Problem 1 - Learning Rate, Batch Size, FashionMNIST

Recall cyclical learning rate policy discussed in Lecture 4. The learning rate changes in cyclical manner between lrmin and lrmax, which are hyperparameters that need to be specified. For this problem you first need to read carefully the article referenced below as you will be making use of the code there (in Keras) and modifying it as needed. For those who want to work in Pytorch there are open source implementations of this policy available which you can easily search for and build over them. You will work with FashionMNIST dataset and LeNet-5.

References:

1. Leslie N. Smith Cyclical Learning Rates for Training Neural Networks. Available at https://arxiv.org/abs/1506.01186.
2. Keras implementation of cyclical learning rate policy. Available at https://www.pyimagesearch.com/2019/08/05/keras- learning-rate-finder/.

1. Fix batch size to 64 and start with 10 candidate learning rates between 10−9 and 101 and train your model for 5 epochs for each learning rate. Plot the training loss as a function of learning rate. You should see a curve like Figure 3 in reference below. From that figure identify the values of lrmin and lrmax.

In [ ]:

```
! conda install ipykernel --name Python3
! python -m ipykernel install
! pip3 install cv2
```

```
EnvironmentLocationNotFound: Not a conda environment: /Users/aragaom/opt/anaco
nda3/envs/Python3

Installed kernelspec python3 in /usr/local/share/jupyter/kernels/python3
ERROR: Could not find a version that satisfies the requirement cv2 (from versi
ons: none)
ERROR: No matching distribution found for cv2
WARNING: You are using pip version 22.0.3; however, version 22.3.1 is availabl
e.
You should consider upgrading via the '/Users/aragaom/opt/anaconda3/bin/python
-m pip install --upgrade pip' command.
```

In [ ]:

```
# import the necessary packages
import os
# initialize the list of class label names
CLASSES = ["top", "trouser", "pullover", "dress", "coat",
        "sandal", "shirt", "sneaker", "bag", "ankle boot"]
# define the minimum learning rate, maximum learning rate, batch size,
# step size, CLR method, and number of epochs
MIN_LR = 1e-10
MAX_LR = 1e1
BATCH_SIZE = 64
STEP_SIZE = 5
CLR_METHOD = "triangular"
NUM_EPOCHS = 50
# define the path to the output learning rate finder plot, training
# history plot and cyclical learning rate plot
LRFIND_PLOT_PATH = os.path.sep.join(["output", "lrfind_plot.png"])
```

```
TRAINING_PLOT_PATH = os.path.sep.join(["output", "training_plot.png"])
CLR_PLOT_PATH = os.path.sep.join(["output", "clr_plot.png"])
```

In [ ]:
```
from tensorflow.keras import datasets, layers, models, losses

def create_model():
  model = models.Sequential()

  model.add(layers.Conv2D(6, 5, activation='tanh', input_shape=trainX.shape[1
  model.add(layers.AveragePooling2D(2))


  model.add(layers.Conv2D(16, 5, activation='tanh'))
  model.add(layers.AveragePooling2D(2))

  model.add(layers.Conv2D(120, 5, activation='tanh'))

  model.add(layers.Flatten()) # dense layer is a linear layer and we flatten
  model.add(layers.Dense(84, activation='tanh'))

  model.add(layers.Dense(10, activation='softmax'))

  return model
```

In [ ]:
```
# import tensorflow as tf
# def load_data_mnist_tf(batch_size, resize=None):

#     # load dataset
#     mnist_train, mnist_test = tf.keras.datasets.mnist.load_data()

#     # normalisation and cast as Int datatype
#     process = lambda X, y: (tf.expand_dims(X, axis=3) / 255,tf.cast(y, dtyp
#     # the pixel values must be personalized, so each feature has the same a

#     # resize images if resize is not None
#     resize_fn = lambda X, y: (tf.image.resize_with_pad(X, resize, resize) i
#     # resizing of the image fucntion ??


#     # load train and test batches
#     train_iter = tf.data.Dataset.from_tensor_slices(process(*mnist_train))..
#     test_iter = tf.data.Dataset.from_tensor_slices(process(*mnist_test)).ba

#     return (train_iter, test_iter)
```

In [ ]:
```
# set the matplotlib backend so figures can be saved in the background
# import matplotlib
# matplotlib.use("Agg")
# import the necessary packages
from learningratefinder import LearningRateFinder
from clr_callback import CyclicLR
from sklearn.preprocessing import LabelBinarizer
from sklearn.metrics import classification_report
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.datasets import fashion_mnist
import matplotlib.pyplot as plt
import numpy as np
import argparse
# import cv2
import sys

# construct the argument parser and parse the arguments
```

```
ap = argparse.ArgumentParser()
ap.add_argument("-f", "--lr-find", type=int, default=0,
        help="whether or not to find optimal learning rate")
args, unknown = ap.parse_known_args()
resize_fn = lambda X, y: (tf.image.resize_with_pad(X, resize, resize) if resi
# load the training and testing data
print("[INFO] loading Fashion MNIST data...")
((trainX, trainY), (testX, testY)) = fashion_mnist.load_data()
# Fashion MNIST images are 28x28 but the network we will be training
# is expecting 32x32 images
# trainX = np.array([tf.image.resize(x, [32,32]) for x in trainX])
# testX = np.array([tf.image.resize(x [32,32]) for x in testX])
# scale the pixel intensities to the range [0, 1]
trainX = tf.pad(trainX, [[0, 0], [2,2], [2,2]])/255
testX = tf.pad(testX, [[0, 0], [2,2], [2,2]])/255

trainX = tf.expand_dims(trainX, axis=3, name=None)
testX = tf.expand_dims(testX, axis=3, name=None)


# reshape the data matrices to include a channel dimension (required
# for training)
# trainX = trainX.reshape((trainX.shape[0], 28, 28, 1))
# testX = testX.reshape((testX.shape[0], 28, 28, 1))
# convert the labels from integers to vectors
lb = LabelBinarizer()
trainY = lb.fit_transform(trainY)
testY = lb.transform(testY)
# construct the image generator for data augmentation
aug = ImageDataGenerator(width_shift_range=0.1,
        height_shift_range=0.1, horizontal_flip=True,
        fill_mode="nearest")
```

```
[INFO] loading Fashion MNIST data...
```

