Lab 2: Cats vs Dogs

In this lab, you will train a convolutional neural network to classify an image into one of two classes: "cat" or "dog". The code for the neural networks you train will be written for you, and you are not (yet!) expected to understand all provided code. However, by the end of the lab, you should be able to:

1. Understand at a high level the training loop for a machine learning model.
2. Understand the distinction between training, validation, and test data.
3. The concepts of overfitting and underfitting.
4. Investigate how different hyperparameters, such as learning rate and batch size, affect the success of training.
5. Compare an ANN (aka Multi-Layer Perceptron) with a CNN.

What to submit

Submit a PDF file containing all your code, outputs, and write-up from parts 1-5. You can produce a PDF of your Google Colab file by going to **File > Print** and then save as PDF. The Colab instructions has more information.

Do not submit any other files produced by your code.

Include a link to your colab file in your submission.

Please use Google Colab to complete this assignment. If you want to use Jupyter Notebook, please complete the assignment and upload your Jupyter Notebook file to Google Colab for submission.

With Colab, you can export a PDF file using the menu option `File -> Print` and save as PDF file. **Adjust the scaling to ensure that the text is not cutoff at the margins.**

Colab Link

Include a link to your colab file here

Colab Link: https://drive.google.com/file/d/1UKqoWYQNnedz_74wCQVTvcVV6Q9jcSdj/view?usp=sharing

```
import numpy as np
import time
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
import torchvision
from torch.utils.data.sampler import SubsetRandomSampler
import torchvision.transforms as transforms
```

Part 0. Helper Functions

We will be making use of the following helper functions. You will be asked to look at and possibly modify some of these, but you are not expected to understand all of them.

You should look at the function names and read the docstrings. If you are curious, come back and explore the code *after* making some progress on the lab.

```
#####################################################################
#########
# Data Loading

def get_relevant_indices(dataset, classes, target_classes):
    """ Return the indices for datapoints in the dataset that belongs
to the
    desired target classes, a subset of all possible classes.

    Args:
        dataset: Dataset object
        classes: A list of strings denoting the name of each class
        target_classes: A list of strings denoting the name of desired
classes
                        Should be a subset of the 'classes'
    Returns:
        indices: list of indices that have labels corresponding to one
of the
                 target classes
    """
    indices = []
    for i in range(len(dataset)):
        # Check if the label is in the target classes
        label_index = dataset[i][1] # ex: 3
        label_class = classes[label_index] # ex: 'cat'
        if label_class in target_classes:
            indices.append(i)
    return indices

def get_data_loader(target_classes, batch_size):
    """ Loads images of cats and dogs, splits the data into training,
validation
    and testing datasets. Returns data loaders for the three
preprocessed datasets.

    Args:
        target_classes: A list of strings denoting the name of the
desired
                        classes. Should be a subset of the argument
'classes'
        batch_size: A int representing the number of samples per batch
```

```
    Returns:
        train_loader: iterable training dataset organized according to
batch size
        val_loader: iterable validation dataset organized according to
batch size
        test_loader: iterable testing dataset organized according to
batch size
        classes: A list of strings denoting the name of each class
    """

    classes = ('plane', 'car', 'bird', 'cat',
               'deer', 'dog', 'frog', 'horse', 'ship', 'truck')

#########################################################################
##
    # The output of torchvision datasets are PILImage images of range
[0, 1].
    # We transform them to Tensors of normalized range [-1, 1].
    transform = transforms.Compose(
        [transforms.ToTensor(),
         transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])
    # Load CIFAR10 training data
    trainset = torchvision.datasets.CIFAR10(root='./data', train=True,
                                            download=True,
transform=transform)
    # Get the list of indices to sample from
    relevant_indices = get_relevant_indices(trainset, classes,
target_classes)

    # Split into train and validation
    np.random.seed(1000) # Fixed numpy random seed for reproducible
shuffling
    np.random.shuffle(relevant_indices)
    split = int(len(relevant_indices) * 0.8) #split at 80%

    # split into training and validation indices
    relevant_train_indices, relevant_val_indices =
relevant_indices[:split], relevant_indices[split:]
    train_sampler = SubsetRandomSampler(relevant_train_indices)
    train_loader = torch.utils.data.DataLoader(trainset,
batch_size=batch_size,
                                               num_workers=1,
sampler=train_sampler)
    val_sampler = SubsetRandomSampler(relevant_val_indices)
    val_loader = torch.utils.data.DataLoader(trainset,
batch_size=batch_size,
                                             num_workers=1,
sampler=val_sampler)
    # Load CIFAR10 testing data
    testset = torchvision.datasets.CIFAR10(root='./data', train=False,
```

```
                                            download=True,
transform=transform)
    # Get the list of indices to sample from
    relevant_test_indices = get_relevant_indices(testset, classes,
target_classes)
    test_sampler = SubsetRandomSampler(relevant_test_indices)
    test_loader = torch.utils.data.DataLoader(testset,
batch_size=batch_size,
                                              num_workers=1,
sampler=test_sampler)
    return train_loader, val_loader, test_loader, classes

################################################################
#########
# Training
def get_model_name(name, batch_size, learning_rate, epoch):
    """ Generate a name for the model consisting of all the
hyperparameter values

    Args:
        config: Configuration object containing the hyperparameters
    Returns:
        path: A string with the hyperparameter name and value
concatenated
    """
    path = "model_{0}_bs{1}_lr{2}_epoch{3}".format(name,
                                                   batch_size,
                                                   learning_rate,
                                                   epoch)
    return path

def normalize_label(labels):
    """
    Given a tensor containing 2 possible values, normalize this to 0/1

    Args:
        labels: a 1D tensor containing two possible scalar values
    Returns:
        A tensor normalize to 0/1 value
    """
    max_val = torch.max(labels)
    min_val = torch.min(labels)
    norm_labels = (labels - min_val)/(max_val - min_val)
    return norm_labels

def evaluate(net, loader, criterion):
    """ Evaluate the network on the validation set.

     Args:
         net: PyTorch neural network object
```

```
        loader: PyTorch data loader for the validation set
        criterion: The loss function
    Returns:
        err: A scalar for the avg classification error over the
validation set
        loss: A scalar for the average loss function over the
validation set
    """
    total_loss = 0.0
    total_err = 0.0
    total_epoch = 0
    for i, data in enumerate(loader, 0):
        inputs, labels = data
        labels = normalize_label(labels)  # Convert labels to 0/1
        outputs = net(inputs)
        loss = criterion(outputs, labels.float())
        corr = (outputs > 0.0).squeeze().long() != labels
        total_err += int(corr.sum())
        total_loss += loss.item()
        total_epoch += len(labels)
    err = float(total_err) / total_epoch
    loss = float(total_loss) / (i + 1)
    return err, loss

######################################################################
#########
# Training Curve
def plot_training_curve(path):
    """ Plots the training curve for a model run, given the csv files
    containing the train/validation error/loss.

    Args:
        path: The base path of the csv files produced during training
    """
    import matplotlib.pyplot as plt
    train_err = np.loadtxt("{}_train_err.csv".format(path))
    val_err = np.loadtxt("{}_val_err.csv".format(path))
    train_loss = np.loadtxt("{}_train_loss.csv".format(path))
    val_loss = np.loadtxt("{}_val_loss.csv".format(path))
    plt.title("Train vs Validation Error")
    n = len(train_err) # number of epochs
    plt.plot(range(1,n+1), train_err, label="Train")
    plt.plot(range(1,n+1), val_err, label="Validation")
    plt.xlabel("Epoch")
    plt.ylabel("Error")
    plt.legend(loc='best')
    plt.show()
    plt.title("Train vs Validation Loss")
    plt.plot(range(1,n+1), train_loss, label="Train")
    plt.plot(range(1,n+1), val_loss, label="Validation")
```

```
    plt.xlabel("Epoch")
    plt.ylabel("Loss")
    plt.legend(loc='best')
    plt.show()
```

Part 1. Visualizing the Data [7 pt]

We will make use of some of the CIFAR-10 data set, which consists of colour images of size 32x32 pixels belonging to 10 categories. You can find out more about the dataset at https://www.cs.toronto.edu/~kriz/cifar.html

For this assignment, we will only be using the cat and dog categories. We have included code that automatically downloads the dataset the first time that the main script is run.

```
# This will download the CIFAR-10 dataset to a folder called "data"
# the first time you run this code.
train_loader, val_loader, test_loader, classes = get_data_loader(
    target_classes=["cat", "dog"],
    batch_size=1) # One image per batch
```

```
Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to
./data/cifar-10-python.tar.gz
```

```
{"model_id":"5aa7b19c40414d95b90e4320d65d0eec","version_major":2,"vers
ion_minor":0}
```

```
Extracting ./data/cifar-10-python.tar.gz to ./data
Files already downloaded and verified
```

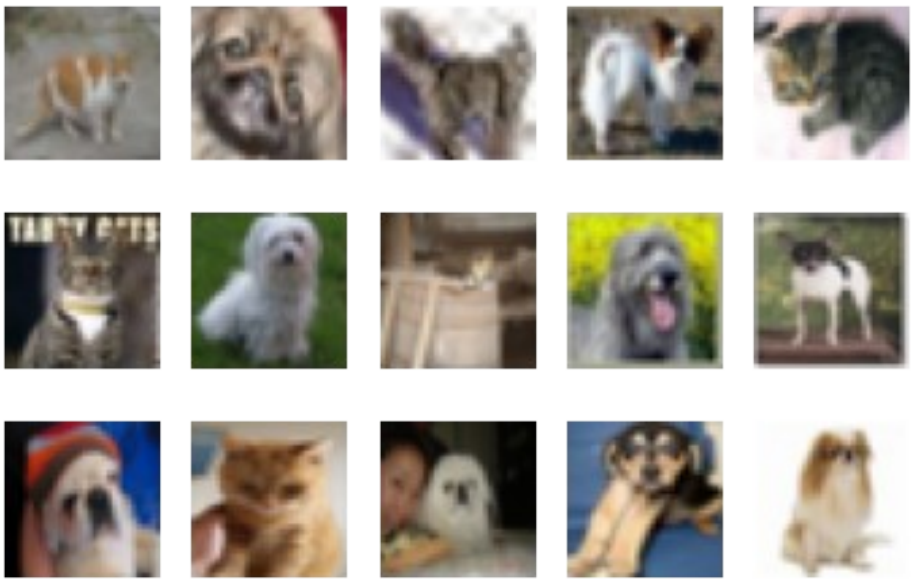
Part (a) -- 1 pt

Visualize some of the data by running the code below. Include the visualization in your writeup.

(You don't need to submit anything else.)

```
import matplotlib.pyplot as plt

k = 0
for images, labels in train_loader:
    # since batch_size = 1, there is only 1 image in `images`
    image = images[0]
    # place the colour channel at the end, instead of at the beginning
    img = np.transpose(image, [1,2,0])
    # normalize pixel intensity values to [0, 1]
    img = img / 2 + 0.5
    plt.subplot(3, 5, k+1)
    plt.axis('off')
    plt.imshow(img)

    k += 1
```

```
    if k > 14:
        break
```



Part (b) -- 3 pt

How many training examples do we have for the combined `cat` and `dog` classes? What about validation examples? What about test examples?

```
print("Training Examples: ", len(train_loader))
print("Validation Examples: ", len(val_loader))
print("Testing Examples: ", len(test_loader))
```

```
Training Examples:  8000
Validation Examples:  2000
Testing Examples:  2000
```

Part (c) -- 3pt

Why do we need a validation set when training our model? What happens if we judge the performance of our models using the training set loss/error instead of the validation set loss/error?

```
"""
If we only judge the performance of our models using the training set,
the model will
start to overfit to the training data and stray away from actually
identifying cats
and dogs in general. We use a validation set to adjust our model and
retrain
using the training set.
"""
```

Part 2. Training [15 pt]

We define two neural networks, a `LargeNet` and `SmallNet`. We'll be training the networks in this section.

You won't understand fully what these networks are doing until the next few classes, and that's okay. For this assignment, please focus on learning how to train networks, and how hyperparameters affect training.

```
class LargeNet(nn.Module):
    def __init__(self):
        super(LargeNet, self).__init__()
        self.name = "large"
        self.conv1 = nn.Conv2d(3, 5, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(5, 10, 5)
        self.fc1 = nn.Linear(10 * 5 * 5, 32)
        self.fc2 = nn.Linear(32, 1)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = x.view(-1, 10 * 5 * 5)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        x = x.squeeze(1) # Flatten to [batch_size]
        return x

class SmallNet(nn.Module):
    def __init__(self):
        super(SmallNet, self).__init__()
        self.name = "small"
        self.conv = nn.Conv2d(3, 5, 3)
        self.pool = nn.MaxPool2d(2, 2)
        self.fc = nn.Linear(5 * 7 * 7, 1)

    def forward(self, x):
        x = self.pool(F.relu(self.conv(x)))
        x = self.pool(x)
        x = x.view(-1, 5 * 7 * 7)
        x = self.fc(x)
        x = x.squeeze(1) # Flatten to [batch_size]
        return x

small_net = SmallNet()
large_net = LargeNet()
```

Part (a) -- 2pt

The methods `small_net.parameters()` and `large_net.parameters()` produces an iterator of all the trainable parameters of the network. These parameters are torch tensors containing many scalar values.

