Lab 2: Cats vs Dogs

Deadline: Feb 01, 5:00pm

Late Penalty: There is a penalty-free grace period of one hour past the deadline. Any work that is submitted between 1 hour and 24 hours past the deadline will receive a 20% grade deduction. No other late work is accepted. Quercus submission time will be used, not your local computer time. You can submit your labs as many times as you want before the deadline, so please submit often and early.

Marking TA: Tinglin (Francis) Duan

This lab is partially based on an assignment developed by Prof. Jonathan Rose and Harris Chan.

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/1Tttzv5OCC9Ry-XkoKD8Dly1k_clCrN0-/view?usp=sharing (https://drive.google.com/file/d/1Tttzv5OCC9Ry-XkoKD8Dly1k_clCrN0-/view?usp=sharing)

In [88]:
```
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.

In [89]:
```
################################################################################
#
# 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, validati
on
    and testing datasets. Returns data loaders for the three preprocessed data
sets.

    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 s
ize
        val_loader: iterable validation dataset organized according to batch s
ize
        test_loader: iterable testing dataset organized according to batch siz
e
        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], r
elevant_indices[split:]
    train_sampler = SubsetRandomSampler(relevant_train_indices)
    train_loader = torch.utils.data.DataLoader(trainset, batch_size=batch_size
,
                                               num_workers=1, sampler=train_sa
mpler)
    val_sampler = SubsetRandomSampler(relevant_val_indices)
    val_loader = torch.utils.data.DataLoader(trainset, batch_size=batch_size,
                                             num_workers=1, sampler=val_sampl
er)
    # 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_clas
ses)
    test_sampler = SubsetRandomSampler(relevant_test_indices)
    test_loader = torch.utils.data.DataLoader(testset, batch_size=batch_size,
                                              num_workers=1, sampler=test_sampl
er)
    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 val
ues

    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 se
t
         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 (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.

In [90]:
```
# 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
```

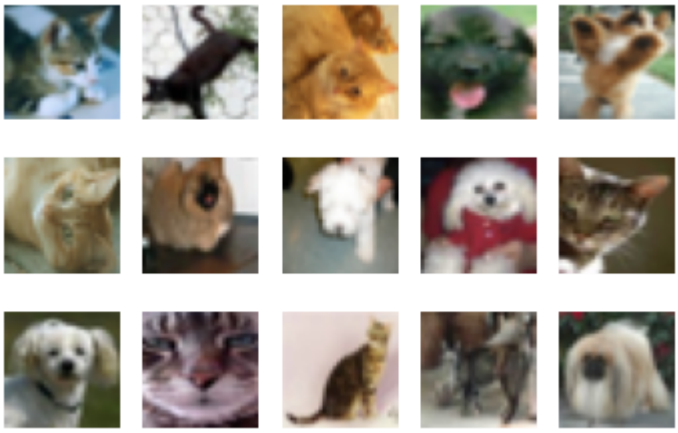
```
Files already downloaded and verified
Files already downloaded and verified
```

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.)

In [91]:
```
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?

In [ ]:
```
#The number of training examples that we have
print(len(train_loader))
#The number of validation examples that we have
print(len(val_loader))
#The number of test examples that we have
print(len(test_loader))
```

```
8000
2000
2000
```

In [ ]:
```
#As we see above, we have 8000 training examples, 2000 validation examples, an
d 2000 test examples.
```

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?

In [ ]:
```
'''
We need a validation set when training our model because we want to test how w
ell our model performs on
data on new data that it hasn't yet seen. Using the validation set, we are to
 test our model more, and tune
the hyperparameters to get more optimal hyperparameters. We are also able to j
udge the performance of model to make
sure that model isn't overfitted or underfitted to the training data.