In [ ]:

```
print(args)
if args.lr_find == 0:
        # initialize the learning rate finder and then train with learning
        # rates ranging from 1e-10 to 1e+1
        print("[INFO] finding learning rate...")
        lrf = LearningRateFinder(model)
        lrf.find(
                aug.flow(trainX, trainY, batch_size=BATCH_SIZE),
                1e-10, 1e+1,
                stepsPerEpoch=np.ceil((len(trainX) / float(BATCH_SIZE))),epoc
                batchSize=BATCH_SIZE)
        # plot the loss for the various learning rates and save the
        # resulting plot to disk
        lrf.plot_loss()
        plt.savefig("learn_rate_finder.png")
        # gracefully exit the script so we can adjust our learning rates
        # in the config and then train the network for our full set of
        # epochs
        print("[INFO] learning rate finder complete")
        print("[INFO] examine plot and adjust learning rates before training"
        sys.exit(0)
```

```
Namespace(lr_find=0)
[INFO] finding learning rate...
Epoch 1/50
938/938 [==============================] - 20s 21ms/step - loss: 2.3042 - accu
racy: 0.0889
Epoch 2/50
938/938 [==============================] - 22s 24ms/step - loss: 2.3040 - accu
racy: 0.0904
```

```
Epoch 3/50
938/938 [==============================] - 26s 27ms/step - loss: 2.3041 - accu
racy: 0.0891
Epoch 4/50
938/938 [==============================] - 22s 23ms/step - loss: 2.3041 - accu
racy: 0.0884
Epoch 5/50
938/938 [==============================] - 21s 23ms/step - loss: 2.3041 - accu
racy: 0.0881
Epoch 6/50
938/938 [==============================] - 24s 26ms/step - loss: 2.3040 - accu
racy: 0.0893
Epoch 7/50
938/938 [==============================] - 21s 22ms/step - loss: 2.3039 - accu
racy: 0.0903
Epoch 8/50
938/938 [==============================] - 23s 25ms/step - loss: 2.3040 - accu
racy: 0.0888
Epoch 9/50
938/938 [==============================] - 21s 23ms/step - loss: 2.3042 - accu
racy: 0.0895
Epoch 10/50
938/938 [==============================] - 20s 21ms/step - loss: 2.3040 - accu
racy: 0.0906
Epoch 11/50
938/938 [==============================] - 20s 22ms/step - loss: 2.3040 - accu
racy: 0.0883
Epoch 12/50
938/938 [==============================] - 18s 20ms/step - loss: 2.3040 - accu
racy: 0.0901
Epoch 13/50
938/938 [==============================] - 19s 20ms/step - loss: 2.3040 - accu
racy: 0.0894
Epoch 14/50
938/938 [==============================] - 21s 22ms/step - loss: 2.3038 - accu
racy: 0.0900
Epoch 15/50
938/938 [==============================] - 19s 21ms/step - loss: 2.3036 - accu
racy: 0.0891
Epoch 16/50
938/938 [==============================] - 21s 22ms/step - loss: 2.3031 - accu
racy: 0.0895
Epoch 17/50
938/938 [==============================] - 25s 26ms/step - loss: 2.3023 - accu
racy: 0.0906
Epoch 18/50
938/938 [==============================] - 18s 20ms/step - loss: 2.3012 - accu
racy: 0.0908
Epoch 19/50
938/938 [==============================] - 23s 24ms/step - loss: 2.2993 - accu
racy: 0.0924
Epoch 20/50
938/938 [==============================] - 20s 21ms/step - loss: 2.2961 - accu
racy: 0.0936
Epoch 21/50
938/938 [==============================] - 19s 20ms/step - loss: 2.2912 - accu
racy: 0.0991
Epoch 22/50
938/938 [==============================] - 23s 24ms/step - loss: 2.2833 - accu
racy: 0.1035
Epoch 23/50
938/938 [==============================] - 22s 23ms/step - loss: 2.2712 - accu
racy: 0.1101
Epoch 24/50
938/938 [==============================] - 20s 21ms/step - loss: 2.2516 - accu
racy: 0.1253
Epoch 25/50
938/938 [==============================] - 20s 22ms/step - loss: 2.2156 - accu
racy: 0.1923
```

```
Epoch 26/50
938/938 [==============================] - 21s 23ms/step - loss: 2.1364 - accu
racy: 0.3104
Epoch 27/50
938/938 [==============================] - 22s 23ms/step - loss: 1.9122 - accu
racy: 0.4261
Epoch 28/50
938/938 [==============================] - 19s 20ms/step - loss: 1.5188 - accu
racy: 0.5212
Epoch 29/50
938/938 [==============================] - 18s 19ms/step - loss: 1.2735 - accu
racy: 0.5747
Epoch 30/50
938/938 [==============================] - 19s 20ms/step - loss: 1.1131 - accu
racy: 0.6197
Epoch 31/50
938/938 [==============================] - 19s 20ms/step - loss: 0.9794 - accu
racy: 0.6554
Epoch 32/50
938/938 [==============================] - 20s 21ms/step - loss: 0.8833 - accu
racy: 0.6795
Epoch 33/50
938/938 [==============================] - 21s 23ms/step - loss: 0.7916 - accu
racy: 0.7053
Epoch 34/50
938/938 [==============================] - 20s 21ms/step - loss: 0.7199 - accu
racy: 0.7270
Epoch 35/50
938/938 [==============================] - 20s 21ms/step - loss: 0.6553 - accu
racy: 0.7496
Epoch 36/50
938/938 [==============================] - 21s 22ms/step - loss: 0.5932 - accu
racy: 0.7745
Epoch 37/50
938/938 [==============================] - 19s 20ms/step - loss: 0.5423 - accu
racy: 0.7920
Epoch 38/50
938/938 [==============================] - 19s 20ms/step - loss: 0.5109 - accu
racy: 0.8069
Epoch 39/50
938/938 [==============================] - 18s 20ms/step - loss: 0.4957 - accu
racy: 0.8136
Epoch 40/50
938/938 [==============================] - 22s 24ms/step - loss: 0.5003 - accu
racy: 0.8129
Epoch 41/50
938/938 [==============================] - 19s 20ms/step - loss: 0.5445 - accu
racy: 0.8008
Epoch 42/50
938/938 [==============================] - 19s 20ms/step - loss: 0.6879 - accu
racy: 0.7588
Epoch 43/50
938/938 [==============================] - 20s 21ms/step - loss: 1.3491 - accu
racy: 0.6153
Epoch 44/50
938/938 [==============================] - 3s 3ms/step - loss: 2.0126 - accura
cy: 0.5329
[INFO] learning rate finder complete
[INFO] examine plot and adjust learning rates before training
An exception has occurred, use %tb to see the full traceback.

SystemExit: 0

/Users/aragaom/opt/anaconda3/lib/python3.8/site-packages/IPython/core/interact
iveshell.py:3426: UserWarning: To exit: use 'exit', 'quit', or Ctrl-D.
  warn("To exit: use 'exit', 'quit', or Ctrl-D.", stacklevel=1)
```

