5/12
469/469 [==============================] - 12s 25ms/step - loss: 0.1090 - accuracy: 0.9679 - val_loss: 0.0908 - val_accuracy: 0.9734
Epoch 6/12
469/469 [==============================] - 12s 25ms/step - loss: 0.0967 - accuracy: 0.9713 - val_loss: 0.0839 - val_accuracy: 0.9745
Epoch 7/12
469/469 [==============================] - 12s 26ms/step - loss: 0.0875 - accuracy: 0.9737 - val_loss: 0.0795 - val_accuracy: 0.9748
Epoch 8/12
469/469 [==============================] - 13s 27ms/step - loss: 0.0801 - accuracy: 0.9758 - val_loss: 0.0709 - val_accuracy: 0.9799
Epoch 9/12
469/469 [==============================] - 12s 25ms/step - loss: 0.0742 - accuracy: 0.9777 - val_loss: 0.0666 - val_accuracy: 0.9799
Epoch 10/12
469/469 [==============================] - 12s 25ms/step - loss: 0.0693 - accuracy: 0.9794 - val_loss: 0.0688 - val_accuracy: 0.9772
Epoch 11/12
469/469 [==============================] - 12s 25ms/step - loss: 0.0653 - accuracy: 0.9804 - val_loss: 0.0587 - val_accuracy: 0.9820
Epoch 12/12
469/469 [==============================] - 12s 25ms/step - loss: 0.0614 - accuracy: 0.9815 - val_loss: 0.0575 - val_accuracy: 0.9831
Test loss: 0.05749049037694931
Test accuracy: 0.9830999970436096
```

```python
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K

batch_size = 128
num_classes = 10
epochs = 12

# input image dimensions
img_rows, img_cols = 28, 28

# the data, split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

if K.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
    x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
    input_shape = (1, img_rows, img_cols)
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

model = Sequential()
model.add(Conv2D(6, kernel_size=(5, 5),
                 activation='relu',
                 input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(16, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(120, activation='relu'))
model.add(Dense(84, activation='relu'))

model.add(Dense(num_classes, activation='softmax'))

# https://keras.io/optimizers/
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.legacy.Adadelta(learning_rate=0.1, rho=0.95, decay=0.0),
              metrics=['accuracy'])

trained_model=model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])

# Plots #
#print(trained_model.item)

from matplotlib import pyplot as plt
train = []
test = []

for item in trained_model.history['loss']:
  train.append(item)
for item in trained_model.history['val_loss']:
  test.append(item)

plt.plot(train)
plt.plot(test)
plt.title('2. Model Loss')
plt.ylabel('Loss')
plt.xlabel('epoch')
plt.legend(['Train - loss', 'Test - val_loss'], loc='upper left')
plt.show()

train = []
test = []

for item in trained_model.history['accuracy']:
  train.append(item)
for item in trained_model.history['val_accuracy']:
  test.append(item)

plt.plot(train)
plt.plot(test)
plt.title('2. Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('epoch')
plt.legend(['Train - accuracy', 'Test - val_accuracy'], loc='upper left')
plt.show()
```

```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
Epoch 1/12
469/469 [==============================] - 16s 33ms/step - loss: 0.7751 - accuracy: 0.7701 - val_loss: 0.2384 - val_accuracy: 0.9309
Epoch 2/12
469/469 [==============================] - 15s 31ms/step - loss: 0.2057 - accuracy: 0.9391 - val_loss: 0.1622 - val_accuracy: 0.9537
Epoch 3/12
469/469 [==============================] - 16s 34ms/step - loss: 0.1565 - accuracy: 0.9534 - val_loss: 0.1280 - val_accuracy: 0.9620
Epoch 4/12
469/469 [==============================] - 15s 31ms/step - loss: 0.1318 - accuracy: 0.9605 - val_loss: 0.1091 - val_accuracy: 0.9668
Epoch 5/12
469/469 [==============================] - 15s 33ms/step - loss: 0.1156 - accuracy: 0.9653 - val_loss: 0.0950 - val_accuracy: 0.9722
Epoch 6/12
469/469 [==============================] - 16s 33ms/step - loss: 0.1037 - accuracy: 0.9690 - val_loss: 0.0869 - val_accuracy: 0.9740
Epoch 7/12
469/469 [==============================] - 15s 32ms/step - loss: 0.0944 - accuracy: 0.9712 - val_loss: 0.0779 - val_accuracy: 0.9768
Epoch 8/12
469/469 [==============================] - 15s 32ms/step - loss: 0.0873 - accuracy: 0.9734 - val_loss: 0.0753 - val_accuracy: 0.9760
Epoch 9/12
469/469 [==============================] - 16s 33ms/step - loss: 0.0812 - accuracy: 0.9748 - val_loss: 0.0747 - val_accuracy: 0.9761
Epoch 10/12
469/469 [==============================] - 16s 35ms/step - loss: 0.0759 - accuracy: 0.9761 - val_loss: 0.0655 - val_accuracy: 0.9801
Epoch 11/12
469/469 [==============================] - 15s 33ms/step - loss: 0.0715 - accuracy: 0.9777 - val_loss: 0.0656 - val_accuracy: 0.9802
Epoch 12/12
469/469 [==============================] - 15s 33ms/step - loss: 0.0678 - accuracy: 0.9793 - val_loss: 0.0600 - val_accuracy: 0.9822
Test loss: 0.05997486412525177
Test accuracy: 0.982200026512146
```

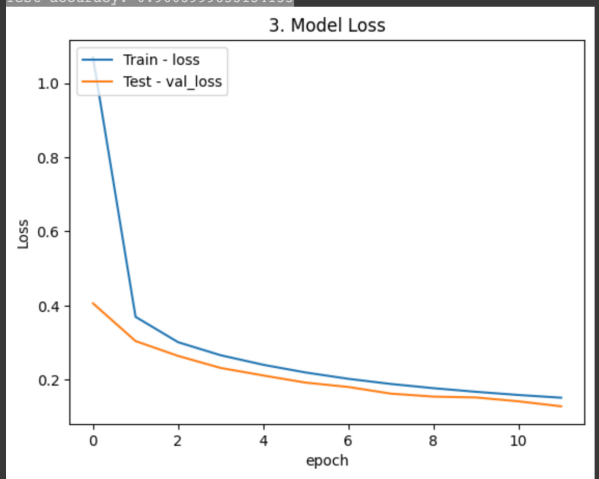


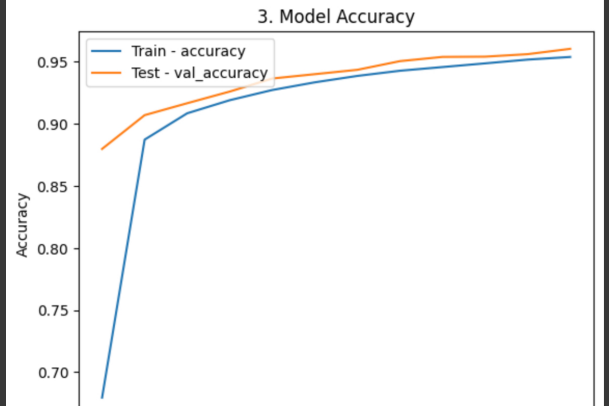
2m

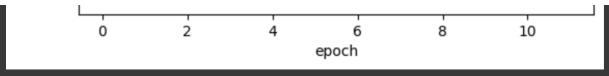
```python
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K