We haven't learned how how the parameters in these high-dimensional tensors will be used, but we should be able to count the number of parameters. Measuring the number of parameters in a network is one way of measuring the "size" of a network.

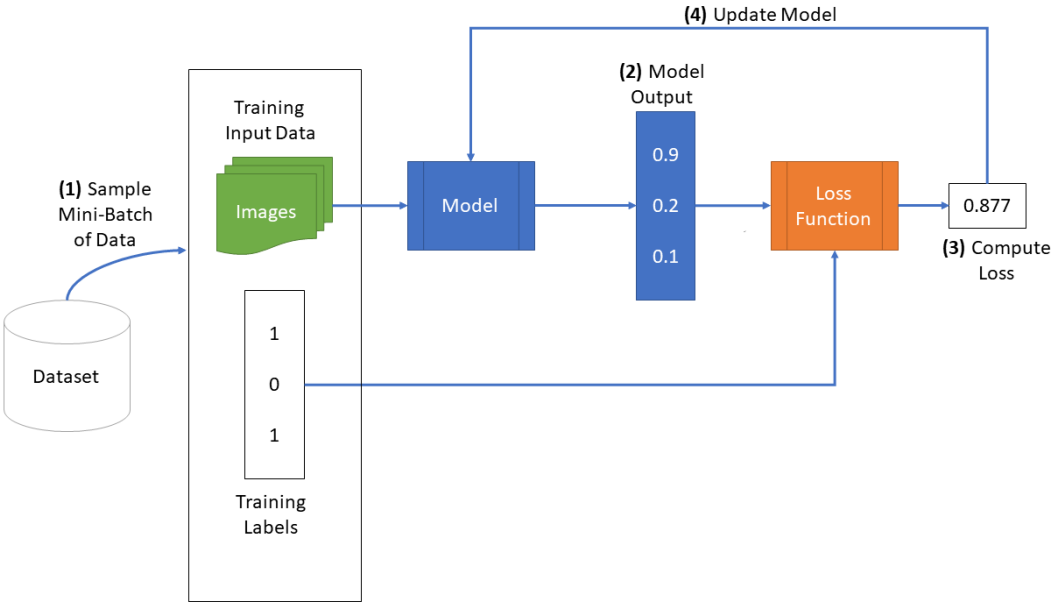
What is the total number of parameters in `small_net` and in `large_net`? (Hint: how many numbers are in each tensor?)

```
from numpy.ma.core import size
print("Small NN Parameters:")
for param in small_net.parameters():
    print(param.shape)

# Small Net: 5*3*3*3 + 5 + 1*245 + 1 = 386

print("\nLarge NN Parameters:")
for param in large_net.parameters():
    print(param.shape)

# Large Net: 5*3*5*5 + 5 + 10*5*5*5 + 10 + 32*250 + 32 + 1*32 + 1 =
9705
```

```
Small NN Parameters:
torch.Size([5, 3, 3, 3])
torch.Size([5])
torch.Size([1, 245])
torch.Size([1])

Large NN Parameters:
torch.Size([5, 3, 5, 5])
torch.Size([5])
torch.Size([10, 5, 5, 5])
torch.Size([10])
torch.Size([32, 250])
torch.Size([32])
torch.Size([1, 32])
torch.Size([1])
```

The function train_net

The function `train_net` below takes an untrained neural network (like `small_net` and `large_net`) and several other parameters. You should be able to understand how this function works. The figure below shows the high level training loop for a machine learning model:




```
def train_net(net, batch_size=64, learning_rate=0.01, num_epochs=30):

########################################################################
##
    # Train a classifier on cats vs dogs
    target_classes = ["cat", "dog"]

########################################################################
##
    # Fixed PyTorch random seed for reproducible result
    torch.manual_seed(1000)

########################################################################
##
    # Obtain the PyTorch data loader objects to load batches of the
datasets
    train_loader, val_loader, test_loader, classes = get_data_loader(
            target_classes, batch_size)

########################################################################
##
    # Define the Loss function and optimizer
    # The loss function will be Binary Cross Entropy (BCE). In this
case we
    # will use the BCEWithLogitsLoss which takes unnormalized output
from
    # the neural network and scalar label.
    # Optimizer will be SGD with Momentum.
    criterion = nn.BCEWithLogitsLoss()
    optimizer = optim.SGD(net.parameters(), lr=learning_rate,
momentum=0.9)
```

```
#########################################################################
##
    # Set up some numpy arrays to store the training/test
loss/erruracy
    train_err = np.zeros(num_epochs)
    train_loss = np.zeros(num_epochs)
    val_err = np.zeros(num_epochs)
    val_loss = np.zeros(num_epochs)

#########################################################################
##
    # Train the network
    # Loop over the data iterator and sample a new batch of training
data
    # Get the output from the network, and optimize our loss function.
    start_time = time.time()
    for epoch in range(num_epochs):  # loop over the dataset multiple
times
        total_train_loss = 0.0
        total_train_err = 0.0
        total_epoch = 0
        for i, data in enumerate(train_loader, 0):
            # Get the inputs
            inputs, labels = data
            labels = normalize_label(labels) # Convert labels to 0/1
            # Zero the parameter gradients
            optimizer.zero_grad()
            # Forward pass, backward pass, and optimize
            outputs = net(inputs)
            loss = criterion(outputs, labels.float())
            loss.backward()
            optimizer.step()
            # Calculate the statistics
            corr = (outputs > 0.0).squeeze().long() != labels
            total_train_err += int(corr.sum())
            total_train_loss += loss.item()
            total_epoch += len(labels)
        train_err[epoch] = float(total_train_err) / total_epoch
        train_loss[epoch] = float(total_train_loss) / (i+1)
        val_err[epoch], val_loss[epoch] = evaluate(net, val_loader,
criterion)
        print(("Epoch {}: Train err: {}, Train loss: {} |"+
               "Validation err: {}, Validation loss: {}").format(
                   epoch + 1,
                   train_err[epoch],
                   train_loss[epoch],
                   val_err[epoch],
                   val_loss[epoch]))
        # Save the current model (checkpoint) to a file
```

```
        model_path = get_model_name(net.name, batch_size,
learning_rate, epoch)
        torch.save(net.state_dict(), model_path)
    print('Finished Training')
    end_time = time.time()
    elapsed_time = end_time - start_time
    print("Total time elapsed: {:.2f} seconds".format(elapsed_time))
    # Write the train/test loss/err into CSV file for plotting later
    epochs = np.arange(1, num_epochs + 1)
    np.savetxt("{}_train_err.csv".format(model_path), train_err)
    np.savetxt("{}_train_loss.csv".format(model_path), train_loss)
    np.savetxt("{}_val_err.csv".format(model_path), val_err)
    np.savetxt("{}_val_loss.csv".format(model_path), val_loss)
```

Part (b) -- 1pt

The parameters to the function `train_net` are hyperparameters of our neural network. We made these hyperparameters easy to modify so that we can tune them later on.

What are the default values of the parameters `batch_size`, `learning_rate`, and `num_epochs`?

```
"""
The default values of the parameters are:
batch_size = 64
learning_rate = 0.01
num_epochs = 30
"""
```

Part (c) -- 3 pt

What files are written to disk when we call `train_net` with `small_net`, and train for 5 epochs? Provide a list of all the files written to disk, and what information the files contain.

```
SmallNet_1 = SmallNet()
train_net(SmallNet_1, num_epochs=5)

# The following 5 files are written to the main directory
# and contain each epochs' parameter values/ model weights:
#
# model_small_bs64_lr0.01_epoch0: checkpoint at epoch 0
# model_small_bs64_lr0.01_epoch1: checkpoint at epoch 1
# model_small_bs64_lr0.01_epoch2: checkpoint at epoch 2
# model_small_bs64_lr0.01_epoch3: checkpoint at epoch 3
# model_small_bs64_lr0.01_epoch4: checkpoint at epoch 4

# The following 4 files are written to the main directory and contain
# the training and validation error and loss from the final epoch:
#
# model_small_bs64_lr0.01_epoch4_train_err.csv: training error
# model_small_bs64_lr0.01_epoch4_train_loss.csv: training loss
```

```
# model_small_bs64_lr0.01_epoch4_val_err.csv: validation error
# model_small_bs64_lr0.01_epoch4_val_loss.csv: validation loss
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.415375, Train loss: 0.670185649394989 |
Validation err: 0.37, Validation loss: 0.6514860149472952
Epoch 2: Train err: 0.36125, Train loss: 0.6439669713973999 |
Validation err: 0.377, Validation loss: 0.657435892149806
Epoch 3: Train err: 0.348875, Train loss: 0.6300341081619263 |
Validation err: 0.3495, Validation loss: 0.6222464125603437
Epoch 4: Train err: 0.337875, Train loss: 0.6165080904960633 |
Validation err: 0.354, Validation loss: 0.6242818050086498
Epoch 5: Train err: 0.32575, Train loss: 0.6065622324943543 |
Validation err: 0.325, Validation loss: 0.611550921574235
Finished Training
Total time elapsed: 30.74 seconds
```

Part (d) -- 2pt

Train both `small_net` and `large_net` using the function `train_net` and its default parameters. The function will write many files to disk, including a model checkpoint (saved values of model weights) at the end of each epoch.

If you are using Google Colab, you will need to mount Google Drive so that the files generated by `train_net` gets saved. We will be using these files in part (d). (See the Google Colab tutorial for more information about this.)