See figure learn_rate_finder.png

1. Use the cyclical learning rate policy (with exponential decay) and train your network using batch size 64 and lrmin and lrmax values obtained in part 1. Plot train/validation loss and accuracy curve (similar to Figure 4 in reference).

Answer:

if you look at the learn_rate_finder.png file, you will notice that the loss starts to decrease with the learn rate 1e-5 and then spikes up after the learn rate 1e-2.

In [ ]:
```
MIN_LR = 1e-5
MAX_LR = 1e-2

# initialize the optimizer and model
print("[INFO] compiling model...")
opt = SGD(learning_rate=MIN_LR, momentum=0.9)
model = create_model()
model.compile(loss="categorical_crossentropy", optimizer=opt,
        metrics=["accuracy"])
```

```
[INFO] compiling model...
```

In [ ]:
```
stepSize = STEP_SIZE * (trainX.shape[0] // BATCH_SIZE)
clr = CyclicLR(
        mode=CLR_METHOD,
        base_lr=MIN_LR,
        max_lr=MAX_LR,
        step_size=stepSize)
# train the network
print("[INFO] training network...")
H = model.fit(
        x=aug.flow(trainX, trainY, batch_size=BATCH_SIZE),
        validation_data=(testX, testY),
        steps_per_epoch=trainX.shape[0] // BATCH_SIZE,
        epochs=NUM_EPOCHS,
        callbacks=[clr],
        verbose=1)
# evaluate the network and show a classification report
print("[INFO] evaluating network...")
predictions = model.predict(x=testX, batch_size=BATCH_SIZE)
print(classification_report(testY.argmax(axis=1),
        predictions.argmax(axis=1), target_names=CLASSES))
```

```
[INFO] training network...
Epoch 1/50
937/937 [==============================] - 22s 22ms/step - loss: 1.6415 - accu
racy: 0.4202 - val_loss: 0.9228 - val_accuracy: 0.6700
Epoch 2/50
937/937 [==============================] - 20s 22ms/step - loss: 0.8634 - accu
racy: 0.6818 - val_loss: 0.7052 - val_accuracy: 0.7247
Epoch 3/50
937/937 [==============================] - 20s 21ms/step - loss: 0.7005 - accu
racy: 0.7338 - val_loss: 0.5840 - val_accuracy: 0.7802
Epoch 4/50
937/937 [==============================] - 24s 26ms/step - loss: 0.6128 - accu
racy: 0.7646 - val_loss: 0.5261 - val_accuracy: 0.7985
Epoch 5/50
937/937 [==============================] - 24s 25ms/step - loss: 0.5496 - accu
racy: 0.7914 - val_loss: 0.4930 - val_accuracy: 0.8094
Epoch 6/50
937/937 [==============================] - 24s 26ms/step - loss: 0.4912 - accu
racy: 0.8147 - val_loss: 0.4462 - val_accuracy: 0.8315
Epoch 7/50
937/937 [==============================] - 23s 25ms/step - loss: 0.4496 - accu
racy: 0.8324 - val_loss: 0.3957 - val_accuracy: 0.8513
Epoch 8/50
```

```
937/937 [==============================] - 20s 21ms/step - loss: 0.4226 - accu
racy: 0.8418 - val_loss: 0.3955 - val_accuracy: 0.8480
Epoch 9/50
937/937 [==============================] - 19s 21ms/step - loss: 0.4006 - accu
racy: 0.8511 - val_loss: 0.3692 - val_accuracy: 0.8629
Epoch 10/50
937/937 [==============================] - 19s 20ms/step - loss: 0.3842 - accu
racy: 0.8564 - val_loss: 0.3635 - val_accuracy: 0.8643
Epoch 11/50
937/937 [==============================] - 19s 20ms/step - loss: 0.3797 - accu
racy: 0.8598 - val_loss: 0.3718 - val_accuracy: 0.8608
Epoch 12/50
937/937 [==============================] - 20s 22ms/step - loss: 0.3880 - accu
racy: 0.8557 - val_loss: 0.3745 - val_accuracy: 0.8602
Epoch 13/50
937/937 [==============================] - 20s 22ms/step - loss: 0.3908 - accu
racy: 0.8538 - val_loss: 0.3970 - val_accuracy: 0.8491
Epoch 14/50
937/937 [==============================] - 20s 21ms/step - loss: 0.3944 - accu
racy: 0.8513 - val_loss: 0.3608 - val_accuracy: 0.8594
Epoch 15/50
937/937 [==============================] - 20s 21ms/step - loss: 0.3955 - accu
racy: 0.8506 - val_loss: 0.3622 - val_accuracy: 0.8656
Epoch 16/50
937/937 [==============================] - 28s 29ms/step - loss: 0.3836 - accu
racy: 0.8566 - val_loss: 0.3626 - val_accuracy: 0.8655
Epoch 17/50
937/937 [==============================] - 23s 25ms/step - loss: 0.3641 - accu
racy: 0.8627 - val_loss: 0.3361 - val_accuracy: 0.8726
Epoch 18/50
937/937 [==============================] - 21s 22ms/step - loss: 0.3494 - accu
racy: 0.8694 - val_loss: 0.3294 - val_accuracy: 0.8765
Epoch 19/50
937/937 [==============================] - 24s 25ms/step - loss: 0.3357 - accu
racy: 0.8746 - val_loss: 0.3277 - val_accuracy: 0.8763
Epoch 20/50
937/937 [==============================] - 20s 21ms/step - loss: 0.3254 - accu
racy: 0.8789 - val_loss: 0.3194 - val_accuracy: 0.8811
Epoch 21/50
937/937 [==============================] - 21s 23ms/step - loss: 0.3216 - accu
racy: 0.8803 - val_loss: 0.3243 - val_accuracy: 0.8774
Epoch 22/50
937/937 [==============================] - 20s 21ms/step - loss: 0.3309 - accu
racy: 0.8770 - val_loss: 0.3399 - val_accuracy: 0.8745
Epoch 23/50
937/937 [==============================] - 20s 21ms/step - loss: 0.3369 - accu