batch_size = 128
num_classes = 10
epochs = 12

# input image dimensions
img_rows, img_cols = 28, 28

# the data, split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

if K.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
    x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
    input_shape = (1, img_rows, img_cols)
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

model = Sequential()
model.add(Conv2D(6, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(3, 3)))
model.add(Conv2D(6, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(120, activation='relu'))
model.add(Dense(84, activation='relu'))
```

```python
model.add(Dense(num_classes, activation='softmax'))

# https://keras.io/optimizers/
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.legacy.Adadelta(learning_rate=0.1, rho=0.95, decay=0.0),
              metrics=['accuracy'])

trained_model=model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])

# Plots #
#print(trained_model.item)

from matplotlib import pyplot as plt
train = []
test = []

for item in trained_model.history['loss']:
  train.append(item)
for item in trained_model.history['val_loss']:
  test.append(item)

plt.plot(train)
plt.plot(test)
plt.title('3. Model Loss')
plt.ylabel('Loss')
plt.xlabel('epoch')
plt.legend(['Train - loss', 'Test - val_loss'], loc='upper left')
plt.show()

train = []
test = []

for item in trained_model.history['accuracy']:
  train.append(item)
for item in trained_model.history['val_accuracy']:
  test.append(item)

plt.plot(train)
plt.plot(test)
plt.title('3. Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('epoch')
plt.legend(['Train - accuracy', 'Test - val_accuracy'], loc='upper left')
plt.show()
```

```
Epoch 12/12
469/469 [==============================] - 12s 25ms/step - loss: 0.1514 - accuracy: 0.9538 - val_loss: 0.1282 - val_accuracy: 0.9604
Test loss: 0.12820255756378174
Test accuracy: 0.9603999853134155
```







CNN_train_test

October 2, 2023

1 Step 1: Import Libraries for this Project

[76]:
```
### Only Numpy Library is necessary for this project and all the layers are␣
↪compiled with Numpy.
import numpy as np
import pickle
import sys
import time
import pdb          ### library for debugging
np.random.seed(1000)
```

2 Step 2: Convolutional Layer Definition

[77]:
```
class Convolution2D:
    # Initialization of convolutional layer
    def __init__(self, inputs_channel, num_filters, kernel_size, padding,␣
↪stride, learning_rate, name):
        # weight size: (F, C, K, K)
        # bias size: (F)
        self.F = num_filters
        self.K = kernel_size
        self.C = inputs_channel

        self.weights = np.zeros((self.F, self.C, self.K, self.K))
        self.bias = np.zeros((self.F, 1))
        for i in range(0,self.F):
            self.weights[i,:,:,:] = np.random.normal(loc=0, scale=np.sqrt(1./
↪(self.C*self.K*self.K)), size=(self.C, self.K, self.K))

        self.p = padding
        self.s = stride
        self.lr = learning_rate
        self.name = name

    # Padding Layer
```

```
    def zero_padding(self, inputs, size):
        w, h = inputs.shape[0], inputs.shape[1]
        new_w = 2 * size + w
        new_h = 2 * size + h
        out = np.zeros((new_w, new_h))
        out[size:w+size, size:h+size] = inputs
        return out

    # Forward propagation
    def forward(self, inputs):
        # input size: (C, W, H)
        # output size: (N, F ,WW, HH)
        C = inputs.shape[0]
        W = inputs.shape[1]+2*self.p
        H = inputs.shape[2]+2*self.p
        self.inputs = np.zeros((C, W, H))
        for c in range(inputs.shape[0]):
            self.inputs[c,:,:] = self.zero_padding(inputs[c,:,:], self.p)
        WW = (W - self.K)//self.s + 1
        HH = (H - self.K)//self.s + 1
        feature_maps = np.zeros((self.F, WW, HH))
        for f in range(self.F):
            for w in range(WW):
                for h in range(HH):
                    feature_maps[f,w,h]=np.sum(self.inputs[:,w:w+self.K,h:
↪h+self.K]*self.weights[f,:,:,:])+self.bias[f]

        return feature_maps

    # Backward Propagation
    def backward(self, dy):

        C, W, H = self.inputs.shape
        dx = np.zeros(self.inputs.shape)
        dw = np.zeros(self.weights.shape)
        db = np.zeros(self.bias.shape)

        F, W, H = dy.shape
        for f in range(F):
            for w in range(W):
                for h in range(H):
                    dw[f,:,:,:]+=dy[f,w,h]*self.inputs[:,w:w+self.K,h:h+self.K]
                    dx[:,w:w+self.K,h:h+self.K]+=dy[f,w,h]*self.weights[f,:,:,:]

        for f in range(F):
            db[f] = np.sum(dy[f, :, :])
```

```
        self.weights -= self.lr * dw
        self.bias -= self.lr * db
        return dx

