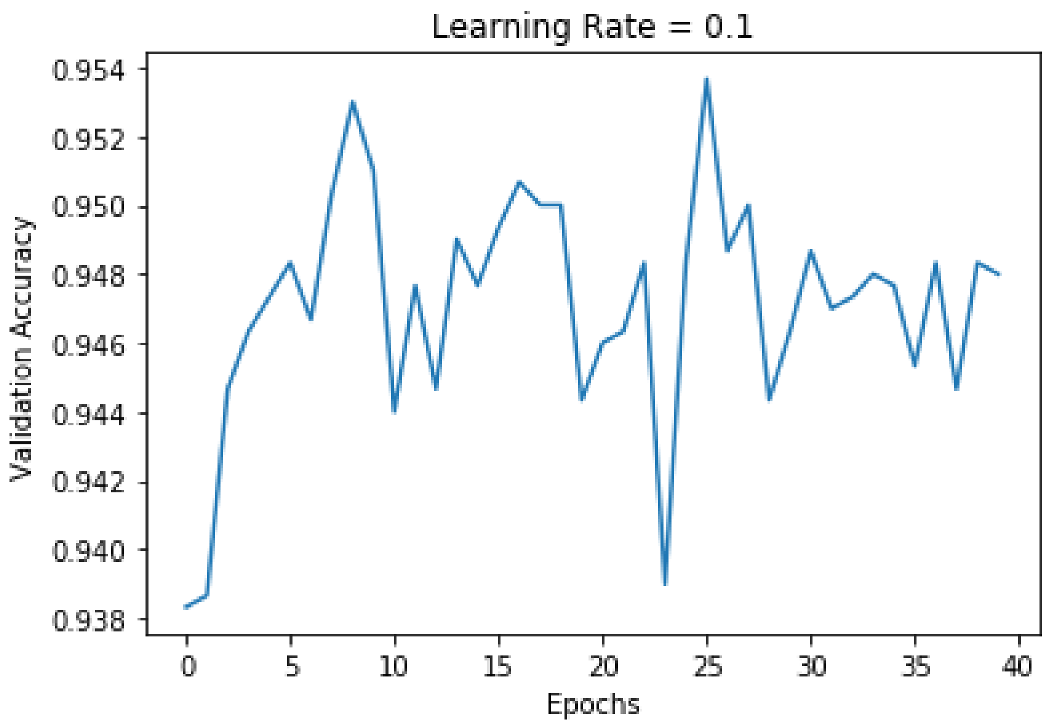
EE449_HW1_Part_4_Scheduled_Learning

5.5 Part 4 Scheduled Learning

[1]:
```
import numpy as np
import tensorflow as tf
import json
from tensorflow.keras.layers import Dense as Dense
from tensorflow.keras.layers import Conv2D as Conv2D
from tensorflow.keras.layers import Input as Input
from tensorflow.keras.layers import MaxPooling2D as MaxPooling2D
from tensorflow.keras.layers import GlobalAveragePooling2D as GlobalAvgPooling2D
from tensorflow.keras.optimizers import Adam
from sklearn.model_selection import train_test_split
from random import shuffle
# HYPERPARAMETERS
BS = 50
EPOCHS = 20
LR = 0.001
# MODEL
def create_mlp2():
    mlp_2 = tf.keras.Sequential(
        [
            Input(shape=(784,)),
            Dense(units=16, activation="relu"),
            Dense(units=64, use_bias=False, activation=None),
            Dense(units=5, activation="softmax"),
        ]
    )
    return mlp_2
# DATA
train_labels = np.load("dataset\\train_labels.npy")
test_labels = np.load("dataset\\test_labels.npy")
train_images = np.load("dataset\\train_images.npy")
test_images = np.load("dataset\\test_images.npy")
## [-1, 1] Scaling
#######################################################
test_images = (
    2 * (test_images - test_images.min()) / (test_images.max() - test_images.
 ↪min())
)  # -1 to 1 scaling
########################################################
train_images = (
    2 * (train_images - train_images.min()) / (train_images.max() -␣
 ↪train_images.min())
)  # -1 to 1 scaling
########################################################
train_image1D, validate_image1D, train_label1D, validate_label1D =␣
 ↪train_test_split(
    train_images, train_labels, test_size=0.1, random_state=42
)
```