racy: 0.8734 - val_loss: 0.3336 - val_accuracy: 0.8752
Epoch 24/50
937/937 [==============================] - 20s 21ms/step - loss: 0.3471 - accu
racy: 0.8695 - val_loss: 0.3476 - val_accuracy: 0.8666
Epoch 25/50
937/937 [==============================] - 20s 21ms/step - loss: 0.3484 - accu
racy: 0.8694 - val_loss: 0.3398 - val_accuracy: 0.8730
Epoch 26/50
937/937 [==============================] - 20s 21ms/step - loss: 0.3430 - accu
racy: 0.8724 - val_loss: 0.3298 - val_accuracy: 0.8756
Epoch 27/50
937/937 [==============================] - 19s 21ms/step - loss: 0.3327 - accu
racy: 0.8753 - val_loss: 0.3150 - val_accuracy: 0.8810
Epoch 28/50
937/937 [==============================] - 23s 24ms/step - loss: 0.3176 - accu
racy: 0.8814 - val_loss: 0.3179 - val_accuracy: 0.8815
Epoch 29/50
937/937 [==============================] - 23s 24ms/step - loss: 0.3064 - accu
racy: 0.8839 - val_loss: 0.3035 - val_accuracy: 0.8857
Epoch 30/50
937/937 [==============================] - 22s 23ms/step - loss: 0.2982 - accu
racy: 0.8882 - val_loss: 0.2984 - val_accuracy: 0.8909
Epoch 31/50
```

```
937/937 [==============================] - 21s 22ms/step - loss: 0.2967 - accu
racy: 0.8884 - val_loss: 0.3024 - val_accuracy: 0.8886
Epoch 32/50
937/937 [==============================] - 21s 22ms/step - loss: 0.3040 - accu
racy: 0.8857 - val_loss: 0.3015 - val_accuracy: 0.8892
Epoch 33/50
937/937 [==============================] - 20s 22ms/step - loss: 0.3129 - accu
racy: 0.8833 - val_loss: 0.3140 - val_accuracy: 0.8821
Epoch 34/50
937/937 [==============================] - 21s 22ms/step - loss: 0.3174 - accu
racy: 0.8815 - val_loss: 0.3066 - val_accuracy: 0.8864
Epoch 35/50
937/937 [==============================] - 20s 22ms/step - loss: 0.3276 - accu
racy: 0.8775 - val_loss: 0.3220 - val_accuracy: 0.8801
Epoch 36/50
937/937 [==============================] - 21s 22ms/step - loss: 0.3239 - accu
racy: 0.8794 - val_loss: 0.3047 - val_accuracy: 0.8855
Epoch 37/50
937/937 [==============================] - 22s 23ms/step - loss: 0.3092 - accu
racy: 0.8840 - val_loss: 0.3072 - val_accuracy: 0.8867
Epoch 38/50
937/937 [==============================] - 23s 24ms/step - loss: 0.3016 - accu
racy: 0.8877 - val_loss: 0.2998 - val_accuracy: 0.8908
Epoch 39/50
937/937 [==============================] - 23s 24ms/step - loss: 0.2911 - accu
racy: 0.8906 - val_loss: 0.2940 - val_accuracy: 0.8912
Epoch 40/50
937/937 [==============================] - 22s 24ms/step - loss: 0.2797 - accu
racy: 0.8955 - val_loss: 0.2860 - val_accuracy: 0.8943
Epoch 41/50
937/937 [==============================] - 20s 21ms/step - loss: 0.2784 - accu
racy: 0.8963 - val_loss: 0.2882 - val_accuracy: 0.8939
Epoch 42/50
937/937 [==============================] - 22s 23ms/step - loss: 0.2859 - accu
racy: 0.8927 - val_loss: 0.3026 - val_accuracy: 0.8877
Epoch 43/50
937/937 [==============================] - 23s 25ms/step - loss: 0.2939 - accu
racy: 0.8908 - val_loss: 0.2933 - val_accuracy: 0.8895
Epoch 44/50
937/937 [==============================] - 22s 23ms/step - loss: 0.3005 - accu
racy: 0.8874 - val_loss: 0.3199 - val_accuracy: 0.8786
Epoch 45/50
937/937 [==============================] - 21s 23ms/step - loss: 0.3113 - accu
racy: 0.8824 - val_loss: 0.3160 - val_accuracy: 0.8809
Epoch 46/50
937/937 [==============================] - 23s 25ms/step - loss: 0.3078 - accu
racy: 0.8850 - val_loss: 0.2991 - val_accuracy: 0.8853
Epoch 47/50
937/937 [==============================] - 28s 30ms/step - loss: 0.2947 - accu
racy: 0.8889 - val_loss: 0.3038 - val_accuracy: 0.8885
Epoch 48/50
937/937 [==============================] - 23s 24ms/step - loss: 0.2851 - accu
racy: 0.8931 - val_loss: 0.2921 - val_accuracy: 0.8920
Epoch 49/50
937/937 [==============================] - 23s 24ms/step - loss: 0.2757 - accu
racy: 0.8972 - val_loss: 0.2776 - val_accuracy: 0.8964
Epoch 50/50
937/937 [==============================] - 22s 23ms/step - loss: 0.2683 - accu
racy: 0.9002 - val_loss: 0.2764 - val_accuracy: 0.8977
[INFO] evaluating network...
157/157 [==============================] - 1s 4ms/step
              precision    recall  f1-score   support

         top       0.84      0.85      0.85      1000
     trouser       0.99      0.97      0.98      1000
    pullover       0.82      0.85      0.83      1000
       dress       0.89      0.92      0.90      1000
        coat       0.82      0.82      0.82      1000
      sandal       0.98      0.97      0.97      1000
```

```
       shirt       0.74      0.69      0.71      1000
     sneaker       0.93      0.97      0.95      1000
         bag       0.98      0.98      0.98      1000
  ankle boot       0.98      0.95      0.96      1000

    accuracy                           0.90     10000
   macro avg       0.90      0.90      0.90     10000
weighted avg       0.90      0.90      0.90     10000
```