    # Function for extract the weights and bias for storage
    def extract(self):
        return {self.name+'.weights':self.weights, self.name+'.bias':self.bias}

    # Feed the pretrained weights and bias for models
    def feed(self, weights, bias):
        self.weights = weights
        self.bias = bias
```

3 Step 3: MaxPooling Layer Definition

[78]:
```
class Maxpooling2D:
    # Initialization of MaxPooling layer
    def __init__(self, pool_size, stride, name):
        self.pool = pool_size
        self.s = stride
        self.name = name

    # Forward propagation
    def forward(self, inputs):
        self.inputs = inputs
        C, W, H = inputs.shape
        new_width = (W - self.pool)//self.s + 1
        new_height = (H - self.pool)//self.s + 1
        out = np.zeros((C, new_width, new_height))
        for c in range(C):
            for w in range(W//self.s):
                for h in range(H//self.s):
                    out[c, w, h] = np.max(self.inputs[c, w*self.s:w*self.s+self.
↪pool, h*self.s:h*self.s+self.pool])
        return out

    # Backward propagation
    def backward(self, dy):
        C, W, H = self.inputs.shape
        dx = np.zeros(self.inputs.shape)

        for c in range(C):
            for w in range(0, W, self.pool):
                for h in range(0, H, self.pool):
                    st = np.argmax(self.inputs[c,w:w+self.pool,h:h+self.pool])
```

```
                    (idx, idy) = np.unravel_index(st, (self.pool, self.pool))
                    dx[c, w+idx, h+idy] = dy[c, w//self.pool, h//self.pool]
        return dx

    # No weights and bias for pooling layer to store
    def extract(self):
        return
```

4 Step 4: Fully-Connected Layer Definition

[79]:
```
class FullyConnected:
    # Initialization of Fully-Connected Layer
    def __init__(self, num_inputs, num_outputs, learning_rate, name):
        self.weights = 0.01*np.random.rand(num_inputs, num_outputs)
        self.bias = np.zeros((num_outputs, 1))
        self.lr = learning_rate
        self.name = name

    # Forward Propagation
    def forward(self, inputs):
        self.inputs = inputs
        return np.dot(self.inputs, self.weights) + self.bias.T

    # Backward Propagation
    def backward(self, dy):

        if dy.shape[0] == self.inputs.shape[0]:
            dy = dy.T
        dw = dy.dot(self.inputs)
        db = np.sum(dy, axis=1, keepdims=True)
        dx = np.dot(dy.T, self.weights.T)

        self.weights -= self.lr * dw.T
        self.bias -= self.lr * db

        return dx

    # Extract weights and bias for storage
    def extract(self):
        return {self.name+'.weights':self.weights, self.name+'.bias':self.bias}

    # Feed the pretrained weights and bias for models
    def feed(self, weights, bias):
        self.weights = weights
        self.bias = bias
```

```
### Flatten function to convert 4D feature maps into 3D feature vectors
class Flatten:
    def __init__(self):
        pass
    def forward(self, inputs):
        self.C, self.W, self.H = inputs.shape
        return inputs.reshape(1, self.C*self.W*self.H)
    def backward(self, dy):
        return dy.reshape(self.C, self.W, self.H)
    def extract(self):
        return
```

5 Step 5: Activation Function Definition

[80]:
```
### ReLU activation function
class ReLu:
    def __init__(self):
        pass
    def forward(self, inputs):
        self.inputs = inputs
        ret = inputs.copy()
        ret[ret < 0] = 0
        return ret
    def backward(self, dy):
        dx = dy.copy()
        dx[self.inputs < 0] = 0
        return dx
    def extract(self):
        return

### Softmax activation function
class Softmax:
    def __init__(self):
        pass
    def forward(self, inputs):
        exp = np.exp(inputs, dtype=np.float)
        self.out = exp/np.sum(exp)
        return self.out
    def backward(self, dy):
        return self.out.T - dy.reshape(dy.shape[0],1)
    def extract(self):
        return
```

6 Step 6: Loss Function Definition

[81]:
```
### Cross-Entropy Loss function
def cross_entropy(inputs, labels):
    out_num = labels.shape[0]
    p = np.sum(labels.reshape(1,out_num)*inputs)
    loss = -np.log(p)
    return loss
```

7 Step 7: Neural Network Definition

[82]:
```
"""
This step shows how to define a simple CNN with all kind of layers which we␣
 ↪introduced above.
"""
class Net:
    def __init__(self):
        # input: 28x28
        # output: 1x4 (only a subset, containing 4 classes, of the MNIST will␣
 ↪be used)
        # conv1:  {(28-5+0x0)/2+1} -> (12x12x6) (output size of convolutional␣
 ↪layer)
        # maxpool2: {(12-2)/2+1} -> (6x6)x6 (output size of pooling layer)
        # fc3: 216 -> 32
        # fc4: 32 -> 4
        # softmax: 4 -> 4
        lr = 0.001
        self.layers = []
        self.layers.append(Convolution2D(inputs_channel=1, num_filters=6,␣
 ↪kernel_size=5, padding=0, stride=2, learning_rate=lr, name='conv1'))
        self.layers.append(ReLu())
        self.layers.append(Maxpooling2D(pool_size=2, stride=2, name='maxpool2'))
        self.layers.append(Flatten())
        self.layers.append(FullyConnected(num_inputs=6*6*6, num_outputs=32,␣
 ↪learning_rate=lr, name='fc3'))
        self.layers.append(ReLu())
        self.layers.append(FullyConnected(num_inputs=32, num_outputs=4,␣
 ↪learning_rate=lr, name='fc4'))
        self.layers.append(Softmax())
        self.lay_num = len(self.layers)