By testing our model on this new set of data, we are able to understand whethe
r the model is actually able to
find patterns in new data, rather than memorizing the training data. If the tr
ainging set loss/error is very low, it
doesn't necessarily mean that our model is perfect. We must test the model on
 new data, otherwise overfitting will
occur and our model will perform poorly on new data. This is why we use the va
lidation set loss and error. Once
the performance of our model on the validation set loss/error is good, we can
 believe that our model is performing
well because it is able to take in new data and is able to find patterns.
'''
```

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.

In [92]:
```
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
```

In [93]:
```
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
```

In [94]:
```
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.

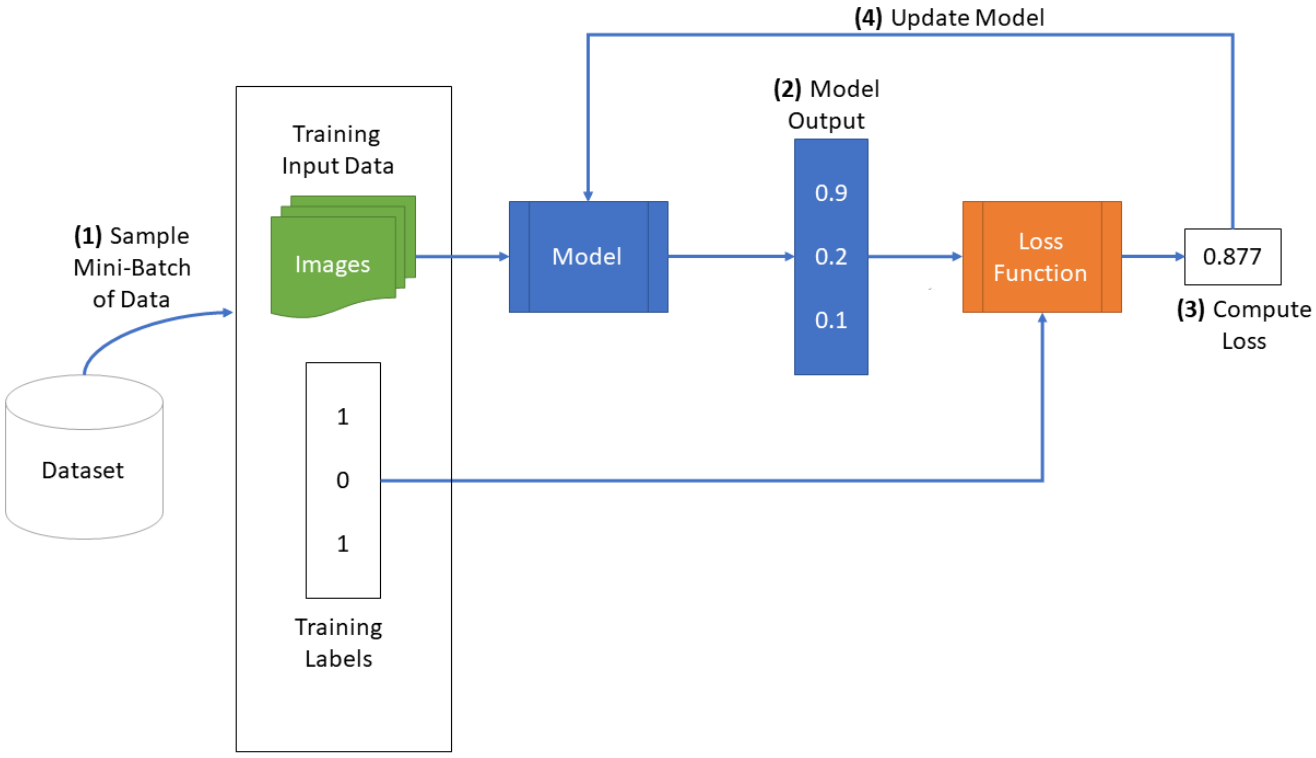
What is the total number of parameters in `small_net` and in `large_net` ? (Hint: how many numbers are in each tensor?)