In [ ]:
```python
N = np.arange(0, NUM_EPOCHS)
# plt.style.use("ggplot")
plt.figure()
plt.plot(N, H.history["loss"], label="train_loss")
plt.plot(N, H.history["val_loss"], label="val_loss")
plt.plot(N, H.history["accuracy"], label="train_acc")
plt.plot(N, H.history["val_accuracy"], label="val_acc")
plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
plt.savefig("training_plot.png")
# plot the learning rate history
N = np.arange(0, len(clr.history["lr"]))
plt.figure()
plt.plot(N, clr.history["lr"])
plt.title("Cyclical Learning Rate (CLR)")
plt.xlabel("Training Iterations")
plt.ylabel("Learning Rate")
plt.savefig("crl_plot.png")
```

1. We want to test if increasing batch size for a fixed learning rate has the same effect as decreasing learning rate for a fixed batch size. Fix learning rate to lrmax and train your network starting with batch size 32 and incrementally going upto 4096 (in increments of a factor of 2; like 32, 64...). You can choose a step size (in terms of number of epochs) to increment the batch size. Plot the training loss vs. log2(batch size). Is the generalization of your final model similar or different than cyclical learning rate policy?

Answer:

Very very similar, but the biggest trade-off is that by increasing the batches I reduce training time by almost 25% of the time spent with the cyclical learning rate, possibly, because as we increase the batch, we need fewer iterations at each epoch.