    ### Function for train the network
    def train(self, data, label):
        batch_size = data.shape[0]
```

```
        loss = 0
        acc = 0
        for b in range(batch_size):
            x = data[b]
            y = label[b]
            # forward pass
            for l in range(self.lay_num):
                output = self.layers[l].forward(x)
                x = output
            loss += cross_entropy(output, y)
            if np.argmax(output) == np.argmax(y):
                acc += 1
            # backward pass
            dy = y
            for l in range(self.lay_num-1, -1, -1):
                dout = self.layers[l].backward(dy)
                dy = dout
        return loss, acc
```

8 Step 8: Create your own subset samples of MNIST

[83]:
```
from precode import *
"""
The subset of MNIST is created based on the last 4 digits of your ASUID. There␣
 ↪are 4 categories and all returned
samples are preprocessed and shuffled.
"""
print('Loading data......')
sub_train_images, sub_train_labels, sub_test_images, sub_test_labels =␣
 ↪init_subset('2289') # input your ASUID last 4 digits here to generate the␣
 ↪subset samples of MNIST for training and testing
```

```
Loading data…
Preparing data…
```

9 Step 9: Initial Network and do the Training and Testing process

[84]:
```
net = Net()
epoch = 10              ### Default number of epochs
batch_size = 100        ### Default batch size
num_batch = sub_train_images.shape[0]/batch_size

test_size = sub_test_images.shape[0]      # Obtain the size of testing samples
```

```
train_size = sub_train_images.shape[0]    # Obtain the size of training samples

### ---------- ###
"""
Please compile your own evaluation code based on the training code
to evaluate the trained network.
The function name and the inputs of the function have been predifined and␣
 ↪please finish the remaining part.
"""
def evaluate(net, images, labels):
    acc = 0
    loss = 0
    batch_size = 1

    pass
    for batch_index in range(0, images.shape[0], batch_size):
        # Extract the data and label, and as we have batch size as 1,␣
 ↪extraction is not looped.
        data=images[batch_index]
        label=labels[batch_index]
        # Forward pass
        for l in range(net.lay_num):
                output = net.layers[l].forward(data)
                data = output
        # Calculate the loss based on the given cross entropy function.
        loss += cross_entropy(output, label)
        # Calculate accuracy
        if np.argmax(output) == np.argmax(label):
            acc += 1
    # Calculate the average loss and accuracy
    loss /= (float(images.shape[0]) / float(batch_size))
    acc /= (float(images.shape[0]) / float(batch_size))

    return acc, loss

### Start training process
for e in range(epoch):
    total_acc = 0
    total_loss = 0
    print('Epoch %d' % e)
    for batch_index in range(0, sub_train_images.shape[0], batch_size):
        # batch input
        if batch_index + batch_size < sub_train_images.shape[0]:
            data = sub_train_images[batch_index:batch_index+batch_size]
            label = sub_train_labels[batch_index:batch_index + batch_size]
        else:
            data = sub_train_images[batch_index:sub_train_images.shape[0]]
```

```
            label = sub_train_labels[batch_index:sub_train_labels.shape[0]]
        # Compute the remaining time
        start_time = time.time()
        batch_loss,batch_acc = net.train(data, label)  # Train the network with␣
↪samples in one batch

        end_time = time.time()
        batch_time = end_time-start_time
        remain_time = (sub_train_images.shape[0]-batch_index)/
↪batch_size*batch_time
        hrs = int(remain_time/3600)
        mins = int((remain_time/60-hrs*60))
        secs = int(remain_time-mins*60-hrs*3600)
        print('=== Iter:{0:d} === Remain: {1:d} Hrs {2:d} Mins {3:d} Secs ==='.
↪format(int(batch_index+batch_size),int(hrs),int(mins),int(secs)))

    # Print out the Performance
    train_acc, train_loss = evaluate(net, sub_train_images, sub_train_labels) ␣
↪# Use the evaluation code to obtain the training accuracy and loss
    test_acc, test_loss = evaluate(net, sub_test_images, sub_test_labels)     ␣
↪# Use the evaluation code to obtain the testing accuracy and loss
    print('=== Epoch:{0:d} Train Size:{1:d}, Train Acc:{2:.3f}, Train Loss:{3:.
↪3f} ===='.format(e, train_size,train_acc,train_loss))

    print('=== Epoch:{0:d} Test Size:{1:d}, Test Acc:{2:.3f}, Test Loss:{3:.3f}␣
↪==='.format(e, test_size, test_acc,test_loss))
```

```
Epoch 0
=== Iter:100 === Remain: 0 Hrs 1 Mins 24 Secs ===
=== Iter:200 === Remain: 0 Hrs 1 Mins 19 Secs ===
=== Iter:300 === Remain: 0 Hrs 1 Mins 15 Secs ===
=== Iter:400 === Remain: 0 Hrs 1 Mins 11 Secs ===
=== Iter:500 === Remain: 0 Hrs 1 Mins 6 Secs ===
=== Iter:600 === Remain: 0 Hrs 1 Mins 1 Secs ===
=== Iter:700 === Remain: 0 Hrs 0 Mins 59 Secs ===
=== Iter:800 === Remain: 0 Hrs 0 Mins 53 Secs ===
=== Iter:900 === Remain: 0 Hrs 0 Mins 50 Secs ===
=== Iter:1000 === Remain: 0 Hrs 0 Mins 46 Secs ===
=== Iter:1100 === Remain: 0 Hrs 0 Mins 42 Secs ===
=== Iter:1200 === Remain: 0 Hrs 0 Mins 37 Secs ===
=== Iter:1300 === Remain: 0 Hrs 0 Mins 33 Secs ===
=== Iter:1400 === Remain: 0 Hrs 0 Mins 29 Secs ===
=== Iter:1500 === Remain: 0 Hrs 0 Mins 24 Secs ===
=== Iter:1600 === Remain: 0 Hrs 0 Mins 20 Secs ===
=== Iter:1700 === Remain: 0 Hrs 0 Mins 16 Secs ===
=== Iter:1800 === Remain: 0 Hrs 0 Mins 12 Secs ===
=== Iter:1900 === Remain: 0 Hrs 0 Mins 8 Secs ===
```

```
=== Iter:2000 === Remain: 0 Hrs 0 Mins 4 Secs ===
[[[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 …