Report the total time elapsed when training each network. Which network took longer to train? Why?

```
# Since the function writes files to disk, you will need to mount
# your Google Drive. If you are working on the lab locally, you
# can comment out this code.
```

```
from google.colab import drive
drive.mount('/content/gdrive/')
```

```
Mounted at /content/gdrive/
```

```
train_net(small_net)
train_net(large_net)
# small_net is faster, because it has fewer parameters than large_net,
so it
# takes less time to do the training
#
# Small NN: 134.56 sec
# Large NN: 156.93 sec
# Surprising, despite the much large number of parameters in the large
NN, the
# time difference isn't that large.
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.43925, Train loss: 0.6837761998176575 |
Validation err: 0.3915, Validation loss: 0.6662304494529963
Epoch 2: Train err: 0.381, Train loss: 0.6527652173042298 |Validation
err: 0.3935, Validation loss: 0.6627088207751513
Epoch 3: Train err: 0.342125, Train loss: 0.6253658514022827 |
Validation err: 0.335, Validation loss: 0.618512349203229
Epoch 4: Train err: 0.324, Train loss: 0.6055522425174713 |Validation
err: 0.3375, Validation loss: 0.6168343275785446
Epoch 5: Train err: 0.314, Train loss: 0.5930070235729218 |Validation
err: 0.32, Validation loss: 0.6096986010670662
Epoch 6: Train err: 0.299875, Train loss: 0.5810152497291565 |
Validation err: 0.326, Validation loss: 0.605014132335782
Epoch 7: Train err: 0.297125, Train loss: 0.5772572119235992 |
Validation err: 0.332, Validation loss: 0.6006468189880252
Epoch 8: Train err: 0.293625, Train loss: 0.5692502632141113 |
Validation err: 0.3085, Validation loss: 0.5972220627591014
Epoch 9: Train err: 0.28975, Train loss: 0.5679050085544586 |
Validation err: 0.3235, Validation loss: 0.5948699712753296
Epoch 10: Train err: 0.289625, Train loss: 0.5611771538257598 |
Validation err: 0.318, Validation loss: 0.5917983828112483
Epoch 11: Train err: 0.28275, Train loss: 0.5590389950275421 |
Validation err: 0.3215, Validation loss: 0.5937980338931084
Epoch 12: Train err: 0.278125, Train loss: 0.552915855884552 |
Validation err: 0.322, Validation loss: 0.5957375196740031
Epoch 13: Train err: 0.280875, Train loss: 0.5536699004173279 |
Validation err: 0.31, Validation loss: 0.5928990021348
Epoch 14: Train err: 0.278, Train loss: 0.5482110195159912 |Validation
err: 0.314, Validation loss: 0.6091358661651611
Epoch 15: Train err: 0.276375, Train loss: 0.5475579555034638 |
Validation err: 0.316, Validation loss: 0.5957867605611682
Epoch 16: Train err: 0.280875, Train loss: 0.5506727433204651 |
Validation err: 0.312, Validation loss: 0.6070869211107492
Epoch 17: Train err: 0.276625, Train loss: 0.549470397233963 |
Validation err: 0.307, Validation loss: 0.5909738149493933
Epoch 18: Train err: 0.271, Train loss: 0.5432693302631378 |Validation
err: 0.3175, Validation loss: 0.5889089312404394
Epoch 19: Train err: 0.271625, Train loss: 0.5406748006343841 |
Validation err: 0.3225, Validation loss: 0.6017081495374441
Epoch 20: Train err: 0.272375, Train loss: 0.5401509728431702 |
Validation err: 0.2995, Validation loss: 0.595357958227396
Epoch 21: Train err: 0.273625, Train loss: 0.5404087448120117 |
Validation err: 0.3095, Validation loss: 0.5910016968846321
Epoch 22: Train err: 0.272, Train loss: 0.5392598848342895 |Validation
err: 0.3075, Validation loss: 0.5957168955355883
Epoch 23: Train err: 0.27, Train loss: 0.5399896326065063 |Validation
err: 0.3035, Validation loss: 0.5936928503215313
Epoch 24: Train err: 0.26875, Train loss: 0.538470311164856 |
Validation err: 0.3095, Validation loss: 0.5963647030293941
```

```
Epoch 25: Train err: 0.269875, Train loss: 0.5359496214389801 |
Validation err: 0.3125, Validation loss: 0.5913036968559027
Epoch 26: Train err: 0.27, Train loss: 0.535960717201233 |Validation
err: 0.3, Validation loss: 0.5957943461835384
Epoch 27: Train err: 0.27075, Train loss: 0.5362830855846406 |
Validation err: 0.3145, Validation loss: 0.6072113066911697
Epoch 28: Train err: 0.269625, Train loss: 0.5371881670951844 |
Validation err: 0.2975, Validation loss: 0.5893184989690781
Epoch 29: Train err: 0.272125, Train loss: 0.5361146366596222 |
Validation err: 0.3025, Validation loss: 0.5972883393988013
Epoch 30: Train err: 0.264625, Train loss: 0.5357177419662476 |
Validation err: 0.32, Validation loss: 0.6011227704584599
Finished Training
Total time elapsed: 134.56 seconds
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.455875, Train loss: 0.6898005352020263 |
Validation err: 0.42, Validation loss: 0.6789227239787579
Epoch 2: Train err: 0.42075, Train loss: 0.6776594214439392 |
Validation err: 0.433, Validation loss: 0.6792406104505062
Epoch 3: Train err: 0.400625, Train loss: 0.6654761843681335 |
Validation err: 0.3835, Validation loss: 0.6486953217536211
Epoch 4: Train err: 0.36925, Train loss: 0.6469117832183838 |
Validation err: 0.3655, Validation loss: 0.6398825291544199
Epoch 5: Train err: 0.35825, Train loss: 0.6307332396507264 |
Validation err: 0.349, Validation loss: 0.6249986849725246
Epoch 6: Train err: 0.339125, Train loss: 0.6158002796173095 |
Validation err: 0.3335, Validation loss: 0.6161199659109116
Epoch 7: Train err: 0.328, Train loss: 0.6027200078964233 |Validation
err: 0.341, Validation loss: 0.6071056127548218
Epoch 8: Train err: 0.31075, Train loss: 0.5856167476177215 |
Validation err: 0.3265, Validation loss: 0.5992275290191174
Epoch 9: Train err: 0.30125, Train loss: 0.5798469822406769 |
Validation err: 0.3185, Validation loss: 0.5926393559202552
Epoch 10: Train err: 0.294625, Train loss: 0.5643817067146302 |
Validation err: 0.3145, Validation loss: 0.5917625175788999
Epoch 11: Train err: 0.291625, Train loss: 0.5561280665397644 |
Validation err: 0.3145, Validation loss: 0.590191300958395
Epoch 12: Train err: 0.274375, Train loss: 0.5398570637702942 |
Validation err: 0.299, Validation loss: 0.578259689733386
Epoch 13: Train err: 0.269875, Train loss: 0.5318400142192841 |
Validation err: 0.322, Validation loss: 0.5947587918490171
Epoch 14: Train err: 0.25925, Train loss: 0.5164860756397247 |
Validation err: 0.31, Validation loss: 0.588407845236361
Epoch 15: Train err: 0.251125, Train loss: 0.5116680881977081 |
Validation err: 0.299, Validation loss: 0.5670721856877208
Epoch 16: Train err: 0.252625, Train loss: 0.5086743261814117 |
Validation err: 0.2935, Validation loss: 0.5707025490701199
Epoch 17: Train err: 0.2475, Train loss: 0.49469798636436463 |
Validation err: 0.292, Validation loss: 0.5724088903516531
```

```
Epoch 18: Train err: 0.241375, Train loss: 0.48258004331588744 |
Validation err: 0.2995, Validation loss: 0.569452466443181
Epoch 19: Train err: 0.236375, Train loss: 0.4779411377906799 |
Validation err: 0.295, Validation loss: 0.5728982025757432
Epoch 20: Train err: 0.229, Train loss: 0.467186149597168 |Validation
err: 0.278, Validation loss: 0.5570702692493796
Epoch 21: Train err: 0.226125, Train loss: 0.45988086342811585 |
Validation err: 0.2855, Validation loss: 0.5718803629279137
Epoch 22: Train err: 0.211, Train loss: 0.44206266045570375 |
Validation err: 0.286, Validation loss: 0.5945248389616609
Epoch 23: Train err: 0.211875, Train loss: 0.44135858583450316 |
Validation err: 0.3035, Validation loss: 0.5813650684431195
Epoch 24: Train err: 0.19825, Train loss: 0.41994793343544007 |
Validation err: 0.2835, Validation loss: 0.5980898588895798
Epoch 25: Train err: 0.190375, Train loss: 0.4071806137561798 |
Validation err: 0.3075, Validation loss: 0.6136367982253432
Epoch 26: Train err: 0.17625, Train loss: 0.3912024952173233 |
Validation err: 0.288, Validation loss: 0.627051935531199
Epoch 27: Train err: 0.176, Train loss: 0.38486890816688535 |
Validation err: 0.2995, Validation loss: 0.6231399439275265
Epoch 28: Train err: 0.178875, Train loss: 0.3806236255168915 |
Validation err: 0.2855, Validation loss: 0.6279410840943456
Epoch 29: Train err: 0.17325, Train loss: 0.3742794303894043 |
Validation err: 0.3015, Validation loss: 0.6410087505355477
Epoch 30: Train err: 0.163625, Train loss: 0.35803296029567716 |
Validation err: 0.3, Validation loss: 0.665555571205914
Finished Training
Total time elapsed: 156.93 seconds
```

Part (e) - 2pt

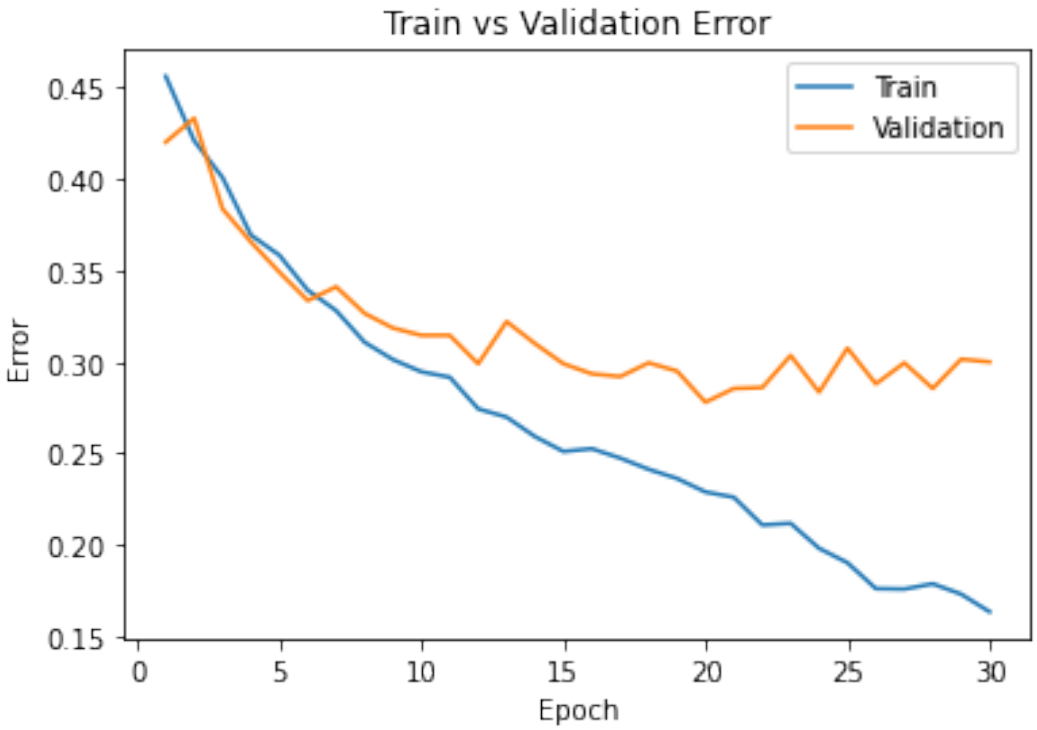
Use the function `plot_training_curve` to display the trajectory of the training/validation error and the training/validation loss. You will need to use the function `get_model_name` to generate the argument to the `plot_training_curve` function.

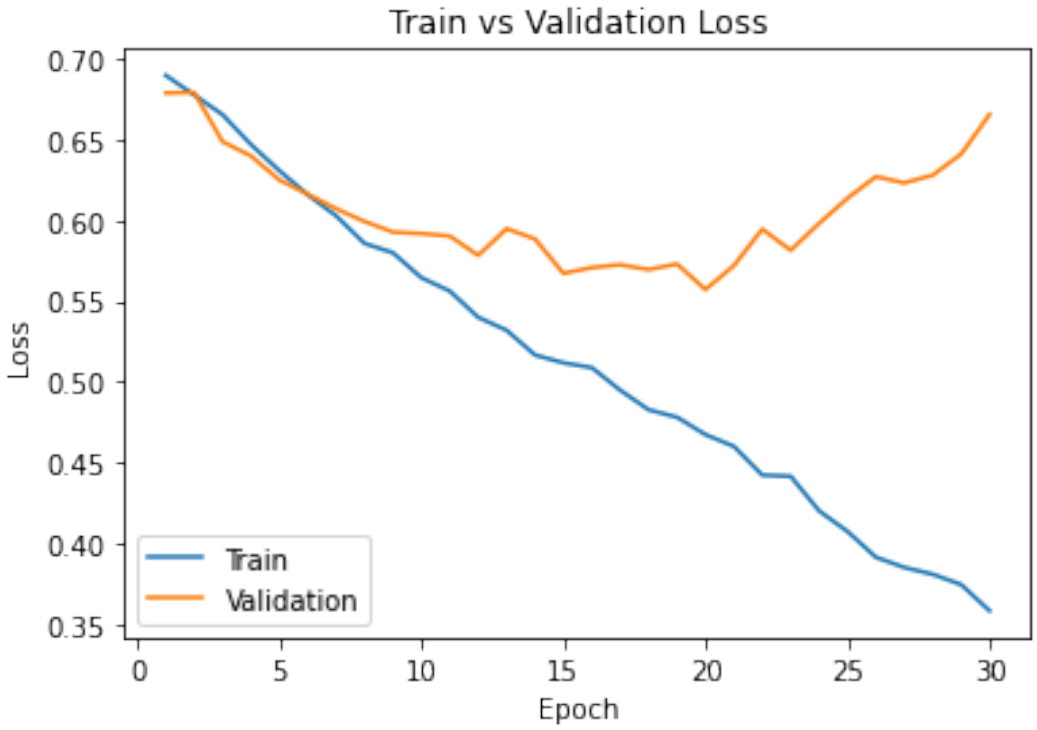
Do this for both the small network and the large network. Include both plots in your writeup.

```
print("Training Curve for Small Network:")
model_path_S = get_model_name("small", batch_size=64,
learning_rate=0.01, epoch=29)
plot_training_curve(model_path_S)
print("Training Curve for large Network:")
model_path_L = get_model_name("large", batch_size=64,
learning_rate=0.01, epoch=29)
plot_training_curve(model_path_L)
```

```
Training Curve for Small Network:
```

Training Curve for large Network:





Part (f) - 5pt

Describe what you notice about the training curve. How do the curves differ for `small_net` and `large_net`? Identify any occurences of underfitting and overfitting.

```
# For the Small NN, we seem to get some underfitting at the start,
# the error and loss both goes down with each additional epoch
# for both training and validation. Eventually, we do see some
plateauing but no
# overfitting yet.

# But for the Large NN, though we seem to see some underfitting at the
beginning
# and we see the training error and loss also going down with each
additional epoch,
# we unfortunately see that for validation, error eventually does
plateau,
# and the loss actually seems to increases after around epoch 20,
# which is a sign of overfitting to the training set.

# Overall, the large net gets a lower final training and validation
error and loss.
```

Part 3. Optimization Parameters [12 pt]

For this section, we will work with `large_net` only.

