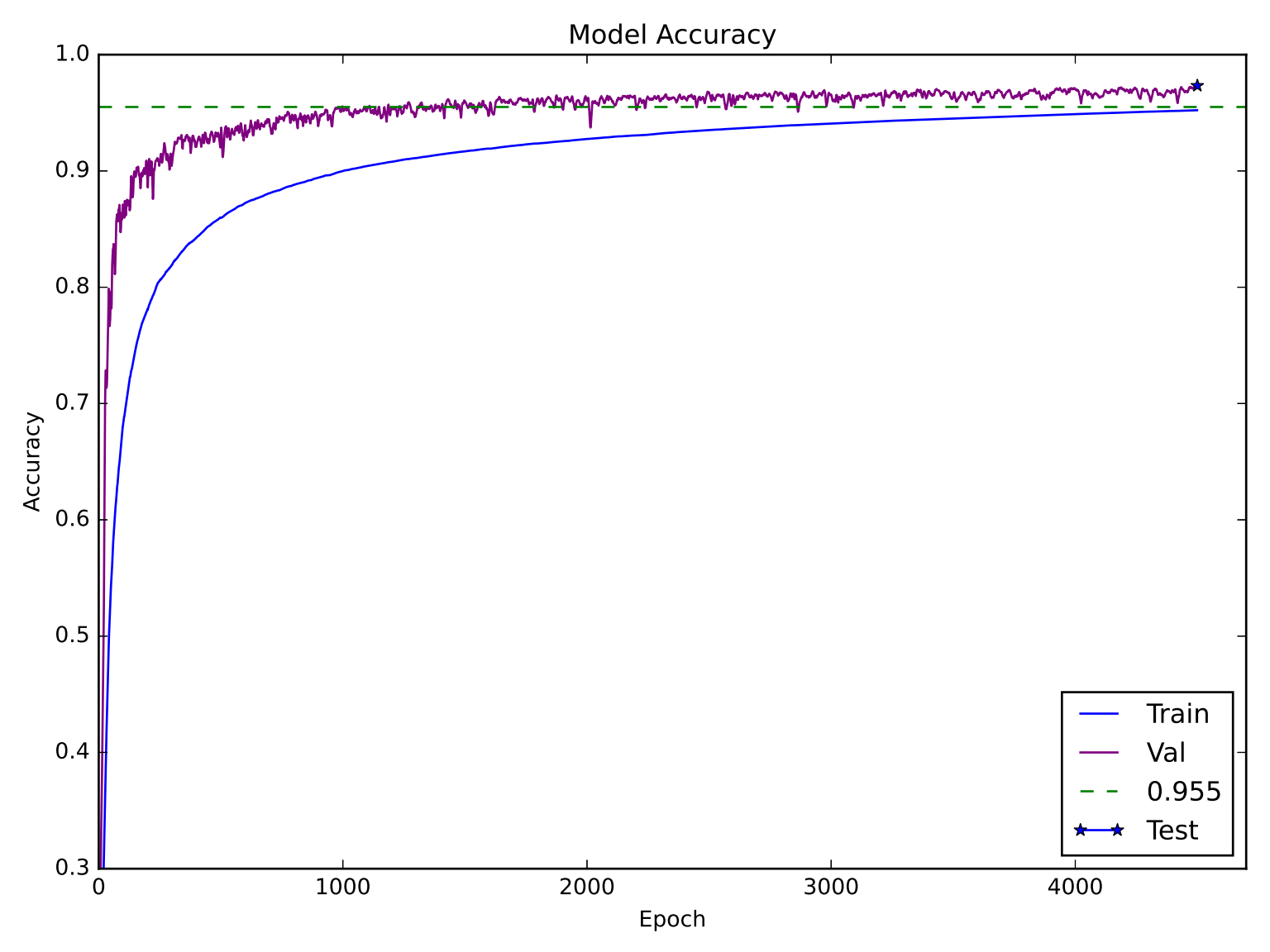
```
#!/bin/env python3.8

"""
Rosemary Cho
Based on Example assignment. Author: Chris Curro
"""
import os
import logging
import matplotlib
import numpy as np
import pandas as pd
from numpy import pi
import matplotlib.pyplot as plt
from pdb import set_trace
import tensorflow as tf
from absl import app
from absl import flags
from tqdm import trange
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from keras.regularizers import l2
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
import keras
import csv

from dataclasses import dataclass, field, InitVar

script_path = os.path.dirname(os.path.realpath(__file__))


@dataclass
class Data:
    def __post_init__(self):
        """Dataset source: https://www.kaggle.com/datasets/oddrationale/mnist-in-csv?select=mnist_test.csv""
"
        # 60,000 images
        df_train = pd.read_csv("mnist_train.csv")
        # 10,000 images
        df_test = pd.read_csv("mnist_test.csv")

        X = df_train[df_train.columns[1:]]
        X_test = df_test[df_test.columns[1:]]
        y = df_train["label"]
        y_test = df_test["label"]

        X_train, X_val, y_train, y_val = train_test_split(
            X, y, test_size=0.2, random_state=1
        )

        scaler = MinMaxScaler(feature_range=(0, 1)).fit(X_train)
        X_train = scaler.transform(X_train)
        X_val = scaler.transform(X_val)
        X_test = scaler.transform(X_test)

        X_train = X_train.reshape(X_train.shape[0], 28, 28)
        X_test = X_test.reshape(X_test.shape[0], 28, 28)
        X_val = X_val.reshape(X_val.shape[0], 28, 28)

        X_train = np.reshape(X_train, (-1, 784))
        X_test = np.reshape(X_test, (-1, 784))
        X_val = np.reshape(X_val, (-1, 784))
```



```
        self.x_val = X_val
        self.y_val = y_val
        self.x_test = X_test

        self.y_test = y_test
        self.x = X_train
        self.y = y_train

    def get_val_test(self):
        return self.x_val, self.y_val, self.x_test, self.y_test

    def get_train(self):
        X = self.x
        y = self.y.values
        return X, y


font = {
    "size": 10,
}

matplotlib.style.use("classic")
matplotlib.rc("font", **font)

FLAGS = flags.FLAGS
flags.DEFINE_integer("batch_size", 16, "Number of samples in batch")
flags.DEFINE_float("learning_rate", 1e-3, "Learning rate / step size for SGD")
flags.DEFINE_integer("num_iters", 300, "Number of SGD iterations")


def main(a):
    logging.basicConfig()

    data = Data()

    input = tf.keras.Input(shape=(784,), name="digits")
    x1 = tf.keras.layers.Dense(128, activation="relu")(input)
    x2 = tf.keras.layers.Dense(128, activation="relu")(x1)
    x3 = tf.keras.layers.Dropout(0.05)(x2)
    x4 = tf.keras.layers.Dense(64, activation="relu", kernel_regularizer=l2(0.001
))(x3)
    output = tf.keras.layers.Dense(10, name="predictions")(x4)