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]
```

```
 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]
[[0. 1. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 …
 [0. 0. 1. 0.]
 [1. 0. 0. 0.]
 [0. 0. 1. 0.]]
[[[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 …


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
```

```
   …
    [6 6 6 … 6 6 6]
    [6 6 6 … 6 6 6]
    [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
    [6 6 6 … 6 6 6]
    [6 6 6 … 6 6 6]
    …
    [6 6 6 … 6 6 6]
    [6 6 6 … 6 6 6]
    [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
    [6 6 6 … 6 6 6]
    [6 6 6 … 6 6 6]
    …
    [6 6 6 … 6 6 6]
    [6 6 6 … 6 6 6]
    [6 6 6 … 6 6 6]]]]
[[0. 0. 1. 0.]
 [1. 0. 0. 0.]
 [0. 0. 1. 0.]
 …
 [0. 1. 0. 0.]
 [0. 0. 0. 1.]
 [1. 0. 0. 0.]]
=== Epoch:0 Train Size:2000, Train Acc:0.490, Train Loss:1.267 ===
=== Epoch:0 Test Size:400, Test Acc:0.443, Test Loss:1.261 ===
Epoch 1
=== Iter:100 === Remain: 0 Hrs 1 Mins 26 Secs ===
=== Iter:200 === Remain: 0 Hrs 1 Mins 13 Secs ===
=== Iter:300 === Remain: 0 Hrs 1 Mins 10 Secs ===
=== Iter:400 === Remain: 0 Hrs 1 Mins 6 Secs ===
=== Iter:500 === Remain: 0 Hrs 1 Mins 1 Secs ===
=== Iter:600 === Remain: 0 Hrs 0 Mins 58 Secs ===
=== Iter:700 === Remain: 0 Hrs 0 Mins 54 Secs ===
=== Iter:800 === Remain: 0 Hrs 0 Mins 50 Secs ===
=== Iter:900 === Remain: 0 Hrs 0 Mins 45 Secs ===
=== Iter:1000 === Remain: 0 Hrs 0 Mins 42 Secs ===
=== Iter:1100 === Remain: 0 Hrs 0 Mins 39 Secs ===
=== Iter:1200 === Remain: 0 Hrs 0 Mins 35 Secs ===
=== Iter:1300 === Remain: 0 Hrs 0 Mins 32 Secs ===
=== Iter:1400 === Remain: 0 Hrs 0 Mins 27 Secs ===
=== Iter:1500 === Remain: 0 Hrs 0 Mins 22 Secs ===
=== Iter:1600 === Remain: 0 Hrs 0 Mins 20 Secs ===
```

```
=== Iter:1700 === Remain: 0 Hrs 0 Mins 15 Secs ===
=== Iter:1800 === Remain: 0 Hrs 0 Mins 11 Secs ===
=== Iter:1900 === Remain: 0 Hrs 0 Mins 7 Secs ===
=== Iter:2000 === Remain: 0 Hrs 0 Mins 4 Secs ===
[[[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 …


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
```

```
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]]
[[0. 1. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 …
 [0. 0. 1. 0.]
 [1. 0. 0. 0.]
 [0. 0. 1. 0.]]
[[[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 …
```

```
 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]]
[[0. 0. 1. 0.]
 [1. 0. 0. 0.]
 [0. 0. 1. 0.]
 …
 [0. 1. 0. 0.]
 [0. 0. 0. 1.]
 [1. 0. 0. 0.]]
=== Epoch:1 Train Size:2000, Train Acc:0.647, Train Loss:0.940 ===
=== Epoch:1 Test Size:400, Test Acc:0.613, Test Loss:0.934 ===
Epoch 2
=== Iter:100 === Remain: 0 Hrs 1 Mins 18 Secs ===
=== Iter:200 === Remain: 0 Hrs 1 Mins 13 Secs ===
=== Iter:300 === Remain: 0 Hrs 1 Mins 9 Secs ===
=== Iter:400 === Remain: 0 Hrs 1 Mins 5 Secs ===
=== Iter:500 === Remain: 0 Hrs 1 Mins 2 Secs ===
=== Iter:600 === Remain: 0 Hrs 0 Mins 57 Secs ===
=== Iter:700 === Remain: 0 Hrs 0 Mins 54 Secs ===
=== Iter:800 === Remain: 0 Hrs 0 Mins 50 Secs ===
=== Iter:900 === Remain: 0 Hrs 0 Mins 46 Secs ===
=== Iter:1000 === Remain: 0 Hrs 0 Mins 42 Secs ===
=== Iter:1100 === Remain: 0 Hrs 0 Mins 38 Secs ===
=== Iter:1200 === Remain: 0 Hrs 0 Mins 35 Secs ===
=== Iter:1300 === Remain: 0 Hrs 0 Mins 31 Secs ===
```

```
=== Iter:1400 === Remain: 0 Hrs 0 Mins 27 Secs ===
=== Iter:1500 === Remain: 0 Hrs 0 Mins 23 Secs ===
=== Iter:1600 === Remain: 0 Hrs 0 Mins 19 Secs ===
=== Iter:1700 === Remain: 0 Hrs 0 Mins 15 Secs ===
=== Iter:1800 === Remain: 0 Hrs 0 Mins 11 Secs ===
=== Iter:1900 === Remain: 0 Hrs 0 Mins 7 Secs ===
=== Iter:2000 === Remain: 0 Hrs 0 Mins 3 Secs ===
[[[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 …


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
```

```
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]]
[[0. 1. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 …
 [0. 0. 1. 0.]
 [1. 0. 0. 0.]
 [0. 0. 1. 0.]]
[[[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]
```