In [ ]:
```python
batch = 32

print("[INFO] compiling model...")
opt = SGD(learning_rate=MAX_LR, momentum=0.9)
model = create_model()
model.compile(loss="categorical_crossentropy", optimizer=opt,
    metrics=["accuracy"])

loss = []
batches = []

while batch <=4096:
    print("[INFO] training network...")
    H_b = model.fit(
        x=aug.flow(trainX, trainY, batch_size=batch),
        validation_data=(testX, testY),
        steps_per_epoch=trainX.shape[0] // batch,
```

```
        epochs=5,
        # callbacks=[clr],
        verbose=1)
    # evaluate the network and show a classification report
    print("[INFO] evaluating network...")
    predictions = model.predict(x=testX, batch_size=batch)
    print(classification_report(testY.argmax(axis=1),
        predictions.argmax(axis=1), target_names=CLASSES))

    loss.append(H_b.history["loss"][-1])
    batches.append(batch)

    # updating the batch size
    batch*=2
```

```
[INFO] compiling model...
[INFO] training network...
Epoch 1/5
1875/1875 [==============================] - 25s 13ms/step - loss: 0.7799 - ac
curacy: 0.7033 - val_loss: 0.5456 - val_accuracy: 0.7917
Epoch 2/5
1875/1875 [==============================] - 23s 12ms/step - loss: 0.5524 - ac
curacy: 0.7894 - val_loss: 0.5259 - val_accuracy: 0.8023
Epoch 3/5
1875/1875 [==============================] - 20s 11ms/step - loss: 0.4855 - ac
curacy: 0.8187 - val_loss: 0.4129 - val_accuracy: 0.8428
Epoch 4/5
1875/1875 [==============================] - 20s 11ms/step - loss: 0.4445 - ac
curacy: 0.8338 - val_loss: 0.4283 - val_accuracy: 0.8403
Epoch 5/5
1875/1875 [==============================] - 21s 11ms/step - loss: 0.4210 - ac
curacy: 0.8427 - val_loss: 0.3864 - val_accuracy: 0.8526
[INFO] evaluating network...
313/313 [==============================] - 1s 3ms/step
              precision    recall  f1-score   support

         top       0.80      0.81      0.80      1000
     trouser       0.98      0.97      0.98      1000
    pullover       0.86      0.62      0.72      1000
       dress       0.85      0.91      0.88      1000
        coat       0.67      0.88      0.76      1000
      sandal       0.94      0.95      0.95      1000
       shirt       0.66      0.57      0.61      1000
     sneaker       0.94      0.90      0.92      1000
         bag       0.96      0.96      0.96      1000
  ankle boot       0.91      0.96      0.93      1000

    accuracy                           0.85     10000
   macro avg       0.86      0.85      0.85     10000
weighted avg       0.86      0.85      0.85     10000

[INFO] training network...
Epoch 1/5
937/937 [==============================] - 18s 20ms/step - loss: 0.3747 - accu
racy: 0.8603 - val_loss: 0.3624 - val_accuracy: 0.8644
Epoch 2/5
937/937 [==============================] - 20s 21ms/step - loss: 0.3638 - accu
racy: 0.8638 - val_loss: 0.3497 - val_accuracy: 0.8715
Epoch 3/5
937/937 [==============================] - 20s 22ms/step - loss: 0.3553 - accu
racy: 0.8678 - val_loss: 0.3425 - val_accuracy: 0.8775
Epoch 4/5
937/937 [==============================] - 21s 23ms/step - loss: 0.3518 - accu
racy: 0.8695 - val_loss: 0.3630 - val_accuracy: 0.8643
Epoch 5/5
937/937 [==============================] - 21s 23ms/step - loss: 0.3458 - accu
racy: 0.8706 - val_loss: 0.3474 - val_accuracy: 0.8711
```

```
[INFO] evaluating network...
157/157 [==============================] - 1s 4ms/step
              precision    recall  f1-score   support

         top       0.81      0.83      0.82      1000
     trouser       0.98      0.98      0.98      1000
    pullover       0.79      0.81      0.80      1000
       dress       0.89      0.88      0.89      1000
        coat       0.75      0.84      0.79      1000
      sandal       0.98      0.89      0.93      1000
       shirt       0.72      0.60      0.66      1000
     sneaker       0.86      0.98      0.92      1000
         bag       0.98      0.97      0.97      1000
  ankle boot       0.97      0.93      0.95      1000

    accuracy                           0.87     10000
   macro avg       0.87      0.87      0.87     10000
weighted avg       0.87      0.87      0.87     10000

[INFO] training network...
Epoch 1/5
468/468 [==============================] - 22s 47ms/step - loss: 0.3265 - accu
racy: 0.8782 - val_loss: 0.3368 - val_accuracy: 0.8733
Epoch 2/5
468/468 [==============================] - 22s 47ms/step - loss: 0.3229 - accu
racy: 0.8788 - val_loss: 0.3228 - val_accuracy: 0.8791
Epoch 3/5
468/468 [==============================] - 25s 53ms/step - loss: 0.3214 - accu
racy: 0.8806 - val_loss: 0.3268 - val_accuracy: 0.8793