    model = tf.keras.Model(inputs=input, outputs=output)
    optimizer = tf.optimizers.Adam(learning_rate=FLAGS.learning_rate)

    lossfunc = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)

    train_metric = keras.metrics.SparseCategoricalAccuracy()
    val_metric = keras.metrics.SparseCategoricalAccuracy()
    test_metric = keras.metrics.SparseCategoricalAccuracy()

    X_train, y_train = data.get_train()
    X_val, y_val, X_test, y_test = data.get_val_test()

    """Reference: https://keras.io/guides/writing_a_training_loop_from_scratch/"""
    train_dataset = tf.data.Dataset.from_tensor_slices((X_train, y_train))
    train_dataset = train_dataset.shuffle(buffer_size=1024).batch(FLAGS.batch_si
ze)
    val_dataset = tf.data.Dataset.from_tensor_slices((X_val, y_val))
    val_dataset = val_dataset.batch(FLAGS.batch_size)
```

```
    train_log = []
    val_log = []
    test_log = []

    bar = trange(FLAGS.num_iters)

    for i in bar:
        # one iteration -> training across entire dataset
        for step, (x_batch, y_batch) in enumerate(train_dataset):
            with tf.GradientTape() as tape:
                logits = model(x_batch, training=True)
                loss = lossfunc(y_batch, logits)

                train_metric.update_state(y_batch, logits)
                train_acc = train_metric.result()
                train_log.append(train_acc.numpy())

                val_logits = model(X_val, training=False)
                val_metric.update_state(y_val, val_logits)
                val_acc = val_metric.result()
                val_metric.reset_states()
                val_log.append(val_acc.numpy())

            grads = tape.gradient(loss, model.trainable_weights)
            optimizer.apply_gradients(zip(grads, model.trainable_weights))
            bar.set_description(f"Loss @ {i} => {loss.numpy():0.6f}")
            bar.refresh()

    test_logits = model(X_test, training=False)
    test_metric.update_state(y_test, test_logits)
    test_acc = test_metric.result()
    test_log.append(test_acc.numpy())

    plt.title("Model Accuracy")
    plt.ylabel("Accuracy")
    plt.xlabel("Epoch")
    plt.plot(train_log, label="Train", c="blue")
    plt.plot(val_log, label="Val", c="purple")
    plt.axhline(y=0.955, linestyle="--", c="green", label="0.955")
    if test_log[0] >= 0.955:
        plt.plot(len(train_log), test_log[0], marker="*", label="Test")
    plt.legend(loc="lower right")
    plt.xlim(0, len(train_log) + 200)
    plt.ylim(0.3, 1)

    plt.tight_layout()
    plt.savefig(f"{script_path}/mnist.pdf")


if __name__ == "__main__":
    app.run(main)

# ./tf.sh python mnist/main.py  --batch_size 32 --num_iters 1
# ./tf.sh  gs -sDEVICE=pdfwrite -dNOPAUSE -dBATCH -dSAFER -sOutputFile=mnist.pdf
 mnist/main.py.pdf mnist/mnist.pdf
```


Model Accuracy
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
Accuracy
0
1000
2000
3000
4000
Epoch
Train
Val
0.955
Test