```
 …


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]]
[[0. 0. 1. 0.]
 [1. 0. 0. 0.]
 [0. 0. 1. 0.]
 …
 [0. 1. 0. 0.]
 [0. 0. 0. 1.]
 [1. 0. 0. 0.]]
=== Epoch:2 Train Size:2000, Train Acc:0.535, Train Loss:1.149 ===
=== Epoch:2 Test Size:400, Test Acc:0.535, Test Loss:1.208 ===
Epoch 3
=== Iter:100 === Remain: 0 Hrs 1 Mins 17 Secs ===
=== Iter:200 === Remain: 0 Hrs 1 Mins 13 Secs ===
=== Iter:300 === Remain: 0 Hrs 1 Mins 10 Secs ===
=== Iter:400 === Remain: 0 Hrs 1 Mins 6 Secs ===
=== Iter:500 === Remain: 0 Hrs 1 Mins 1 Secs ===
=== Iter:600 === Remain: 0 Hrs 0 Mins 58 Secs ===
=== Iter:700 === Remain: 0 Hrs 0 Mins 57 Secs ===
=== Iter:800 === Remain: 0 Hrs 0 Mins 50 Secs ===
=== Iter:900 === Remain: 0 Hrs 0 Mins 46 Secs ===
=== Iter:1000 === Remain: 0 Hrs 0 Mins 42 Secs ===
```

```
=== Iter:1100 === Remain: 0 Hrs 0 Mins 38 Secs ===
=== Iter:1200 === Remain: 0 Hrs 0 Mins 34 Secs ===
=== Iter:1300 === Remain: 0 Hrs 0 Mins 31 Secs ===
=== Iter:1400 === Remain: 0 Hrs 0 Mins 27 Secs ===
=== Iter:1500 === Remain: 0 Hrs 0 Mins 23 Secs ===
=== Iter:1600 === Remain: 0 Hrs 0 Mins 19 Secs ===
=== Iter:1700 === Remain: 0 Hrs 0 Mins 15 Secs ===
=== Iter:1800 === Remain: 0 Hrs 0 Mins 11 Secs ===
=== Iter:1900 === Remain: 0 Hrs 0 Mins 8 Secs ===
=== Iter:2000 === Remain: 0 Hrs 0 Mins 3 Secs ===
[[[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 …


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]
```

```
 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]]
[[0. 1. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 …
 [0. 0. 1. 0.]
 [1. 0. 0. 0.]
 [0. 0. 1. 0.]]
[[[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
```

```
   [6 6 6 … 6 6 6]]]


 …


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]]
[[0. 0. 1. 0.]
 [1. 0. 0. 0.]
 [0. 0. 1. 0.]
 …
 [0. 1. 0. 0.]
 [0. 0. 0. 1.]
 [1. 0. 0. 0.]]
=== Epoch:3 Train Size:2000, Train Acc:0.788, Train Loss:0.575 ===
=== Epoch:3 Test Size:400, Test Acc:0.807, Test Loss:0.587 ===
Epoch 4
=== Iter:100 === Remain: 0 Hrs 1 Mins 17 Secs ===
=== Iter:200 === Remain: 0 Hrs 1 Mins 14 Secs ===
=== Iter:300 === Remain: 0 Hrs 1 Mins 9 Secs ===
=== Iter:400 === Remain: 0 Hrs 1 Mins 5 Secs ===
=== Iter:500 === Remain: 0 Hrs 1 Mins 2 Secs ===
=== Iter:600 === Remain: 0 Hrs 0 Mins 58 Secs ===
=== Iter:700 === Remain: 0 Hrs 0 Mins 54 Secs ===
```

```
=== Iter:800 === Remain: 0 Hrs 0 Mins 50 Secs ===
=== Iter:900 === Remain: 0 Hrs 0 Mins 46 Secs ===
=== Iter:1000 === Remain: 0 Hrs 0 Mins 43 Secs ===
=== Iter:1100 === Remain: 0 Hrs 0 Mins 38 Secs ===
=== Iter:1200 === Remain: 0 Hrs 0 Mins 34 Secs ===
=== Iter:1300 === Remain: 0 Hrs 0 Mins 31 Secs ===
=== Iter:1400 === Remain: 0 Hrs 0 Mins 27 Secs ===
=== Iter:1500 === Remain: 0 Hrs 0 Mins 22 Secs ===
=== Iter:1600 === Remain: 0 Hrs 0 Mins 19 Secs ===
=== Iter:1700 === Remain: 0 Hrs 0 Mins 15 Secs ===
=== Iter:1800 === Remain: 0 Hrs 0 Mins 11 Secs ===
=== Iter:1900 === Remain: 0 Hrs 0 Mins 7 Secs ===
=== Iter:2000 === Remain: 0 Hrs 0 Mins 3 Secs ===
[[[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 …


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
```

```
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]]
[[0. 1. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 …
 [0. 0. 1. 0.]
 [1. 0. 0. 0.]
 [0. 0. 1. 0.]]
[[[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
```

```
  …
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]]]


 …


 [[[6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  …
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  …
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  …
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]]]]
[[0. 0. 1. 0.]
 [1. 0. 0. 0.]
 [0. 0. 1. 0.]
 …
 [0. 1. 0. 0.]
 [0. 0. 0. 1.]
 [1. 0. 0. 0.]]
=== Epoch:4 Train Size:2000, Train Acc:0.807, Train Loss:0.532 ===
=== Epoch:4 Test Size:400, Test Acc:0.787, Test Loss:0.572 ===
Epoch 5
=== Iter:100 === Remain: 0 Hrs 1 Mins 16 Secs ===
=== Iter:200 === Remain: 0 Hrs 1 Mins 13 Secs ===
=== Iter:300 === Remain: 0 Hrs 1 Mins 10 Secs ===
=== Iter:400 === Remain: 0 Hrs 1 Mins 5 Secs ===
```