Epoch 4/5
468/468 [==============================] - 23s 49ms/step - loss: 0.3164 - accu
racy: 0.8815 - val_loss: 0.3118 - val_accuracy: 0.8844
Epoch 5/5
468/468 [==============================] - 24s 50ms/step - loss: 0.3159 - accu
racy: 0.8811 - val_loss: 0.3079 - val_accuracy: 0.8886
[INFO] evaluating network...
79/79 [==============================] - 1s 6ms/step
              precision    recall  f1-score   support

         top       0.78      0.89      0.83      1000
     trouser       0.99      0.97      0.98      1000
    pullover       0.83      0.81      0.82      1000
       dress       0.89      0.91      0.90      1000
        coat       0.81      0.82      0.82      1000
      sandal       0.97      0.96      0.96      1000
       shirt       0.75      0.64      0.69      1000
     sneaker       0.95      0.94      0.94      1000
         bag       0.98      0.98      0.98      1000
  ankle boot       0.95      0.96      0.95      1000

    accuracy                           0.89     10000
   macro avg       0.89      0.89      0.89     10000
weighted avg       0.89      0.89      0.89     10000

[INFO] training network...
Epoch 1/5
234/234 [==============================] - 23s 99ms/step - loss: 0.3011 - accu
racy: 0.8870 - val_loss: 0.3068 - val_accuracy: 0.8879
Epoch 2/5
234/234 [==============================] - 25s 106ms/step - loss: 0.3026 - acc
uracy: 0.8877 - val_loss: 0.3095 - val_accuracy: 0.8842
Epoch 3/5
234/234 [==============================] - 23s 96ms/step - loss: 0.2999 - accu
racy: 0.8879 - val_loss: 0.3031 - val_accuracy: 0.8874
Epoch 4/5
234/234 [==============================] - 21s 92ms/step - loss: 0.2983 - accu
racy: 0.8882 - val_loss: 0.3170 - val_accuracy: 0.8818
Epoch 5/5
234/234 [==============================] - 22s 93ms/step - loss: 0.2969 - accu
```

```
racy: 0.8890 - val_loss: 0.3027 - val_accuracy: 0.8857
[INFO] evaluating network...
40/40 [==============================] - 1s 17ms/step
              precision    recall  f1-score   support

         top       0.83      0.82      0.83      1000
     trouser       0.99      0.98      0.98      1000
    pullover       0.77      0.86      0.81      1000
       dress       0.87      0.92      0.89      1000
        coat       0.85      0.74      0.79      1000
      sandal       0.96      0.97      0.96      1000
       shirt       0.71      0.69      0.70      1000
     sneaker       0.94      0.96      0.95      1000
         bag       0.98      0.97      0.98      1000
  ankle boot       0.97      0.95      0.96      1000

    accuracy                           0.89     10000
   macro avg       0.89      0.89      0.89     10000
weighted avg       0.89      0.89      0.89     10000

[INFO] training network...
Epoch 1/5
117/117 [==============================] - 21s 182ms/step - loss: 0.2918 - acc
uracy: 0.8918 - val_loss: 0.2995 - val_accuracy: 0.8888
Epoch 2/5
117/117 [==============================] - 23s 199ms/step - loss: 0.2931 - acc
uracy: 0.8905 - val_loss: 0.2957 - val_accuracy: 0.8915
Epoch 3/5
117/117 [==============================] - 24s 206ms/step - loss: 0.2902 - acc
uracy: 0.8914 - val_loss: 0.3016 - val_accuracy: 0.8874
Epoch 4/5
117/117 [==============================] - 25s 211ms/step - loss: 0.2898 - acc
uracy: 0.8931 - val_loss: 0.2968 - val_accuracy: 0.8899
Epoch 5/5
117/117 [==============================] - 22s 187ms/step - loss: 0.2889 - acc
uracy: 0.8923 - val_loss: 0.3004 - val_accuracy: 0.8904
[INFO] evaluating network...
20/20 [==============================] - 1s 31ms/step
              precision    recall  f1-score   support

         top       0.82      0.86      0.84      1000
     trouser       0.99      0.98      0.98      1000
    pullover       0.80      0.84      0.82      1000
       dress       0.89      0.91      0.90      1000
        coat       0.79      0.83      0.81      1000
      sandal       0.98      0.95      0.97      1000
       shirt       0.77      0.63      0.69      1000
     sneaker       0.92      0.98      0.95      1000
         bag       0.98      0.98      0.98      1000
  ankle boot       0.97      0.94      0.96      1000

    accuracy                           0.89     10000
   macro avg       0.89      0.89      0.89     10000
weighted avg       0.89      0.89      0.89     10000

[INFO] training network...
Epoch 1/5
58/58 [==============================] - 22s 377ms/step - loss: 0.2848 - accur
acy: 0.8949 - val_loss: 0.2964 - val_accuracy: 0.8903
Epoch 2/5
58/58 [==============================] - 22s 371ms/step - loss: 0.2843 - accur
acy: 0.8937 - val_loss: 0.2940 - val_accuracy: 0.8922
