ECE472 - Deep Learning: Assignment 3

Authors
Alexander Koldy

September 22 2021
ECE472 - Fall 2021

Notes

There are two functions containing two separate neural networks to predict MNIST data. The first one, *proven_case* has an "overly-safe" convolutional neural network that has greater than a 95.5% prediction percentage on testing data. The second function, *less_parameters_case* should not be considered, as I ran out of time to properly implement anything. I would have liked to attempt the following to reduce the number of parameters: only selecting a small chunk of the image (perhaps a 3x3 square in the middle) and using that as an input. Additionally, I would have attempted to "predict" based off of 1 output, rather than 10, such that the output was a float in the range of zero to nine, and the prediction was just the closest integer to that decimal. This would prevent the need for having a 10 output system that automatically has at least 10 weights. Please note, this is just something I thought about, but not in great detail - it could be completely wrong.

```
'''
Alexander Koldy
ECE 472 - Deep Learning
Assignment 3: Classify MNIST digits with a
(optionally convoultional) neural network.
Get at least95.5% accuracy on the test test.
'''

import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
import gzip

class Data():
  def __init__(self):
    '''Training, validation, and testing data parameters'''
    self.image_shape = (28, 28)
    self.num_training = 50000
    self.num_validation = 10000
    self.num_testing = 10000

  def load_data(self, training_data_path, training_label_path, testing_data_path, testing_label_path):
    training_data = gzip.open(training_data_path, 'r')
    training_labels = gzip.open(training_label_path, 'r')
    testing_data = gzip.open(testing_data_path, 'r')
    testing_labels = gzip.open(testing_label_path, 'r')
    training_data.read(16)
    training_labels.read(8)
    testing_data.read(16)
    testing_labels.read(8)

    '''Establish training and validation data'''
    training_data = np.frombuffer(training_data.read(self.image_shape[0]*self.image_shape[1]*(self.num_training + self.num_testing)), dtype=np.uint8).astype(np.float64)
    training_data = training_data.reshape(self.num_training + self.num_validation, self.image_shape[0], self.image_shape[1], 1)
    new_order = np.random.permutation(self.num_training + self.num_validation)
    training_data = training_data[new_order, :, :, :]
    self.validation_data = training_data[50000:, :, :, :]
    self.training_data = training_data[:50000, :, :, :]


    '''Establish training and validation labels'''
    training_labels = np.frombuffer(training_labels.read(self.num_training + self.num_validation), dtype=np.uint8).astype(np.float64)
    training_labels = training_labels[new_order]
    self.validation_labels = training_labels[50000:]
    self.training_labels = training_labels[:50000]

    '''Establish testing data'''
    testing_data = np.frombuffer(testing_data.read(self.image_shape[0]*self.image_shape[1]*self.num_testing), dtype=np.uint8).astype(np.float64)
    testing_data = testing_data.reshape(self.num_testing, self.image_shape[0], self.image_shape[1], 1)
    self.testing_data = testing_data

```

```
    '''Establish testing labels'''
    testing_labels = np.frombuffer(testing_labels.read(self.num_testing), dtype=np.uint8).astype(np.float64)
    self.testing_labels = testing_labels

    #print(self.training_labels[107])
    #image = np.asarray(self.training_data[107]).squeeze()
    #plt.imshow(image)
    #plt.show()

'''
Network that gets >= 95.5% accuracy not test case
'''
def proven_case():
  '''Create data'''
  training_data_path = 'data/train-images-idx3-ubyte.gz'
  training_label_path = 'data/train-labels-idx1-ubyte.gz'
  testing_data_path = 'data/t10k-images-idx3-ubyte.gz'
  testing_label_path = 'data/t10k-labels-idx1-ubyte.gz'

  data = Data()
  data.load_data(training_data_path, training_label_path, testing_data_path, testing_label_path)

  '''Establish Convolutional Neural Network'''
  model = tf.keras.models.Sequential()
  model.add(tf.keras.layers.Conv2D(filters=28, kernel_size=(3, 3), activation='relu', input_shape=(28, 28, 1), kernel_regularizer=tf.keras.regularizers.l2(0.01)))
  model.add(tf.keras.layers.MaxPooling2D((2, 2)))
  model.add(tf.keras.layers.Conv2D(filters=56, kernel_size=(3, 3), activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.01)))
  model.add(tf.keras.layers.MaxPooling2D((2, 2)))
  model.add(tf.keras.layers.Conv2D(filters=56, kernel_size=(3, 3), activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.01)))
  model.add(tf.keras.layers.Dropout(0.2))
  model.add(tf.keras.layers.Flatten())
  model.add(tf.keras.layers.Dense(56, activation='relu'))
  model.add(tf.keras.layers.Dense(10, activation='softmax'))

  '''Training/Validation'''
  model.compile(optimizer='Adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

  print("Training/Validation Data!")
  history = model.fit(data.training_data, data.training_labels, epochs=3, validation_data=(data.validation_data, data.validation_labels))

  '''Testing'''
  print("Testing Data!")
  history = model.evaluate(data.testing_data, data.testing_labels)

  '''Check summary'''
  model.summary()

'''
Attempt at using less parameters. NOT WORKING right now, since its not technically part
of the assignment (just a bonus), there is no reason to look at this
'''
```

```
def less_parameters_case():
  '''Create data'''
  training_data_path = 'data/train-images-idx3-ubyte.gz'
  training_label_path = 'data/train-labels-idx1-ubyte.gz'
  testing_data_path = 'data/t10k-images-idx3-ubyte.gz'
  testing_label_path = 'data/t10k-labels-idx1-ubyte.gz'

  data = Data()
  data.load_data(training_data_path, training_label_path, testing_data_path, testing_label_path)

  '''Establish Convolutional Neural Network'''
  model = tf.keras.models.Sequential()
  model.add(tf.keras.layers.Dense(1, activation='softmax'))

  '''Training/Validation'''
  model.compile(optimizer='Adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

  history = model.fit(np.reshape(data.training_data, (data.num_training, 28*28)).mean(1), data.training_labels, epochs=1, validation_data=(np.reshape(data.validation_data, (data.num_validation, 28*28)).mean(1), data.validation_labels))

  '''Testing'''
  history = model.evaluate(np.reshape(data.testing_data, (data.num_testing, 28*28)).mean(1), data.testing_labels)

  '''Check summary'''
  model.summary()

proven_case()

'''
References:
(1) Cooper Union ECE-472: Deep Learning - Learning Materials
(2) https://stackoverflow.com/questions/40427435/extract-images-from-idx3-ubyte-file-or-gzip-via-python
(3) https://www.tensorflow.org/tutorials/images/cnn
'''
```

```
Epoch 1/3
1563/1563 [==============================] - 16s 10ms/step - loss: 0.9506 - accuracy: 0.9060 - val_loss: 0.3912 - val_accuracy: 0.9796
Epoch 2/3
1563/1563 [==============================] - 15s 10ms/step - loss: 0.2997 - accuracy: 0.9705 - val_loss: 0.2013 - val_accuracy: 0.9773
Epoch 3/3
1563/1563 [==============================] - 15s 10ms/step - loss: 0.1747 - accuracy: 0.9764 - val_loss: 0.1325 - val_accuracy: 0.9822
Testing Data!
313/313 [==============================] - 1s 3ms/step - loss: 0.1260 - accuracy: 0.9829
```