```
=== Iter:500 === Remain: 0 Hrs 1 Mins 2 Secs ===
=== Iter:600 === Remain: 0 Hrs 0 Mins 57 Secs ===
=== Iter:700 === Remain: 0 Hrs 0 Mins 54 Secs ===
=== Iter:800 === Remain: 0 Hrs 0 Mins 50 Secs ===
=== Iter:900 === Remain: 0 Hrs 0 Mins 46 Secs ===
=== Iter:1000 === Remain: 0 Hrs 0 Mins 42 Secs ===
=== Iter:1100 === Remain: 0 Hrs 0 Mins 38 Secs ===
=== Iter:1200 === Remain: 0 Hrs 0 Mins 34 Secs ===
=== Iter:1300 === Remain: 0 Hrs 0 Mins 31 Secs ===
=== Iter:1400 === Remain: 0 Hrs 0 Mins 27 Secs ===
=== Iter:1500 === Remain: 0 Hrs 0 Mins 23 Secs ===
=== Iter:1600 === Remain: 0 Hrs 0 Mins 20 Secs ===
=== Iter:1700 === Remain: 0 Hrs 0 Mins 15 Secs ===
=== Iter:1800 === Remain: 0 Hrs 0 Mins 11 Secs ===
=== Iter:1900 === Remain: 0 Hrs 0 Mins 7 Secs ===
=== Iter:2000 === Remain: 0 Hrs 0 Mins 3 Secs ===
[[[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 …


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
```

```
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]]
[[0. 1. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 …
 [0. 0. 1. 0.]
 [1. 0. 0. 0.]
 [0. 0. 1. 0.]]
[[[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]
```

```
 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 …


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]]
[[0. 0. 1. 0.]
 [1. 0. 0. 0.]
 [0. 0. 1. 0.]
 …
 [0. 1. 0. 0.]
 [0. 0. 0. 1.]
 [1. 0. 0. 0.]]
=== Epoch:5 Train Size:2000, Train Acc:0.818, Train Loss:0.496 ===
=== Epoch:5 Test Size:400, Test Acc:0.792, Test Loss:0.552 ===
Epoch 6
=== Iter:100 === Remain: 0 Hrs 1 Mins 19 Secs ===
```

```
=== Iter:200 === Remain: 0 Hrs 1 Mins 13 Secs ===
=== Iter:300 === Remain: 0 Hrs 1 Mins 8 Secs ===
=== Iter:400 === Remain: 0 Hrs 1 Mins 6 Secs ===
=== Iter:500 === Remain: 0 Hrs 1 Mins 3 Secs ===
=== Iter:600 === Remain: 0 Hrs 0 Mins 58 Secs ===
=== Iter:700 === Remain: 0 Hrs 0 Mins 54 Secs ===
=== Iter:800 === Remain: 0 Hrs 0 Mins 49 Secs ===
=== Iter:900 === Remain: 0 Hrs 0 Mins 47 Secs ===
=== Iter:1000 === Remain: 0 Hrs 0 Mins 42 Secs ===
=== Iter:1100 === Remain: 0 Hrs 0 Mins 38 Secs ===
=== Iter:1200 === Remain: 0 Hrs 0 Mins 35 Secs ===
=== Iter:1300 === Remain: 0 Hrs 0 Mins 31 Secs ===
=== Iter:1400 === Remain: 0 Hrs 0 Mins 26 Secs ===
=== Iter:1500 === Remain: 0 Hrs 0 Mins 23 Secs ===
=== Iter:1600 === Remain: 0 Hrs 0 Mins 19 Secs ===
=== Iter:1700 === Remain: 0 Hrs 0 Mins 15 Secs ===
=== Iter:1800 === Remain: 0 Hrs 0 Mins 11 Secs ===
=== Iter:1900 === Remain: 0 Hrs 0 Mins 7 Secs ===
=== Iter:2000 === Remain: 0 Hrs 0 Mins 3 Secs ===
[[[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 …
```

```
 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]]
[[0. 1. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 …
 [0. 0. 1. 0.]
 [1. 0. 0. 0.]
 [0. 0. 1. 0.]]
[[[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
```

```
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 …


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]]
[[0. 0. 1. 0.]
 [1. 0. 0. 0.]
 [0. 0. 1. 0.]
 …
 [0. 1. 0. 0.]
 [0. 0. 0. 1.]
 [1. 0. 0. 0.]]
=== Epoch:6 Train Size:2000, Train Acc:0.828, Train Loss:0.470 ===
```

```
=== Epoch:6 Test Size:400, Test Acc:0.797, Test Loss:0.534 ===
Epoch 7
=== Iter:100 === Remain: 0 Hrs 1 Mins 17 Secs ===
=== Iter:200 === Remain: 0 Hrs 1 Mins 13 Secs ===
=== Iter:300 === Remain: 0 Hrs 1 Mins 8 Secs ===
=== Iter:400 === Remain: 0 Hrs 1 Mins 6 Secs ===
=== Iter:500 === Remain: 0 Hrs 1 Mins 2 Secs ===
=== Iter:600 === Remain: 0 Hrs 0 Mins 57 Secs ===
=== Iter:700 === Remain: 0 Hrs 0 Mins 54 Secs ===
=== Iter:800 === Remain: 0 Hrs 0 Mins 50 Secs ===
=== Iter:900 === Remain: 0 Hrs 0 Mins 46 Secs ===
=== Iter:1000 === Remain: 0 Hrs 0 Mins 42 Secs ===
=== Iter:1100 === Remain: 0 Hrs 0 Mins 38 Secs ===
=== Iter:1200 === Remain: 0 Hrs 0 Mins 35 Secs ===
=== Iter:1300 === Remain: 0 Hrs 0 Mins 31 Secs ===
=== Iter:1400 === Remain: 0 Hrs 0 Mins 27 Secs ===
=== Iter:1500 === Remain: 0 Hrs 0 Mins 23 Secs ===
=== Iter:1600 === Remain: 0 Hrs 0 Mins 19 Secs ===
=== Iter:1700 === Remain: 0 Hrs 0 Mins 15 Secs ===
=== Iter:1800 === Remain: 0 Hrs 0 Mins 11 Secs ===
=== Iter:1900 === Remain: 0 Hrs 0 Mins 7 Secs ===
=== Iter:2000 === Remain: 0 Hrs 0 Mins 3 Secs ===
[[[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]
```

```
 …

 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]

 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]

 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]]
[[0. 1. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 …
 [0. 0. 1. 0.]
 [1. 0. 0. 0.]
 [0. 0. 1. 0.]]
[[[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]