[7]:
```
model_name = "mlp2"
model = create_mlp2()
optimizer = tf.keras.optimizers.SGD(learning_rate=0.1)
loss = tf.keras.losses.SparseCategoricalCrossentropy()
metrics = tf.keras.metrics.SparseCategoricalAccuracy()
model.compile(optimizer=optimizer, loss=loss, metrics=metrics)
H = model.fit(
    x=train_image1D,
    y=train_label1D,
    shuffle=True,
    batch_size=BS,
    epochs=24,
    validation_data=(validate_image1D, validate_label1D),
    verbose = 0
)
```

[3]:
```
import matplotlib.pyplot as plt
%matplotlib inline
plt.plot(H.history["val_sparse_categorical_accuracy"])
plt.xlabel('Epochs')
plt.ylabel("Validation Accuracy")
plt.title("Learning Rate = 0.1")
plt.savefig("SLR01")
```

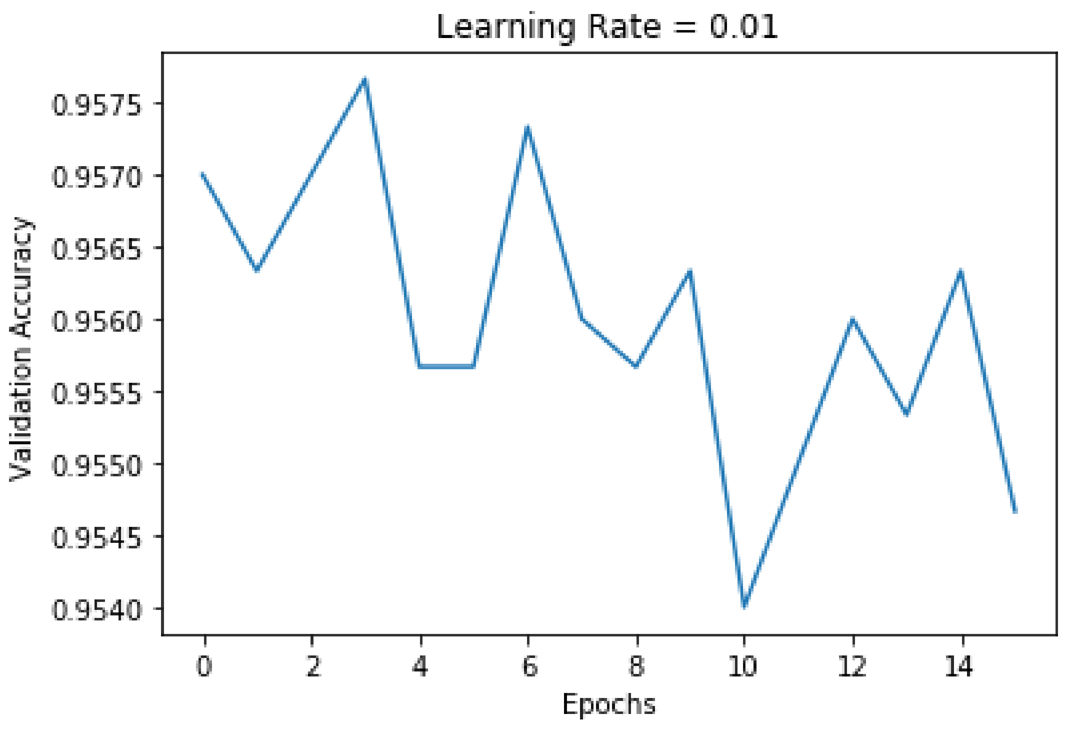


Accuracy stopped increasing at epoch 24. Training 16 epochs for learning_rate = 0.01

```
[8]: optimizer = tf.keras.optimizers.SGD(learning_rate=0.01)
model.compile(optimizer=optimizer, loss=loss, metrics=metrics)

H2 = model.fit(
    x=train_image1D,
    y=train_label1D,
    shuffle=True,
    batch_size=BS,
    epochs=3,
    validation_data=(validate_image1D, validate_label1D),
    verbose=0
)
```

```
[6]: plt.plot(H2.history["val_sparse_categorical_accuracy"])
plt.xlabel('Epochs')
plt.ylabel("Validation Accuracy")
plt.title("Learning Rate = 0.01")
plt.savefig("SLR001")
```