Part (a) - 3pt

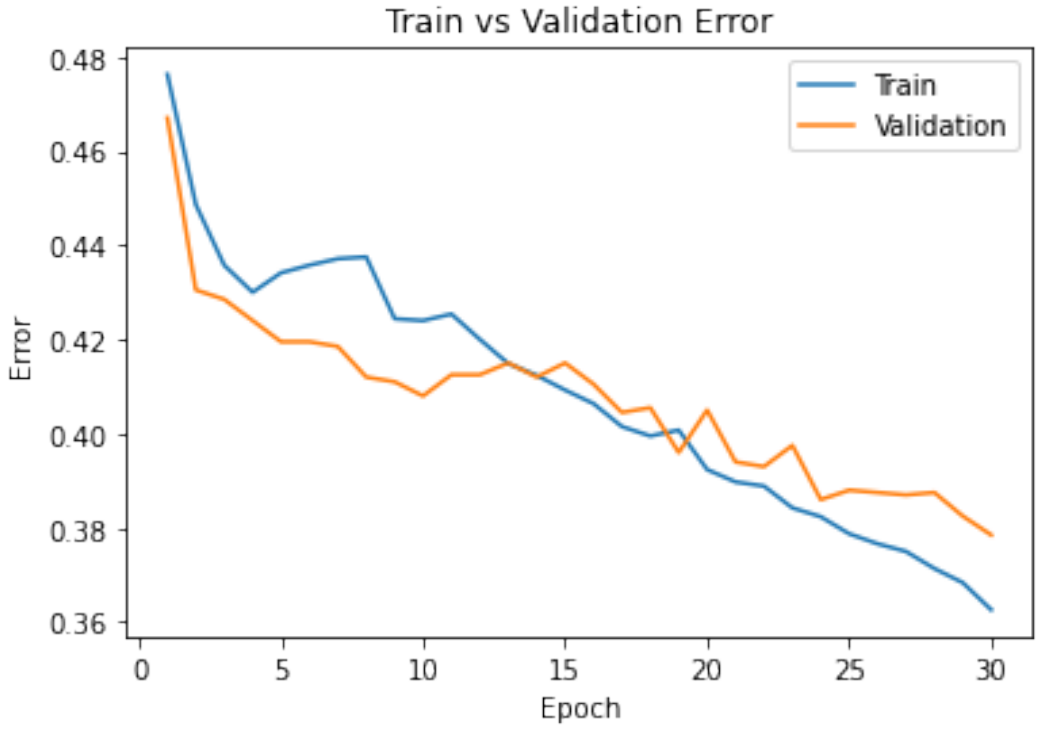
Train `large_net` with all default parameters, except set `learning_rate=0.001`. Does the model take longer/shorter to train? Plot the training curve. Describe the effect of *lowering* the learning rate.

```
# Note: When we re-construct the model, we start the training
# with *random weights*. If we omit this code, the values of
# the weights will still be the previously trained values.
large_net = LargeNet()
train_net(large_net, 64, 0.001, 30)
model_path_large = get_model_name("large", batch_size=64,
learning_rate=0.001, epoch=29)
plot_training_curve(model_path_large)

# The model takes around the same time to train since the learning
rate mainly
# affects the amount at which the models adjusts its weights.
#
# Lowering the learning rate results in higher error and higher loss
in
# training data compared to a lr of 0.01. In validation, the error is
higher,
# but the loss shows no signs of overfitting.
# Essentially, 0.001 seems too slow for training the model.
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.47625, Train loss: 0.6928360013961792 |
```

Validation err: 0.467, Validation loss: 0.6924686580896378
Epoch 2: Train err: 0.448625, Train loss: 0.6922589712142945 |
Validation err: 0.4305, Validation loss: 0.691649341955781
Epoch 3: Train err: 0.43575, Train loss: 0.6916067280769348 |
Validation err: 0.4285, Validation loss: 0.690854424610734
Epoch 4: Train err: 0.43, Train loss: 0.690861343383789 |Validation
err: 0.424, Validation loss: 0.6896595880389214
Epoch 5: Train err: 0.434125, Train loss: 0.6899195008277893 |
Validation err: 0.4195, Validation loss: 0.6886935643851757
Epoch 6: Train err: 0.43575, Train loss: 0.6887411961555481 |
Validation err: 0.4195, Validation loss: 0.6867824867367744
Epoch 7: Train err: 0.437125, Train loss: 0.6873774147033691 |
Validation err: 0.4185, Validation loss: 0.6851982977241278
Epoch 8: Train err: 0.4375, Train loss: 0.6859278454780579 |Validation
err: 0.412, Validation loss: 0.683199780061841
Epoch 9: Train err: 0.424375, Train loss: 0.6844058036804199 |
Validation err: 0.411, Validation loss: 0.6808880660682917
Epoch 10: Train err: 0.424, Train loss: 0.6828502931594849 |Validation
err: 0.408, Validation loss: 0.6783502567559481
Epoch 11: Train err: 0.425375, Train loss: 0.6812348766326904 |
Validation err: 0.4125, Validation loss: 0.6780214440077543
Epoch 12: Train err: 0.42, Train loss: 0.6796319708824158 |Validation
err: 0.4125, Validation loss: 0.6753159202635288
Epoch 13: Train err: 0.414875, Train loss: 0.6777918744087219 |
Validation err: 0.415, Validation loss: 0.6757059413939714
Epoch 14: Train err: 0.412375, Train loss: 0.6761112003326416 |
Validation err: 0.412, Validation loss: 0.6739734839648008
Epoch 15: Train err: 0.40925, Train loss: 0.674472680568695 |
Validation err: 0.415, Validation loss: 0.6706762500107288
Epoch 16: Train err: 0.406375, Train loss: 0.6727448840141297 |
Validation err: 0.4105, Validation loss: 0.6707733049988747
Epoch 17: Train err: 0.4015, Train loss: 0.6713076601028443 |
Validation err: 0.4045, Validation loss: 0.6671545393764973
Epoch 18: Train err: 0.3995, Train loss: 0.6696742882728577 |
Validation err: 0.4055, Validation loss: 0.6646782550960779
Epoch 19: Train err: 0.40075, Train loss: 0.6679086356163025 |
Validation err: 0.396, Validation loss: 0.6655019577592611
Epoch 20: Train err: 0.392375, Train loss: 0.665787980556488 |
Validation err: 0.405, Validation loss: 0.6626011095941067
Epoch 21: Train err: 0.38975, Train loss: 0.6646300601959229 |
Validation err: 0.394, Validation loss: 0.660687854513526
Epoch 22: Train err: 0.388875, Train loss: 0.662373058795929 |
Validation err: 0.393, Validation loss: 0.6616998575627804
Epoch 23: Train err: 0.38425, Train loss: 0.6601516346931458 |
Validation err: 0.3975, Validation loss: 0.6573981791734695
Epoch 24: Train err: 0.382375, Train loss: 0.6584009389877319 |
Validation err: 0.386, Validation loss: 0.6561364810913801
Epoch 25: Train err: 0.37875, Train loss: 0.6554971766471863 |
Validation err: 0.388, Validation loss: 0.6552744228392839
Epoch 26: Train err: 0.376625, Train loss: 0.6531173253059387 |

```
Validation err: 0.3875, Validation loss: 0.6531743723899126
Epoch 27: Train err: 0.375, Train loss: 0.6503696331977844 |Validation
err: 0.387, Validation loss: 0.6519789285957813
Epoch 28: Train err: 0.371375, Train loss: 0.6476435809135437 |
Validation err: 0.3875, Validation loss: 0.6483502741903067
Epoch 29: Train err: 0.368375, Train loss: 0.6451257643699646 |
Validation err: 0.3825, Validation loss: 0.6459067314863205
Epoch 30: Train err: 0.362625, Train loss: 0.6423329524993896 |
Validation err: 0.3785, Validation loss: 0.6439237017184496
Finished Training
Total time elapsed: 147.38 seconds
```



Part (b) - 3pt

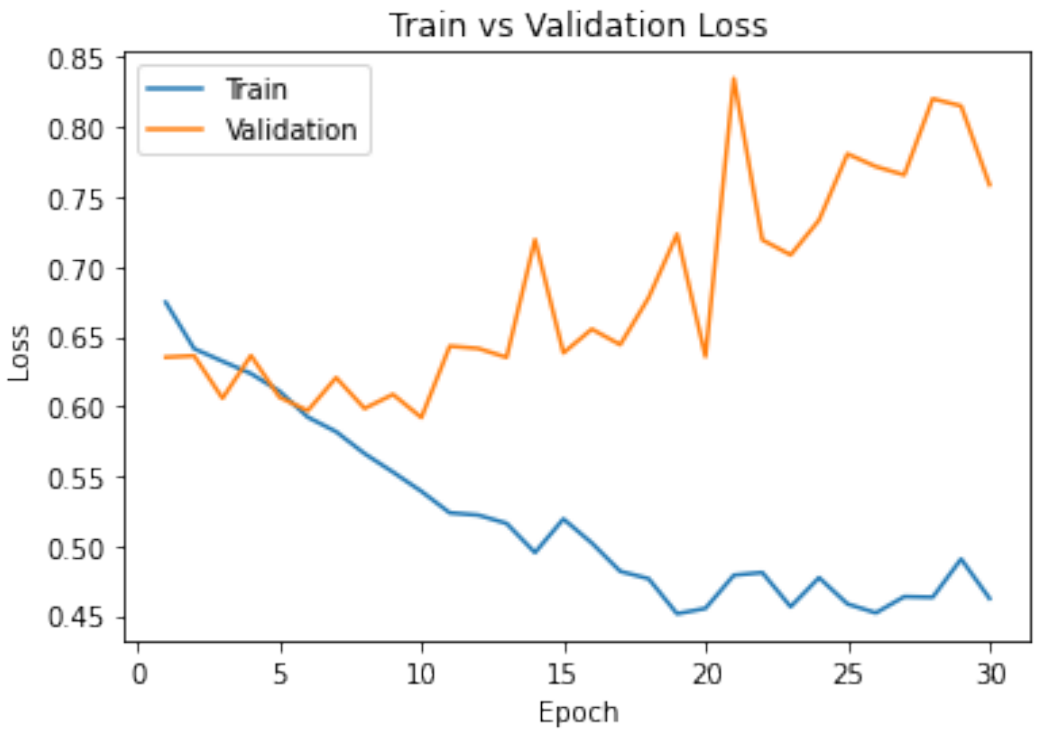
Train `large_net` with all default parameters, except set `learning_rate=0.1`. Does the model take longer/shorter to train? Plot the training curve. Describe the effect of *increasing* the learning rate.

```
large_net = LargeNet()
train_net(large_net, 64, 0.1, 30)
model_path_large = get_model_name("large", batch_size=64,
learning_rate=0.1, epoch=29)
plot_training_curve(model_path_large)

# As before, the model takes around the same time to train. Since the
learning rate mainly
# affects the amount at which the models adjusts its weights.
#
# Increasing the learning rate makes the model get higher error
# and higher loss in both the training and validation set,
# The Validation error and loss increases quickly and jump around
after around epoch 15,
# which is a sign of overfitting.
# A learning rate of 0.1 is too fast in this case.
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.4295, Train loss: 0.67437779712677 |Validation
err: 0.3595, Validation loss: 0.6350857093930244
Epoch 2: Train err: 0.36075, Train loss: 0.6411805458068848 |
```