 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
```

```
  …
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  …
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]]]


 …


 [[[6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  …
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  …
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  …
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]]]]
[[0. 0. 1. 0.]
 [1. 0. 0. 0.]
 [0. 0. 1. 0.]
 …
 [0. 1. 0. 0.]
```

```
 [0. 0. 0. 1.]
 [1. 0. 0. 0.]]
=== Epoch:7 Train Size:2000, Train Acc:0.834, Train Loss:0.450 ===
=== Epoch:7 Test Size:400, Test Acc:0.802, Test Loss:0.527 ===
Epoch 8
=== Iter:100 === Remain: 0 Hrs 1 Mins 17 Secs ===
=== Iter:200 === Remain: 0 Hrs 1 Mins 13 Secs ===
=== Iter:300 === Remain: 0 Hrs 1 Mins 10 Secs ===
=== Iter:400 === Remain: 0 Hrs 1 Mins 4 Secs ===
=== Iter:500 === Remain: 0 Hrs 1 Mins 1 Secs ===
=== Iter:600 === Remain: 0 Hrs 0 Mins 58 Secs ===
=== Iter:700 === Remain: 0 Hrs 0 Mins 55 Secs ===
=== Iter:800 === Remain: 0 Hrs 0 Mins 50 Secs ===
=== Iter:900 === Remain: 0 Hrs 0 Mins 45 Secs ===
=== Iter:1000 === Remain: 0 Hrs 0 Mins 43 Secs ===
=== Iter:1100 === Remain: 0 Hrs 0 Mins 39 Secs ===
=== Iter:1200 === Remain: 0 Hrs 0 Mins 35 Secs ===
=== Iter:1300 === Remain: 0 Hrs 0 Mins 30 Secs ===
=== Iter:1400 === Remain: 0 Hrs 0 Mins 27 Secs ===
=== Iter:1500 === Remain: 0 Hrs 0 Mins 23 Secs ===
=== Iter:1600 === Remain: 0 Hrs 0 Mins 19 Secs ===
=== Iter:1700 === Remain: 0 Hrs 0 Mins 15 Secs ===
=== Iter:1800 === Remain: 0 Hrs 0 Mins 11 Secs ===
=== Iter:1900 === Remain: 0 Hrs 0 Mins 8 Secs ===
=== Iter:2000 === Remain: 0 Hrs 0 Mins 4 Secs ===
[[[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
```

```
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 …


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]]
[[0. 1. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 …
 [0. 0. 1. 0.]
 [1. 0. 0. 0.]
 [0. 0. 1. 0.]]
[[[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]
```

```
 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 …


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]]
[[0. 0. 1. 0.]
 [1. 0. 0. 0.]
```

```
 [0. 0. 1. 0.]
 …
 [0. 1. 0. 0.]
 [0. 0. 0. 1.]
 [1. 0. 0. 0.]]
=== Epoch:8 Train Size:2000, Train Acc:0.843, Train Loss:0.434 ===
=== Epoch:8 Test Size:400, Test Acc:0.800, Test Loss:0.528 ===
Epoch 9
=== Iter:100 === Remain: 0 Hrs 1 Mins 33 Secs ===
=== Iter:200 === Remain: 0 Hrs 1 Mins 14 Secs ===
=== Iter:300 === Remain: 0 Hrs 1 Mins 10 Secs ===
=== Iter:400 === Remain: 0 Hrs 1 Mins 11 Secs ===
=== Iter:500 === Remain: 0 Hrs 1 Mins 2 Secs ===
=== Iter:600 === Remain: 0 Hrs 0 Mins 59 Secs ===
=== Iter:700 === Remain: 0 Hrs 0 Mins 54 Secs ===
=== Iter:800 === Remain: 0 Hrs 0 Mins 54 Secs ===
=== Iter:900 === Remain: 0 Hrs 0 Mins 46 Secs ===
=== Iter:1000 === Remain: 0 Hrs 0 Mins 42 Secs ===
=== Iter:1100 === Remain: 0 Hrs 0 Mins 39 Secs ===
=== Iter:1200 === Remain: 0 Hrs 0 Mins 36 Secs ===
=== Iter:1300 === Remain: 0 Hrs 0 Mins 32 Secs ===
=== Iter:1400 === Remain: 0 Hrs 0 Mins 27 Secs ===
=== Iter:1500 === Remain: 0 Hrs 0 Mins 24 Secs ===
=== Iter:1600 === Remain: 0 Hrs 0 Mins 20 Secs ===
=== Iter:1700 === Remain: 0 Hrs 0 Mins 17 Secs ===
=== Iter:1800 === Remain: 0 Hrs 0 Mins 12 Secs ===
=== Iter:1900 === Remain: 0 Hrs 0 Mins 7 Secs ===
=== Iter:2000 === Remain: 0 Hrs 0 Mins 3 Secs ===
[[[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   …
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
   [6 6 6 … 6 6 6]
```

```
  [6 6 6 … 6 6 6]
  …
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]]]


 …


 [[[6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  …
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  …
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]]]


 [[[6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  …
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]]]]
[[0. 1. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 …
 [0. 0. 1. 0.]
 [1. 0. 0. 0.]
 [0. 0. 1. 0.]]
[[[[6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  …
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
```

```
  [6 6 6 … 6 6 6]]]


[[[6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  …
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]]]


[[[6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  …
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]]]


…


[[[6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  …
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]]]


[[[6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  …
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]]]


[[[6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
  …
  [6 6 6 … 6 6 6]
  [6 6 6 … 6 6 6]
```

```
   [6 6 6 … 6 6 6]]]]
[[0. 0. 1. 0.]
 [1. 0. 0. 0.]
 [0. 0. 1. 0.]
 …
 [0. 1. 0. 0.]
 [0. 0. 0. 1.]
 [1. 0. 0. 0.]]
=== Epoch:9 Train Size:2000, Train Acc:0.850, Train Loss:0.420 ===
=== Epoch:9 Test Size:400, Test Acc:0.807, Test Loss:0.530 ===
```