In [ ]:
```
# Total number of parameters in small_net
numSmallNet = 0
for param in small_net.parameters():
  print(param.shape)
  temp = 1
  for num in param.shape:
    temp *= num
  numSmallNet += temp
print("The total number of parameters in small_net are:", numSmallNet, "\n")

# Total number of parameters in large_net
numLargeNet = 0
for param in large_net.parameters():
  print(param.shape)
  temp = 1
  for num in param.shape:
    temp *= num
  numLargeNet += temp
print("The total number of parameters in large_net are:", numLargeNet)
```

```
torch.Size([5, 3, 3, 3])
torch.Size([5])
torch.Size([1, 245])
torch.Size([1])
The total number of parameters in small_net are: 386

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 total number of parameters in large_net are: 9705
```

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:



In [95]:
```
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`?

In [ ]:
```
'''
The default value for batch_size is:64
The default value for learning_rate is: 0.01
The default value for num_epochs is: 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.

In [97]:
```
train_net(small_net, num_epochs = 5)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.413875, Train loss: 0.6668427748680115 |Validation err:
0.376, Validation loss: 0.6483204085379839
Epoch 2: Train err: 0.353625, Train loss: 0.6354253726005554 |Validation err:
0.3645, Validation loss: 0.6413258500397205
Epoch 3: Train err: 0.341, Train loss: 0.6217396693229675 |Validation err: 0.
3435, Validation loss: 0.6201915051788092
Epoch 4: Train err: 0.329875, Train loss: 0.60835813498497 |Validation err:
0.364, Validation loss: 0.6329441871494055
Epoch 5: Train err: 0.31775, Train loss: 0.5969561800956726 |Validation err:
0.328, Validation loss: 0.6153274364769459
Finished Training
Total time elapsed: 24.74 seconds
```

In [ ]:
```
'''
The following files are written to disk when we call train_net with small_net
 fpr 5 epochs:
- 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

- 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
'''
```

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?