Validation err: 0.3535, Validation loss: 0.6361209936439991
Epoch 3: Train err: 0.365125, Train loss: 0.6321813461780548 |
Validation err: 0.3385, Validation loss: 0.6056603882461786
Epoch 4: Train err: 0.352625, Train loss: 0.6233456182479858 |
Validation err: 0.3575, Validation loss: 0.6362800188362598
Epoch 5: Train err: 0.34075, Train loss: 0.6108013873100281 |
Validation err: 0.3305, Validation loss: 0.6064918786287308
Epoch 6: Train err: 0.323375, Train loss: 0.5921835997104645 |
Validation err: 0.317, Validation loss: 0.5967769594863057
Epoch 7: Train err: 0.3145, Train loss: 0.5817317583560944 |Validation
err: 0.3365, Validation loss: 0.6204487886279821
Epoch 8: Train err: 0.29825, Train loss: 0.5660300073623658 |
Validation err: 0.3285, Validation loss: 0.5983372200280428
Epoch 9: Train err: 0.290875, Train loss: 0.552809501171112 |
Validation err: 0.3315, Validation loss: 0.6084455158561468
Epoch 10: Train err: 0.278625, Train loss: 0.539032607793808 |
Validation err: 0.306, Validation loss: 0.5918631898239255
Epoch 11: Train err: 0.272375, Train loss: 0.5236025826931 |Validation
err: 0.33, Validation loss: 0.6430060230195522
Epoch 12: Train err: 0.267375, Train loss: 0.5220149435997009 |
Validation err: 0.2925, Validation loss: 0.6413561534136534
Epoch 13: Train err: 0.266, Train loss: 0.5160510110855102 |Validation
err: 0.3125, Validation loss: 0.6349832843989134
Epoch 14: Train err: 0.24875, Train loss: 0.4951590054035187 |
Validation err: 0.3145, Validation loss: 0.7193072671070695
Epoch 15: Train err: 0.264625, Train loss: 0.519231944322586 |
Validation err: 0.314, Validation loss: 0.6381420725956559
Epoch 16: Train err: 0.252625, Train loss: 0.5020012385845184 |
Validation err: 0.3225, Validation loss: 0.6551959458738565
Epoch 17: Train err: 0.23875, Train loss: 0.481714787364006 |
Validation err: 0.357, Validation loss: 0.6440742611885071
Epoch 18: Train err: 0.23375, Train loss: 0.47645506453514097 |
Validation err: 0.3375, Validation loss: 0.6777342790737748
Epoch 19: Train err: 0.218125, Train loss: 0.45134368968009947 |
Validation err: 0.3445, Validation loss: 0.7232250478118658
Epoch 20: Train err: 0.217875, Train loss: 0.45516350817680357 |
Validation err: 0.3245, Validation loss: 0.6354950983077288
Epoch 21: Train err: 0.23275, Train loss: 0.47897080445289614 |
Validation err: 0.3255, Validation loss: 0.8348110988736153
Epoch 22: Train err: 0.234875, Train loss: 0.4808810565471649 |
Validation err: 0.334, Validation loss: 0.7191346418112516
Epoch 23: Train err: 0.21575, Train loss: 0.4563647754192352 |
Validation err: 0.316, Validation loss: 0.7083508176729083
Epoch 24: Train err: 0.2355, Train loss: 0.47718250966072084 |
Validation err: 0.327, Validation loss: 0.7333047650754452
Epoch 25: Train err: 0.22025, Train loss: 0.4583414270877838 |
Validation err: 0.3315, Validation loss: 0.7806987538933754
Epoch 26: Train err: 0.209625, Train loss: 0.4519626965522766 |
Validation err: 0.3435, Validation loss: 0.7715998776257038
Epoch 27: Train err: 0.22175, Train loss: 0.4636160457134247 |

```
Validation err: 0.3215, Validation loss: 0.7656293725594878
Epoch 28: Train err: 0.219375, Train loss: 0.46314777398109436 |
Validation err: 0.348, Validation loss: 0.8202023077756166
Epoch 29: Train err: 0.235875, Train loss: 0.49053542733192446 |
Validation err: 0.326, Validation loss: 0.8150460105389357
Epoch 30: Train err: 0.22, Train loss: 0.4623157248497009 |Validation
err: 0.3165, Validation loss: 0.7585078496485949
Finished Training
Total time elapsed: 169.31 seconds
```



Part (c) - 3pt

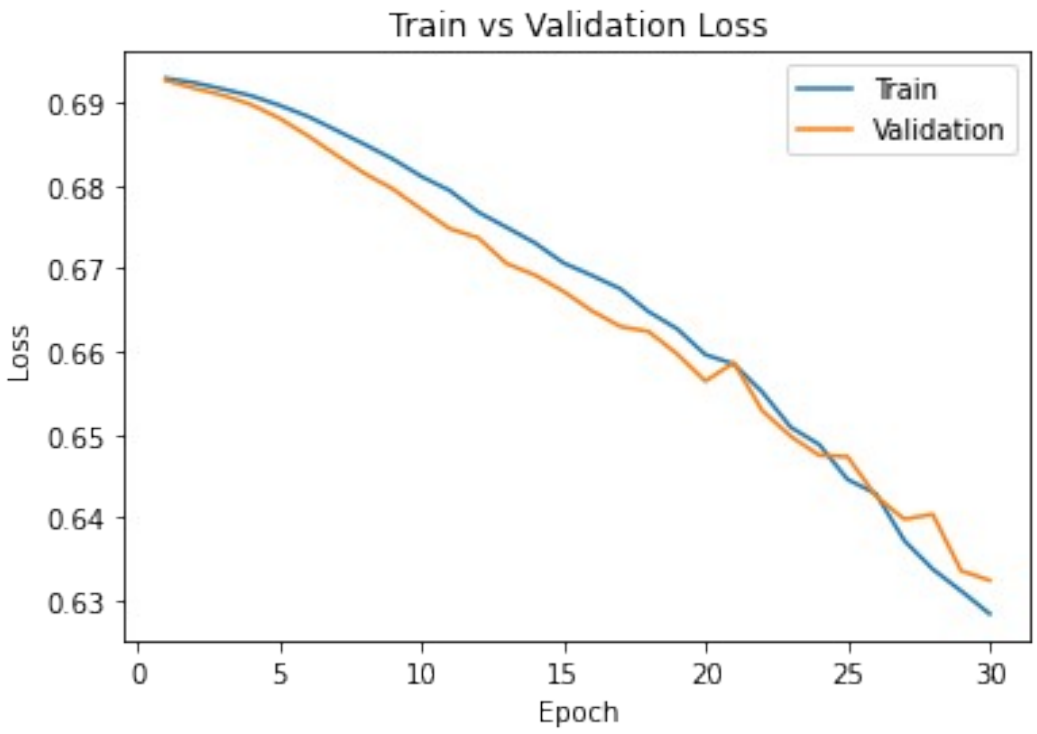
Train `large_net` with all default parameters, including with `learning_rate=0.01`. Now, set `batch_size=512`. Does the model take longer/shorter to train? Plot the training curve. Describe the effect of *increasing* the batch size.

```
large_net = LargeNet()
train_net(large_net, 512, 0.01, 30)
model_path_large = get_model_name("large", batch_size=512,
learning_rate=0.01, epoch=29)
plot_training_curve(model_path_large)

# The model now takes less time to train.
# This is because of the increased batch size which increases the
number of
# training examples used per optimization step/iteration,
# which means that for each epoch, there is a smaller total number of
iteration.
#
# Increasing the batch size makes the model get higher error and
higher loss in
# training data than the default batch size.
# And for validation, the error and loss are around the same value
compared to the default.
#
# Though here, we see no signs of overfitting. But the decreasing loss
of the
```

```
# validation data set is too small.
# This batch_size seems to be a bit too large.
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.48175, Train loss: 0.6929379552602768 |
Validation err: 0.478, Validation loss: 0.6926824003458023
Epoch 2: Train err: 0.457625, Train loss: 0.6924104019999504 |
Validation err: 0.434, Validation loss: 0.6917425245046616
Epoch 3: Train err: 0.437, Train loss: 0.6916500590741634 |Validation
err: 0.4265, Validation loss: 0.6909129917621613
Epoch 4: Train err: 0.433625, Train loss: 0.6908449940383434 |
Validation err: 0.424, Validation loss: 0.6897870451211929
Epoch 5: Train err: 0.434, Train loss: 0.6896935552358627 |Validation
err: 0.424, Validation loss: 0.6881355047225952
Epoch 6: Train err: 0.438, Train loss: 0.688353206962347 |Validation
err: 0.4285, Validation loss: 0.686011865735054
Epoch 7: Train err: 0.439375, Train loss: 0.6866871677339077 |
Validation err: 0.426, Validation loss: 0.6836968809366226
Epoch 8: Train err: 0.43525, Train loss: 0.6849770769476891 |
Validation err: 0.4115, Validation loss: 0.6814671903848648
Epoch 9: Train err: 0.42375, Train loss: 0.6832009293138981 |
Validation err: 0.414, Validation loss: 0.679591491818428
Epoch 10: Train err: 0.421, Train loss: 0.6811089366674423 |Validation
err: 0.416, Validation loss: 0.6771548539400101
Epoch 11: Train err: 0.420875, Train loss: 0.6794026419520378 |
Validation err: 0.4095, Validation loss: 0.6748111099004745
Epoch 12: Train err: 0.41475, Train loss: 0.6768048219382763 |
Validation err: 0.412, Validation loss: 0.6737060546875
Epoch 13: Train err: 0.4105, Train loss: 0.6749702803790569 |
Validation err: 0.412, Validation loss: 0.6706101596355438
Epoch 14: Train err: 0.407125, Train loss: 0.6730880849063396 |
Validation err: 0.4125, Validation loss: 0.6692148000001907
Epoch 15: Train err: 0.4005, Train loss: 0.6706806942820549 |
Validation err: 0.4105, Validation loss: 0.667252704501152
Epoch 16: Train err: 0.397625, Train loss: 0.6691771410405636 |
Validation err: 0.405, Validation loss: 0.6649097055196762
Epoch 17: Train err: 0.393875, Train loss: 0.6675694733858109 |
Validation err: 0.401, Validation loss: 0.6630224883556366
Epoch 18: Train err: 0.393, Train loss: 0.6648042872548103 |Validation
err: 0.3945, Validation loss: 0.6624014377593994
Epoch 19: Train err: 0.38625, Train loss: 0.662746611982584 |
Validation err: 0.388, Validation loss: 0.6597220152616501
Epoch 20: Train err: 0.38175, Train loss: 0.6596181839704514 |
Validation err: 0.4005, Validation loss: 0.6564337313175201
Epoch 21: Train err: 0.38575, Train loss: 0.6584899798035622 |
Validation err: 0.3885, Validation loss: 0.6586423963308334
Epoch 22: Train err: 0.378125, Train loss: 0.655123382806778 |
Validation err: 0.3855, Validation loss: 0.6528600305318832
Epoch 23: Train err: 0.372125, Train loss: 0.6508794128894806 |
```

```
Validation err: 0.3835, Validation loss: 0.6497963815927505
Epoch 24: Train err: 0.37675, Train loss: 0.6488028429448605 |
Validation err: 0.385, Validation loss: 0.6474899500608444
Epoch 25: Train err: 0.368625, Train loss: 0.6445869170129299 |
Validation err: 0.382, Validation loss: 0.6473268568515778
Epoch 26: Train err: 0.372625, Train loss: 0.6428566053509712 |
Validation err: 0.3745, Validation loss: 0.6425703465938568
Epoch 27: Train err: 0.359375, Train loss: 0.6372117549180984 |
Validation err: 0.379, Validation loss: 0.6397799849510193
Epoch 28: Train err: 0.35425, Train loss: 0.6337667480111122 |
Validation err: 0.3695, Validation loss: 0.6403783112764359
Epoch 29: Train err: 0.3535, Train loss: 0.6311353109776974 |
Validation err: 0.366, Validation loss: 0.6335585117340088
Epoch 30: Train err: 0.353, Train loss: 0.6283832415938377 |Validation
err: 0.3675, Validation loss: 0.6324127316474915
Finished Training
Total time elapsed: 130.28 seconds
```



Part (d) - 3pt

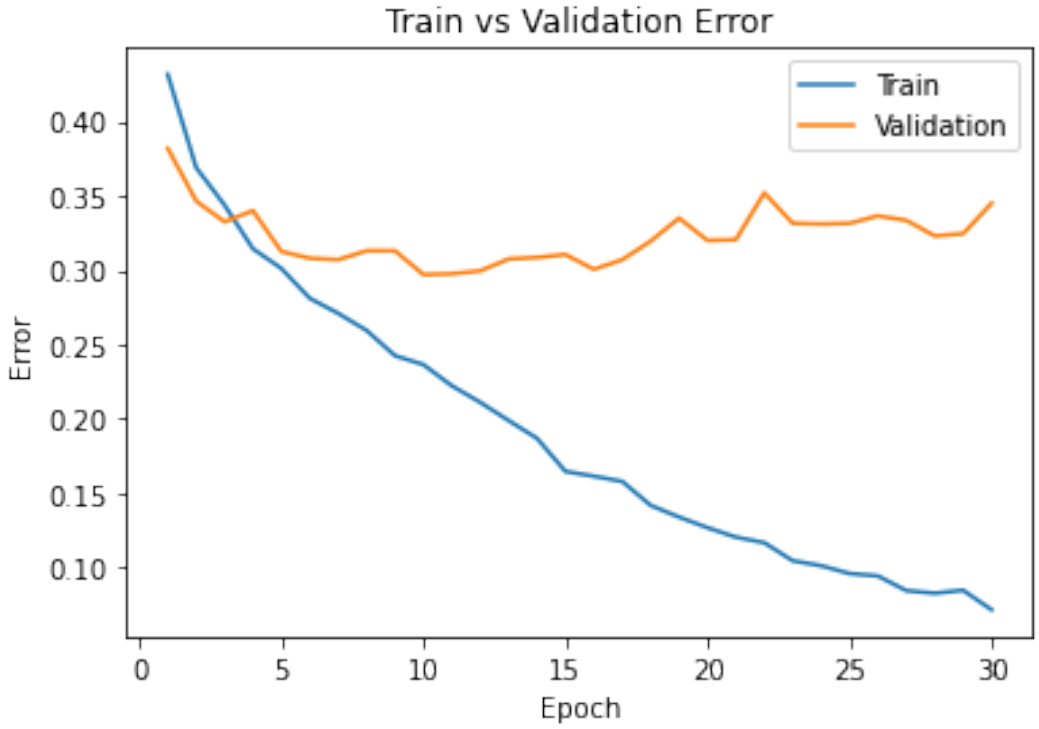
Train `large_net` with all default parameters, including with `learning_rate=0.01`. Now, set `batch_size=16`. Does the model take longer/shorter to train? Plot the training curve. Describe the effect of *decreasing* the batch size.

