ECE-471 Selected Topics in Machine Learning
Prof. Curro
Assignment 3

Evan Bubniak

September 26, 2019

1 Results

The CNN specified below achieves 96.28% accuracy on the training set, 96.41% accuracy on the validation set (a random selection of 10000 samples from the training set), and 96.81% accuracy on the test set. The MNIST data is loaded by extracting a ZIP archive hosted on Kaggle containing `.png` files of each sample, organized by label and set.

2 Code

```
import numpy as np
import tensorflow as tf
import tensorflow.keras as keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D
from sklearn.model_selection import train_test_split

import os, os.path
import zipfile, tarfile
import numpy as np
from matplotlib.image import imread

RANDOM_SEED = 31415
BATCH_SIZE = 128
NUM_EPOCH = 6
DROPOUT_RATIO = 0.1
L2_PENALTY = 0.01

tf.random.set_seed(RANDOM_SEED)

class Data:
        def __init__(self, data_path):
                self.data_path = data_path

        def build_dataset(self, dataset_kind):
                X = []
                y = []
                self.labels = os.listdir(os.path.join(self.data_path, dataset_kind))
                self.num_labels = len(self.labels)
                for label in self.labels:
                        img_path = os.path.join(self.data_path, dataset_kind, label)
                        for img_file in os.listdir(img_path):
                                img_filepath = os.path.join(img_path, img_file)
                                img = self.get_nparray_from_imgfile(img_filepath)
                                X.append(img)
                                y.append(label)
```

```
                X, y = np.asarray(X), np.asarray(y)
                self.X_dim = X.shape[1]
                X = self.format_img(X)

                return X, self.format_labels(y)

        def get_nparray_from_imgfile(self, img_path):
                img = imread(img_path)
                return img

        def format_img(self, X):
                return X.reshape(X.shape[0], self.X_dim, self.X_dim, 1)

        def format_labels(self, y):
                return keras.utils.to_categorical(y, self.num_labels)

# Data source: https://www.kaggle.com/jidhumohan/mnist-png/downloads/mnist-png.zip
# extracted from zip via the following function
        def fetch_data(zip_location, data_destination):
                with zipfile.ZipFile(zip_location, 'r') as zip_ref:
                        zip_ref.extractall(data_destination)
                tar = tarfile.open(os.path.join(data_destination, "MNIST_png.tar"))
                tar.extractall(path=data_destination)
                tar.close()

class Model:
        def __init__(self):

                self.sequential_model = Sequential([
                Conv2D(data.X_dim, (3, 3),
                        activation='relu',
                        input_shape = (data.X_dim, data.X_dim, 1)
                ),
                MaxPooling2D(pool_size=(2, 2)),
                Dropout(DROPOUT_RATIO),
                Flatten(),
                Dense(data.num_labels,
```

```
                        activation='softmax',
                        kernel_regularizer=keras.regularizers.l2(L2_PENALTY))
                ])

                self.sequential_model.compile(loss=keras.losses.categorical_crossentropy,
                optimizer=keras.optimizers.Adamax(),
                metrics=['accuracy'])

        def train(self, X_train, y_train, X_val, y_val):
                self.model_log = self.sequential_model.fit(X_train, y_train,
                batch_size=BATCH_SIZE,
                epochs=NUM_EPOCH,
                verbose=1,
                validation_data=(X_val, y_val)
                )
                return self.model_log

        def test(self, X_test, y_test):
                self.sequential_model.evaluate(X_test, y_test, verbose = 1)


data = Data(os.path.join(os.getcwd(), "dataset", "mnist_png"))

X_training_set, y_training_set = data.build_dataset("training")
X_test, y_test = data.build_dataset("testing")
X_train, X_val, y_train, y_val = train_test_split(X_training_set, y_training_set,
        test_size=1/6,
        random_state=RANDOM_SEED)

model = Model()
model_log = model.train(X_train, y_train,X_val, y_val)
model.test(X_test, y_test)
```