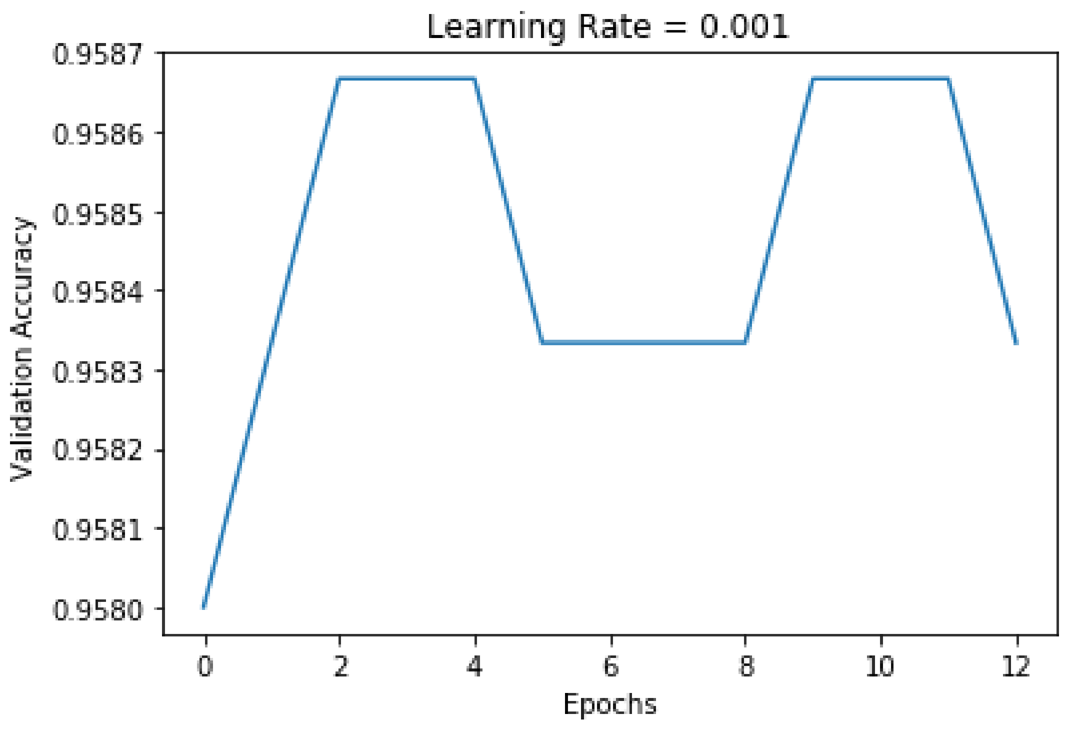


Accuracy stopped increasing at epoch 3. Training 13 epochs for learning_rate = 0.001

[9]:
```
optimizer = tf.keras.optimizers.SGD(learning_rate=0.001)
model.compile(optimizer=optimizer, loss=loss, metrics=metrics)

H3 = model.fit(
    x=train_image1D,
    y=train_label1D,
    shuffle=True,
    batch_size=BS,
    epochs=13,
    validation_data=(validate_image1D, validate_label1D),
    verbose=0
)
```

[10]:
```
plt.plot(H3.history["val_sparse_categorical_accuracy"])
plt.xlabel('Epochs')
plt.ylabel("Validation Accuracy")
plt.title("Learning Rate = 0.001")
plt.savefig("SLR0001")
```



[11]:
```
model.evaluate(x=test_images,y=test_labels,batch_size=BS)
```

```
100/100 [==============================] - 0s 549us/step - loss: 0.1942 -
sparse_categorical_accuracy: 0.9520
```

[11]:
```
[0.19416680932044983, 0.9520000219345093]
```