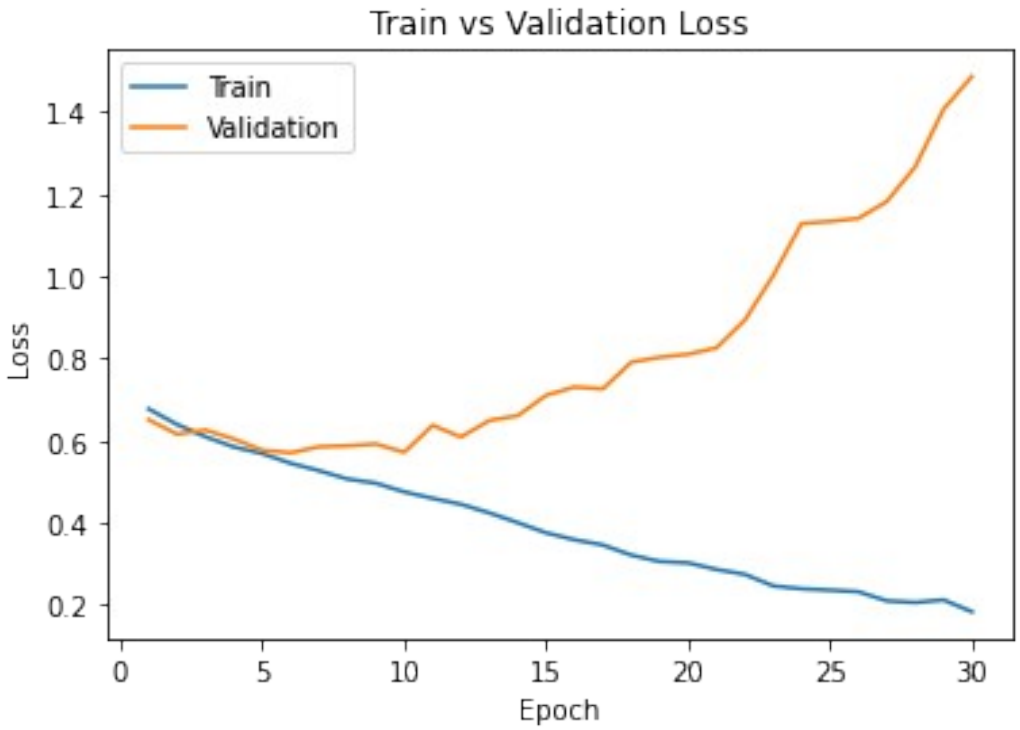
```
large_net = LargeNet()
train_net(large_net, 16, 0.01, 30)
model_path_large = get_model_name("large", batch_size=16,
learning_rate=0.01, epoch=29)
plot_training_curve(model_path_large)

# With the decreased batch_size, the model now takes longer to train
than the default batch_size.
# This is due to the increased number of iterations resulting in each
epoch taking longer.
#
# With a lower batch_size, we see a lower error and lower loss with
the training data set.
# But, with the validation data set, we see higher error and an
incresing loss with each epoch.
# This is a sign of overfitting likely due to the small batch size
making the model
# more so memorize the model than identify cats/dogs in general.
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.43175, Train loss: 0.6774994022846222 |
```

Validation err: 0.382, Validation loss: 0.6513170118331909
Epoch 2: Train err: 0.369, Train loss: 0.639639899969101 |Validation
err: 0.3465, Validation loss: 0.6161113576889038
Epoch 3: Train err: 0.34375, Train loss: 0.6098222947120666 |
Validation err: 0.3325, Validation loss: 0.6260210764408112
Epoch 4: Train err: 0.314375, Train loss: 0.5849691489338875 |
Validation err: 0.34, Validation loss: 0.6044013917446136
Epoch 5: Train err: 0.301125, Train loss: 0.5689119303822517 |
Validation err: 0.3125, Validation loss: 0.576918310880661
Epoch 6: Train err: 0.281, Train loss: 0.5452213581204415 |Validation
err: 0.308, Validation loss: 0.5708447456359863
Epoch 7: Train err: 0.270875, Train loss: 0.5272981298565864 |
Validation err: 0.307, Validation loss: 0.5854293291568756
Epoch 8: Train err: 0.259375, Train loss: 0.5070905526578426 |
Validation err: 0.313, Validation loss: 0.5877130818367005
Epoch 9: Train err: 0.242375, Train loss: 0.4968344421982765 |
Validation err: 0.313, Validation loss: 0.5922425072193146
Epoch 10: Train err: 0.236375, Train loss: 0.4756101597249508 |
Validation err: 0.297, Validation loss: 0.5718690166473389
Epoch 11: Train err: 0.222125, Train loss: 0.4599769461452961 |
Validation err: 0.2975, Validation loss: 0.6376970833539963
Epoch 12: Train err: 0.211, Train loss: 0.4454492371380329 |Validation
err: 0.2995, Validation loss: 0.609202565908432
Epoch 13: Train err: 0.19875, Train loss: 0.4245421719551086 |
Validation err: 0.3075, Validation loss: 0.6494987765550614
Epoch 14: Train err: 0.18675, Train loss: 0.4007472907453775 |
Validation err: 0.3085, Validation loss: 0.6610016552209854
Epoch 15: Train err: 0.1645, Train loss: 0.3759974058121443 |
Validation err: 0.3105, Validation loss: 0.7106090537309646
Epoch 16: Train err: 0.16125, Train loss: 0.3591455406397581 |
Validation err: 0.3005, Validation loss: 0.7310364942550659
Epoch 17: Train err: 0.15775, Train loss: 0.3463234790861607 |
Validation err: 0.307, Validation loss: 0.7263009325265884
Epoch 18: Train err: 0.141625, Train loss: 0.32175366275012496 |
Validation err: 0.3195, Validation loss: 0.7913952842950821
Epoch 19: Train err: 0.13375, Train loss: 0.30618105667084455 |
Validation err: 0.335, Validation loss: 0.8032052783966065
Epoch 20: Train err: 0.126625, Train loss: 0.3029071792438626 |
Validation err: 0.32, Validation loss: 0.8106685240268707
Epoch 21: Train err: 0.12025, Train loss: 0.28682796490937473 |
Validation err: 0.3205, Validation loss: 0.8259474284648896
Epoch 22: Train err: 0.1165, Train loss: 0.27489088076353074 |
Validation err: 0.352, Validation loss: 0.8937610774040222
Epoch 23: Train err: 0.104375, Train loss: 0.2467898527495563 |
Validation err: 0.3315, Validation loss: 1.0021928198337555
Epoch 24: Train err: 0.101, Train loss: 0.23970085787773132 |
Validation err: 0.331, Validation loss: 1.1290796399116516
Epoch 25: Train err: 0.09575, Train loss: 0.23643119425699116 |
Validation err: 0.3315, Validation loss: 1.1338514368534087
Epoch 26: Train err: 0.094125, Train loss: 0.2325953512713313 |

```
Validation err: 0.3365, Validation loss: 1.1414263204336166
Epoch 27: Train err: 0.08425, Train loss: 0.21040759468451142 |
Validation err: 0.3335, Validation loss: 1.1823678107261657
Epoch 28: Train err: 0.0825, Train loss: 0.20643112615589052 |
Validation err: 0.323, Validation loss: 1.266836181640625
Epoch 29: Train err: 0.0845, Train loss: 0.21273409337876364 |
Validation err: 0.3245, Validation loss: 1.406717705130577
Epoch 30: Train err: 0.071375, Train loss: 0.18387044295761734 |
Validation err: 0.345, Validation loss: 1.4871552000045776
Finished Training
Total time elapsed: 200.81 seconds
```





Part 4. Hyperparameter Search [6 pt]

Part (a) - 2pt

Based on the plots from above, choose another set of values for the hyperparameters (network, batch_size, learning_rate) that you think would help you improve the validation accuracy. Justify your choice.

```
# My chosen parameters:
# network = large_net
# batch_size = 256
# learning_rate = 0.01
# epoch = 30

# From the initial test, we saw that the large NN had a smaller error
and loss
# at the start (before the large NN started to overfit);

# From testing the lr, it seems like 0.01 is a good learning rate,
going to far in
# increasing or decreasing the lr has negative impacts on training the
model,
# there might be a better lr close to this like 0.015, but that's a
future concern;

# From testing different batch sizes, when it is small, the model tend
to overfitting,
```

```
# and when it is large, we see no sign of overfitting but the loss
decreases too slowly.
# Therefore, a batch_size of 256 (between 64 and 512) may be a good
value.

# The default number of epoch of 30 seems good. We see the loss
flattening around here.
# If it is too small, we will see underfitting,
# and uf it is too large, we will see overfitting.
```

Part (b) - 1pt

Train the model with the hyperparameters you chose in part(a), and include the training curve.

```
large_net = LargeNet()
train_net(large_net, 256, 0.01, 30)
model_path_large = get_model_name("large", batch_size=256,
learning_rate=0.01, epoch=29)
plot_training_curve(model_path_large)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.504, Train loss: 0.694011740386486 |Validation
err: 0.4715, Validation loss: 0.6925566866993904
Epoch 2: Train err: 0.455375, Train loss: 0.6913719363510609 |
Validation err: 0.424, Validation loss: 0.6898768916726112
Epoch 3: Train err: 0.435375, Train loss: 0.6885282509028912 |
Validation err: 0.4205, Validation loss: 0.686040386557579
Epoch 4: Train err: 0.424625, Train loss: 0.6835071798413992 |
Validation err: 0.4185, Validation loss: 0.6799078956246376
Epoch 5: Train err: 0.421125, Train loss: 0.6772986091673374 |
Validation err: 0.409, Validation loss: 0.6730506420135498
Epoch 6: Train err: 0.40675, Train loss: 0.6715615149587393 |
Validation err: 0.4015, Validation loss: 0.6689667105674744
Epoch 7: Train err: 0.388125, Train loss: 0.6626836936920881 |
Validation err: 0.379, Validation loss: 0.6593648418784142
Epoch 8: Train err: 0.385125, Train loss: 0.6555437333881855 |
Validation err: 0.3815, Validation loss: 0.6485779136419296
Epoch 9: Train err: 0.363875, Train loss: 0.6445739157497883 |
Validation err: 0.36, Validation loss: 0.6355830579996109
Epoch 10: Train err: 0.35175, Train loss: 0.6363527402281761 |
Validation err: 0.3395, Validation loss: 0.6283280998468399
Epoch 11: Train err: 0.347, Train loss: 0.6289401836693287 |Validation
err: 0.3355, Validation loss: 0.620981827378273
Epoch 12: Train err: 0.339, Train loss: 0.6198319401592016 |Validation
err: 0.3345, Validation loss: 0.6154309138655663
Epoch 13: Train err: 0.332375, Train loss: 0.6165138408541679 |
Validation err: 0.3285, Validation loss: 0.612803727388382
Epoch 14: Train err: 0.330625, Train loss: 0.6104299630969763 |
Validation err: 0.3425, Validation loss: 0.6208969876170158
```

```
Epoch 15: Train err: 0.323, Train loss: 0.6036981232464314 |Validation
err: 0.3435, Validation loss: 0.620563767850399
Epoch 16: Train err: 0.319, Train loss: 0.5967491716146469 |Validation
err: 0.3265, Validation loss: 0.6030048504471779
Epoch 17: Train err: 0.318125, Train loss: 0.6008494906127453 |
Validation err: 0.3325, Validation loss: 0.6110973507165909
Epoch 18: Train err: 0.312625, Train loss: 0.5925734825432301 |
Validation err: 0.3185, Validation loss: 0.5979076772928238
Epoch 19: Train err: 0.314875, Train loss: 0.5893678311258554 |
Validation err: 0.3305, Validation loss: 0.6031572222709656
Epoch 20: Train err: 0.305, Train loss: 0.5796078033745289 |Validation
err: 0.316, Validation loss: 0.5931820049881935
Epoch 21: Train err: 0.293375, Train loss: 0.573655879124999 |
Validation err: 0.3135, Validation loss: 0.5982393845915794
Epoch 22: Train err: 0.29625, Train loss: 0.5721764508634806 |
Validation err: 0.326, Validation loss: 0.5985486432909966
Epoch 23: Train err: 0.293, Train loss: 0.5688072871416807 |Validation
err: 0.324, Validation loss: 0.6017276346683502
Epoch 24: Train err: 0.301, Train loss: 0.5709631387144327 |Validation
err: 0.323, Validation loss: 0.596871092915535
Epoch 25: Train err: 0.29075, Train loss: 0.5617755651473999 |
Validation err: 0.3095, Validation loss: 0.5894144028425217
Epoch 26: Train err: 0.28425, Train loss: 0.558282951824367 |
Validation err: 0.323, Validation loss: 0.6019551381468773
Epoch 27: Train err: 0.275625, Train loss: 0.5510303908959031 |
Validation err: 0.31, Validation loss: 0.5966653749346733
Epoch 28: Train err: 0.280875, Train loss: 0.5498981112614274 |
Validation err: 0.31, Validation loss: 0.5914952084422112
Epoch 29: Train err: 0.28275, Train loss: 0.5559950321912766 |
Validation err: 0.307, Validation loss: 0.5860651507973671
Epoch 30: Train err: 0.28575, Train loss: 0.5526535408571362 |
Validation err: 0.3155, Validation loss: 0.592315748333931
Finished Training
Total time elapsed: 119.14 seconds
```