Epoch 3/5
58/58 [==============================] - 21s 355ms/step - loss: 0.2863 - accur
acy: 0.8927 - val_loss: 0.2934 - val_accuracy: 0.8919
Epoch 4/5
58/58 [==============================] - 24s 407ms/step - loss: 0.2856 - accur
acy: 0.8934 - val_loss: 0.2951 - val_accuracy: 0.8890
Epoch 5/5
```

```
58/58 [==============================] - 23s 384ms/step - loss: 0.2818 - accur
acy: 0.8963 - val_loss: 0.2942 - val_accuracy: 0.8905
[INFO] evaluating network...
10/10 [==============================] - 1s 62ms/step
              precision    recall  f1-score   support

         top       0.82      0.85      0.84      1000
     trouser       0.99      0.98      0.98      1000
    pullover       0.79      0.85      0.82      1000
       dress       0.90      0.90      0.90      1000
        coat       0.81      0.81      0.81      1000
      sandal       0.96      0.97      0.97      1000
       shirt       0.75      0.65      0.69      1000
     sneaker       0.94      0.96      0.95      1000
         bag       0.98      0.98      0.98      1000
  ankle boot       0.96      0.95      0.96      1000

    accuracy                           0.89     10000
   macro avg       0.89      0.89      0.89     10000
weighted avg       0.89      0.89      0.89     10000

[INFO] training network...
Epoch 1/5
29/29 [==============================] - 22s 787ms/step - loss: 0.2824 - accur
acy: 0.8948 - val_loss: 0.2923 - val_accuracy: 0.8906
Epoch 2/5
29/29 [==============================] - 26s 900ms/step - loss: 0.2820 - accur
acy: 0.8949 - val_loss: 0.2914 - val_accuracy: 0.8922
Epoch 3/5
29/29 [==============================] - 23s 771ms/step - loss: 0.2811 - accur
acy: 0.8964 - val_loss: 0.2909 - val_accuracy: 0.8921
Epoch 4/5
29/29 [==============================] - 24s 808ms/step - loss: 0.2804 - accur
acy: 0.8969 - val_loss: 0.2925 - val_accuracy: 0.8917
Epoch 5/5
29/29 [==============================] - 29s 998ms/step - loss: 0.2795 - accur
acy: 0.8966 - val_loss: 0.2917 - val_accuracy: 0.8912
[INFO] evaluating network...
5/5 [==============================] - 1s 128ms/step
              precision    recall  f1-score   support

         top       0.82      0.85      0.84      1000
     trouser       0.99      0.98      0.98      1000
    pullover       0.82      0.83      0.82      1000
       dress       0.90      0.90      0.90      1000
        coat       0.80      0.83      0.81      1000
      sandal       0.97      0.96      0.97      1000
       shirt       0.74      0.66      0.70      1000
     sneaker       0.94      0.97      0.95      1000
         bag       0.98      0.98      0.98      1000
  ankle boot       0.97      0.95      0.96      1000

    accuracy                           0.89     10000
   macro avg       0.89      0.89      0.89     10000
weighted avg       0.89      0.89      0.89     10000

[INFO] training network...
Epoch 1/5
14/14 [==============================] - 20s 1s/step - loss: 0.2809 - accurac
y: 0.8951 - val_loss: 0.2918 - val_accuracy: 0.8923
Epoch 2/5
14/14 [==============================] - 24s 2s/step - loss: 0.2797 - accurac
y: 0.8950 - val_loss: 0.2908 - val_accuracy: 0.8918
Epoch 3/5
14/14 [==============================] - 19s 1s/step - loss: 0.2805 - accurac
y: 0.8971 - val_loss: 0.2909 - val_accuracy: 0.8927
Epoch 4/5
14/14 [==============================] - 19s 1s/step - loss: 0.2817 - accurac
y: 0.8951 - val_loss: 0.2909 - val_accuracy: 0.8927
```

```
Epoch 5/5
14/14 [==============================] - 19s 1s/step - loss: 0.2801 - accurac
y: 0.8967 - val_loss: 0.2906 - val_accuracy: 0.8918
[INFO] evaluating network...
3/3 [==============================] - 1s 186ms/step
              precision    recall  f1-score   support

         top       0.82      0.86      0.84      1000
     trouser       0.99      0.98      0.98      1000
    pullover       0.81      0.84      0.82      1000
       dress       0.89      0.91      0.90      1000
        coat       0.81      0.81      0.81      1000
      sandal       0.97      0.96      0.97      1000
       shirt       0.75      0.67      0.71      1000
     sneaker       0.94      0.96      0.95      1000
         bag       0.98      0.98      0.98      1000
  ankle boot       0.97      0.95      0.96      1000

    accuracy                           0.89     10000
   macro avg       0.89      0.89      0.89     10000
weighted avg       0.89      0.89      0.89     10000
```

In [ ]:

```
plt.figure()
plt.plot(batches, loss)
plt.title("Batches vs loss")
plt.xlabel("Batch Size")
plt.ylabel("Training Loss")
plt.savefig("batches_loss.png")
```