Part (c) - 2pt

Based on your result from Part(a), suggest another set of hyperparameter values to try. Justify your choice.

```
# My new chosen parameters:
# network = large_net
# batch_size = 128
# learning_rate = 0.012
# epoch = 30

# I have increased the learning rate a bit as we see no signs of over
fitting
# I have also reduced the batch size to hopefully increase the
decrease in loss.
```

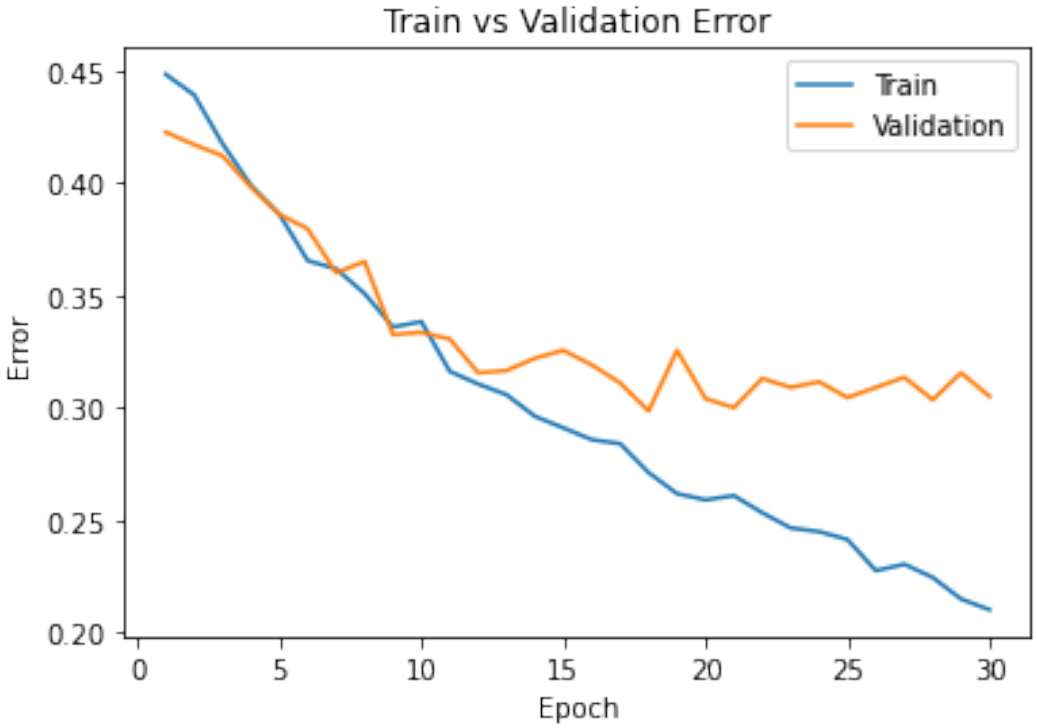
Part (d) - 1pt

Train the model with the hyperparameters you chose in part(c), and include the training curve.

```
large_net = LargeNet()
train_net(large_net, 128, 0.012, 30)
model_path_large = get_model_name("large", batch_size=128,
learning_rate=0.012, epoch=29)
plot_training_curve(model_path_large)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.44825, Train loss: 0.691808421460409 |Validation
err: 0.4225, Validation loss: 0.6888818927109241
Epoch 2: Train err: 0.439125, Train loss: 0.6863207656239706 |
Validation err: 0.417, Validation loss: 0.6800182536244392
Epoch 3: Train err: 0.417375, Train loss: 0.678412399594746 |
Validation err: 0.412, Validation loss: 0.6695855595171452
Epoch 4: Train err: 0.398875, Train loss: 0.6697994820655339 |
Validation err: 0.398, Validation loss: 0.6620188914239407
Epoch 5: Train err: 0.386, Train loss: 0.6596346279931447 |Validation
err: 0.386, Validation loss: 0.6511647067964077
Epoch 6: Train err: 0.365375, Train loss: 0.6462826142235408 |
Validation err: 0.3795, Validation loss: 0.6402678675949574
Epoch 7: Train err: 0.36175, Train loss: 0.6377827052086119 |
Validation err: 0.36, Validation loss: 0.6300922073423862
Epoch 8: Train err: 0.35075, Train loss: 0.6254830795621115 |
Validation err: 0.365, Validation loss: 0.6356193237006664
Epoch 9: Train err: 0.335875, Train loss: 0.6141640127651276 |
Validation err: 0.3325, Validation loss: 0.6121256947517395
Epoch 10: Train err: 0.33825, Train loss: 0.6082990226291475 |
Validation err: 0.3335, Validation loss: 0.610305842012167
Epoch 11: Train err: 0.316125, Train loss: 0.5949523638165186 |
Validation err: 0.3305, Validation loss: 0.605425339192152
Epoch 12: Train err: 0.3105, Train loss: 0.5838511207747081 |
Validation err: 0.3155, Validation loss: 0.594910841435194
Epoch 13: Train err: 0.30575, Train loss: 0.5779793792300754 |
Validation err: 0.3165, Validation loss: 0.5980239100754261
Epoch 14: Train err: 0.296125, Train loss: 0.5648010510300833 |
```

```
Validation err: 0.322, Validation loss: 0.613241758197546
Epoch 15: Train err: 0.291, Train loss: 0.556833269104125 |Validation
err: 0.3255, Validation loss: 0.603080365806818
Epoch 16: Train err: 0.28575, Train loss: 0.5511208914575123 |
Validation err: 0.319, Validation loss: 0.5969000086188316
Epoch 17: Train err: 0.284, Train loss: 0.544311159186893 |Validation
err: 0.311, Validation loss: 0.5950912162661552
Epoch 18: Train err: 0.271125, Train loss: 0.5337979329956902 |
Validation err: 0.2985, Validation loss: 0.5854161456227303
Epoch 19: Train err: 0.26175, Train loss: 0.5197710111027672 |
Validation err: 0.3255, Validation loss: 0.5973407663404942
Epoch 20: Train err: 0.259, Train loss: 0.5155839560523866 |Validation
err: 0.304, Validation loss: 0.6008226126432419
Epoch 21: Train err: 0.260875, Train loss: 0.512496816260474 |
Validation err: 0.3, Validation loss: 0.5795645043253899
Epoch 22: Train err: 0.25325, Train loss: 0.5056628083425855 |
Validation err: 0.313, Validation loss: 0.5890360623598099
Epoch 23: Train err: 0.2465, Train loss: 0.4974015989000835 |
Validation err: 0.309, Validation loss: 0.5844534039497375
Epoch 24: Train err: 0.244875, Train loss: 0.4938811172568609 |
Validation err: 0.3115, Validation loss: 0.6211344748735428
Epoch 25: Train err: 0.241375, Train loss: 0.4855541899090722 |
Validation err: 0.3045, Validation loss: 0.5943238269537687
Epoch 26: Train err: 0.2275, Train loss: 0.47125892033652655 |
Validation err: 0.309, Validation loss: 0.5895782150328159
Epoch 27: Train err: 0.230375, Train loss: 0.4732734972522372 |
Validation err: 0.3135, Validation loss: 0.6030163243412971
Epoch 28: Train err: 0.2245, Train loss: 0.45975668515477863 |
Validation err: 0.3035, Validation loss: 0.6019302234053612
Epoch 29: Train err: 0.214875, Train loss: 0.4526614717074803 |
Validation err: 0.3155, Validation loss: 0.6146877594292164
Epoch 30: Train err: 0.210125, Train loss: 0.442081931564543 |
Validation err: 0.305, Validation loss: 0.6026543043553829
Finished Training
Total time elapsed: 151.26 seconds
```


Train vs Validation Error
Train
Validation
Error
Epoch
0.45
0.40
0.35
0.30
0.25
0.20
0
5
10
15
20
25
30


Train vs Validation Loss
Train
Validation
Loss
Epoch
0.70
0.65
0.60
0.55
0.50
0.45
0
5
10
15
20
25
30

Part 4. Evaluating the Best Model [15 pt]

Part (a) - 1pt

Choose the **best** model that you have so far. This means choosing the best model checkpoint, including the choice of `small_net` vs `large_net`, the `batch_size`, `learning_rate`, **and the epoch number**.

Modify the code below to load your chosen set of weights to the model object `net`.

```
net = LargeNet()
train_net(net, 128, 0.012, 21)
model_path = get_model_name("large", batch_size=128,
learning_rate=0.012, epoch=20)
state = torch.load(model_path)
net.load_state_dict(state)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.44825, Train loss: 0.691808421460409 |Validation
err: 0.4225, Validation loss: 0.6888818927109241
Epoch 2: Train err: 0.439125, Train loss: 0.6863207656239706 |
Validation err: 0.417, Validation loss: 0.6800182536244392
Epoch 3: Train err: 0.417375, Train loss: 0.678412399594746 |
Validation err: 0.412, Validation loss: 0.6695855595171452
Epoch 4: Train err: 0.398875, Train loss: 0.6697994820655339 |
Validation err: 0.398, Validation loss: 0.6620188914239407
Epoch 5: Train err: 0.386, Train loss: 0.6596346279931447 |Validation
err: 0.386, Validation loss: 0.6511647067964077
Epoch 6: Train err: 0.365375, Train loss: 0.6462826142235408 |
Validation err: 0.3795, Validation loss: 0.6402678675949574
Epoch 7: Train err: 0.36175, Train loss: 0.6377827052086119 |
Validation err: 0.36, Validation loss: 0.6300922073423862
Epoch 8: Train err: 0.35075, Train loss: 0.6254830795621115 |
Validation err: 0.365, Validation loss: 0.6356193237006664
Epoch 9: Train err: 0.335875, Train loss: 0.6141640127651276 |
Validation err: 0.3325, Validation loss: 0.6121256947517395
Epoch 10: Train err: 0.33825, Train loss: 0.6082990226291475 |
Validation err: 0.3335, Validation loss: 0.610305842012167
Epoch 11: Train err: 0.316125, Train loss: 0.5949523638165186 |
Validation err: 0.3305, Validation loss: 0.605425339192152
Epoch 12: Train err: 0.3105, Train loss: 0.5838511207747081 |
Validation err: 0.3155, Validation loss: 0.594910841435194
Epoch 13: Train err: 0.30575, Train loss: 0.5779793792300754 |
Validation err: 0.3165, Validation loss: 0.5980239100754261
Epoch 14: Train err: 0.296125, Train loss: 0.5648010510300833 |
Validation err: 0.322, Validation loss: 0.613241758197546
Epoch 15: Train err: 0.291, Train loss: 0.556833269104125 |Validation
err: 0.3255, Validation loss: 0.603080365806818
Epoch 16: Train err: 0.28575, Train loss: 0.5511208914575123 |
Validation err: 0.319, Validation loss: 0.5969000086188316
```

```
Epoch 17: Train err: 0.284, Train loss: 0.544311159186893 |Validation
err: 0.311, Validation loss: 0.5950912162661552
Epoch 18: Train err: 0.271125, Train loss: 0.5337979329956902 |
Validation err: 0.2985, Validation loss: 0.5854161456227303
Epoch 19: Train err: 0.26175, Train loss: 0.5197710111027672 |
Validation err: 0.3255, Validation loss: 0.5973407663404942
Epoch 20: Train err: 0.259, Train loss: 0.5155839560523866 |Validation
err: 0.304, Validation loss: 0.6008226126432419
Epoch 21: Train err: 0.260875, Train loss: 0.512496816260474 |
Validation err: 0.3, Validation loss: 0.5795645043253899
Finished Training
Total time elapsed: 107.30 seconds

<All keys matched successfully>
```

Part (b) - 2pt

Justify your choice of model from part (a).

```
# From my latest model, I see the lowest validation error and loss on
epoch 21
# out of the other models.
# Validation error: 0.300
# Validation loss: 0.579
```

Part (c) - 2pt

Using the code in Part 0, any code from lecture notes, or any code that you write, compute and report the **test classification error** for your chosen model.

```
# If you use the `evaluate` function provided in part 0, you will need
to
# set batch_size > 1
train_loader, val_loader, test_loader, classes =
get_data_loader(target_classes=["cat", "dog"], batch_size=128)

criterion = nn.BCEWithLogitsLoss()
test_err, test_loss = evaluate(net, test_loader, criterion)
print("Test classification error & loss: ", test_err, test_loss)
```

```
Files already downloaded and verified
Files already downloaded and verified
Test classification error & loss:  0.291 0.5570533685386181
```

Part (d) - 3pt

How does the test classification error compare with the **validation error**? Explain why you would expect the test error to be *higher* than the validation error.

```
# The test classification error is somehow smaller than the validation
error.
```

```
# This is unexpected since with each iterarion, we are training our
model with the
# training set, and validated with the validation set, the model is
gradually
# becoming a good fit for the training dataset and we stop training
when we get
# a good validation error. This specified training of the model with
the training
# and validation set should result in a higher test classification
error. As it
# was never seen or used by the model.
```

Part (e) - 2pt

Why did we only use the test data set at the very end? Why is it important that we use the test data as little as possible?

```
# We keep the test data aside and they are not used in any way,
# which means it is a good data set to evaluate how good our model
# actually is at classify cats and dogs.
# If we use the test data in any way, it will affect the model
# and it may 'memorize' a bit of the test data and thus
# it won't be able to truly measure the accuracy of the model.
```

Part (f) - 5pt

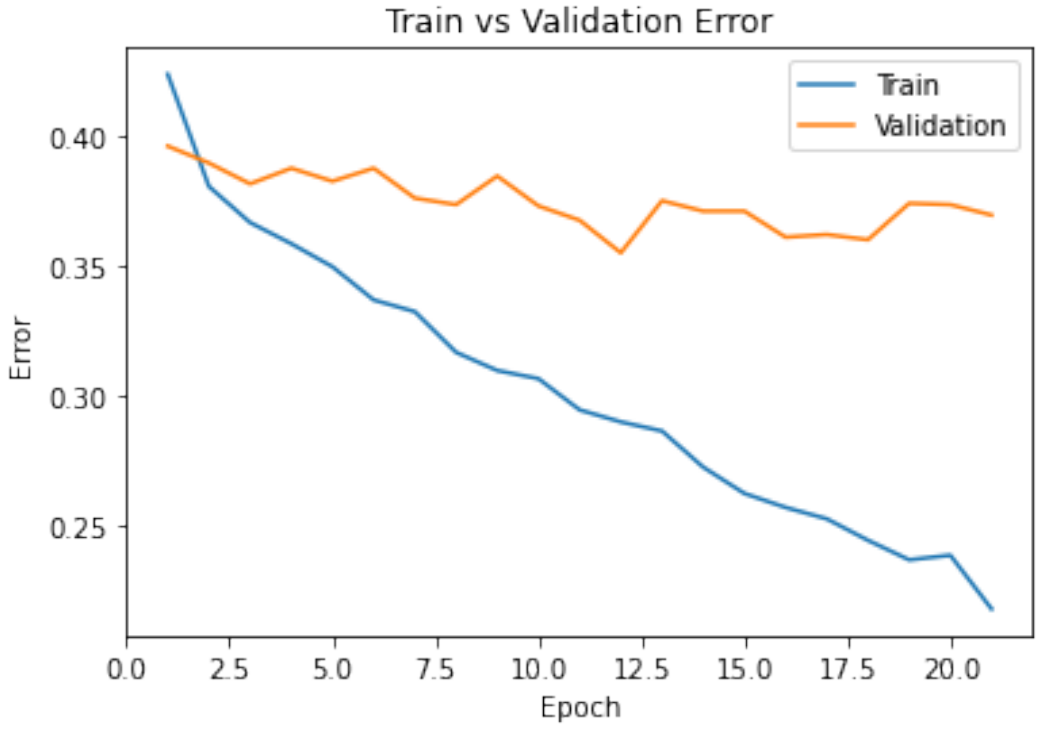
How does the your best CNN model compare with an 2-layer ANN model (no convolutional layers) on classifying cat and dog images. You can use a 2-layer ANN architecture similar to what you used in Lab 1. You should explore different hyperparameter settings to determine how well you can do on the validation dataset. Once satisified with the performance, you may test it out on the test data.

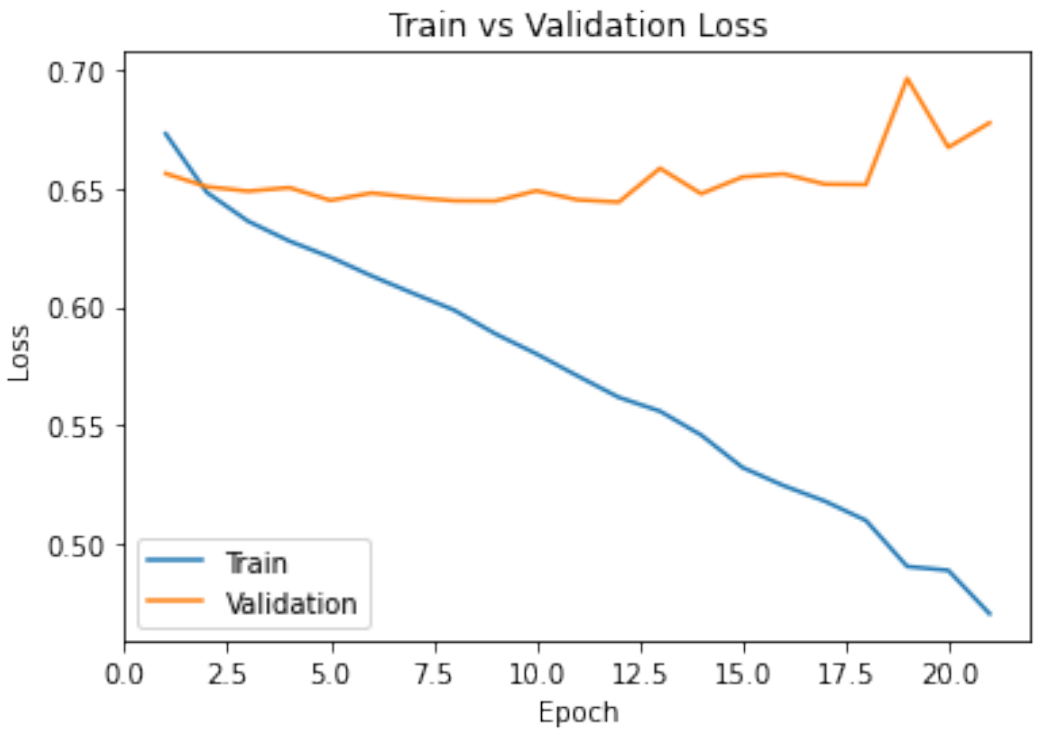
Hint: The ANN in lab 1 was applied on greyscale images. The cat and dog images are colour (RGB) and so you will need to flatted and concatinate all three colour layers before feeding them into an ANN.

```
import torch
import torch.nn as nn
import torch.nn.functional as F
from torchvision import datasets, transforms
import matplotlib.pyplot as plt # for plotting
import torch.optim as optim

torch.manual_seed(1) # set the random seed

# define a 2-layer artificial neural network
class Pigeon(nn.Module):
    def __init__(self):
        super(Pigeon, self).__init__()
        self.layer1 = nn.Linear(3*32*32, 30)
        self.layer2 = nn.Linear(30, 1)
```

```
        self.name = "ANN"
    def forward(self, img):
        flattened = img.view(-1, 3*32*32)
        activation1 = self.layer1(flattened)
        activation1 = F.relu(activation1)
        activation2 = self.layer2(activation1)
        return activation2.squeeze()

ANN_model = Pigeon()
train_net(ANN_model, 256, 0.012, 21)
model_path_ANN = get_model_name("ANN", batch_size=256,
learning_rate=0.012, epoch=20)
plot_training_curve(model_path_ANN)

train_loader, val_loader, test_loader, classes =
get_data_loader(target_classes=["cat", "dog"], batch_size = 256)

criterion = nn.BCEWithLogitsLoss()
test_err, test_loss = evaluate(ANN_model, test_loader, criterion)
print("Test classification error & loss:", test_err, test_loss)

# We see a larger and fairly stagnent validation error and loss with
the ANN model.
# And as expected, we see a larger test classification error and loss.
# Therefore CNN is better for this problem especially since here, we
aren't just
# looking for 'patterns', but 'features' that can be found anythere
and in any
# orientation (etc.) within each image.

Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.423625, Train loss: 0.6732779536396265 |
Validation err: 0.396, Validation loss: 0.6564163491129875
Epoch 2: Train err: 0.3805, Train loss: 0.6484026610851288 |Validation
err: 0.3895, Validation loss: 0.6508420407772064
Epoch 3: Train err: 0.366625, Train loss: 0.6362212598323822 |
Validation err: 0.3815, Validation loss: 0.6489596590399742
Epoch 4: Train err: 0.3585, Train loss: 0.6278365030884743 |Validation
err: 0.3875, Validation loss: 0.6504446268081665
Epoch 5: Train err: 0.349625, Train loss: 0.6210001818835735 |
Validation err: 0.3825, Validation loss: 0.6450524106621742
Epoch 6: Train err: 0.336875, Train loss: 0.6130423974245787 |
Validation err: 0.3875, Validation loss: 0.6481978595256805
Epoch 7: Train err: 0.332375, Train loss: 0.6056976336985826 |
Validation err: 0.376, Validation loss: 0.6462835296988487
Epoch 8: Train err: 0.31675, Train loss: 0.598584245890379 |Validation
err: 0.3735, Validation loss: 0.644893579185009
Epoch 9: Train err: 0.30975, Train loss: 0.5885412935167551 |
Validation err: 0.3845, Validation loss: 0.6447923332452774
```

```
Epoch 10: Train err: 0.306625, Train loss: 0.5800145827233791 |
Validation err: 0.373, Validation loss: 0.6491403430700302
Epoch 11: Train err: 0.294625, Train loss: 0.5706486105918884 |
Validation err: 0.3675, Validation loss: 0.6452895253896713
Epoch 12: Train err: 0.29, Train loss: 0.5615821648389101 |Validation
err: 0.355, Validation loss: 0.6443315967917442
Epoch 13: Train err: 0.2865, Train loss: 0.5557048842310905 |
Validation err: 0.375, Validation loss: 0.6586569845676422
Epoch 14: Train err: 0.272625, Train loss: 0.5456112921237946 |
Validation err: 0.371, Validation loss: 0.6478543877601624
Epoch 15: Train err: 0.2625, Train loss: 0.5318823605775833 |
Validation err: 0.371, Validation loss: 0.6549616381525993
Epoch 16: Train err: 0.257125, Train loss: 0.5241568675264716 |
Validation err: 0.361, Validation loss: 0.6562584936618805
Epoch 17: Train err: 0.25275, Train loss: 0.5176172284409404 |
Validation err: 0.362, Validation loss: 0.6519330441951752
Epoch 18: Train err: 0.2445, Train loss: 0.5094163371250033 |
Validation err: 0.36, Validation loss: 0.651750348508358
Epoch 19: Train err: 0.237, Train loss: 0.48990136105567217 |
Validation err: 0.374, Validation loss: 0.6967718228697777
Epoch 20: Train err: 0.23875, Train loss: 0.48828090261667967 |
Validation err: 0.3735, Validation loss: 0.6675217151641846
Epoch 21: Train err: 0.218125, Train loss: 0.46987520810216665 |
Validation err: 0.3695, Validation loss: 0.6778889298439026
Finished Training
Total time elapsed: 65.23 seconds
```






```
Files already downloaded and verified
Files already downloaded and verified
Test classification error & loss: 0.3545 0.665071614086628
```