In [ ]:
```
# 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
```

In [99]:
```
train_net(small_net)
train_net(large_net)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.309625, Train loss: 0.5879180536270142 |Validation err:
0.317, Validation loss: 0.6042345855385065
Epoch 2: Train err: 0.304, Train loss: 0.5802587697505951 |Validation err: 0.
317, Validation loss: 0.6066851885989308
Epoch 3: Train err: 0.302625, Train loss: 0.5755624876022339 |Validation err:
0.3095, Validation loss: 0.5905197225511074
Epoch 4: Train err: 0.292375, Train loss: 0.5687170197963715 |Validation err:
0.3275, Validation loss: 0.5995452860370278
Epoch 5: Train err: 0.291, Train loss: 0.5653229866027832 |Validation err: 0.
305, Validation loss: 0.5978708108887076
Epoch 6: Train err: 0.290125, Train loss: 0.5602969827651978 |Validation err:
0.299, Validation loss: 0.5889843497425318
Epoch 7: Train err: 0.290875, Train loss: 0.5589898490905761 |Validation err:
0.303, Validation loss: 0.5814108997583389
Epoch 8: Train err: 0.288625, Train loss: 0.5546396696567535 |Validation err:
0.3015, Validation loss: 0.5801545893773437
Epoch 9: Train err: 0.28825, Train loss: 0.5532357187271119 |Validation err:
0.303, Validation loss: 0.5883431695401669
Epoch 10: Train err: 0.281125, Train loss: 0.5478386735916138 |Validation er
r: 0.2955, Validation loss: 0.5735764410346746
Epoch 11: Train err: 0.28125, Train loss: 0.546239798784256 |Validation err:
0.294, Validation loss: 0.5808581309393048
Epoch 12: Train err: 0.28375, Train loss: 0.5446567549705505 |Validation err:
0.3125, Validation loss: 0.586477586068213
Epoch 13: Train err: 0.2815, Train loss: 0.5474525845050812 |Validation err:
0.297, Validation loss: 0.5817755600437522
Epoch 14: Train err: 0.28025, Train loss: 0.5417408337593078 |Validation err:
0.3065, Validation loss: 0.5958071257919073
Epoch 15: Train err: 0.273875, Train loss: 0.5407234272956848 |Validation er
r: 0.2945, Validation loss: 0.5806952025741339
Epoch 16: Train err: 0.280125, Train loss: 0.546589670419693 |Validation err:
0.2915, Validation loss: 0.5865953778848052
Epoch 17: Train err: 0.277875, Train loss: 0.5414178128242493 |Validation er
r: 0.299, Validation loss: 0.5809450093656778
Epoch 18: Train err: 0.274625, Train loss: 0.5387701487541199 |Validation er
r: 0.29, Validation loss: 0.5786385675892234
Epoch 19: Train err: 0.274125, Train loss: 0.536606663942337 |Validation err:
0.304, Validation loss: 0.5885510966181755
Epoch 20: Train err: 0.274625, Train loss: 0.5381642262935639 |Validation er
r: 0.295, Validation loss: 0.5866779051721096
Epoch 21: Train err: 0.28025, Train loss: 0.5387669327259064 |Validation err:
0.292, Validation loss: 0.5769840655848384
Epoch 22: Train err: 0.270125, Train loss: 0.5355625414848327 |Validation er
r: 0.3065, Validation loss: 0.5972487954422832
Epoch 23: Train err: 0.272625, Train loss: 0.5372099514007569 |Validation er
r: 0.305, Validation loss: 0.5880988147109747
Epoch 24: Train err: 0.271875, Train loss: 0.5358914661407471 |Validation er
r: 0.308, Validation loss: 0.5899906009435654
Epoch 25: Train err: 0.27475, Train loss: 0.5339503724575043 |Validation err:
0.3075, Validation loss: 0.5878982208669186
Epoch 26: Train err: 0.268, Train loss: 0.5331314561367035 |Validation err:
0.2935, Validation loss: 0.5783721078187227
Epoch 27: Train err: 0.272, Train loss: 0.533193708896637 |Validation err: 0.
3005, Validation loss: 0.6001411937177181
Epoch 28: Train err: 0.269125, Train loss: 0.5336558721065521 |Validation er
```

```
r: 0.299, Validation loss: 0.5867011845111847
Epoch 29: Train err: 0.272625, Train loss: 0.5354579954147339 |Validation er
r: 0.3025, Validation loss: 0.5912659149616957
Epoch 30: Train err: 0.272625, Train loss: 0.5360015184879303 |Validation er
r: 0.3075, Validation loss: 0.5939730918034911
Finished Training
Total time elapsed: 132.78 seconds
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.458625, Train loss: 0.6912559385299683 |Validation err:
0.4125, Validation loss: 0.6856102962046862
Epoch 2: Train err: 0.4335, Train loss: 0.6814668297767639 |Validation err:
0.41, Validation loss: 0.6759910173714161
Epoch 3: Train err: 0.398875, Train loss: 0.6667328643798828 |Validation err:
0.3815, Validation loss: 0.6537610292434692
Epoch 4: Train err: 0.370625, Train loss: 0.6467131075859069 |Validation err:
0.3695, Validation loss: 0.6425430569797754
Epoch 5: Train err: 0.3545, Train loss: 0.6303386397361755 |Validation err:
0.3495, Validation loss: 0.6228064205497503
Epoch 6: Train err: 0.330625, Train loss: 0.6097808685302735 |Validation err:
0.3485, Validation loss: 0.620961170643568
Epoch 7: Train err: 0.325, Train loss: 0.5954954442977906 |Validation err: 0.
328, Validation loss: 0.6003127247095108
Epoch 8: Train err: 0.306375, Train loss: 0.5788433806896209 |Validation err:
0.3135, Validation loss: 0.5895236246287823
Epoch 9: Train err: 0.296625, Train loss: 0.5700030171871185 |Validation err:
0.3305, Validation loss: 0.5882030054926872
Epoch 10: Train err: 0.2905, Train loss: 0.5563972742557526 |Validation err:
0.303, Validation loss: 0.5782106500118971
Epoch 11: Train err: 0.2765, Train loss: 0.5452412524223328 |Validation err:
0.307, Validation loss: 0.5930136684328318
Epoch 12: Train err: 0.271875, Train loss: 0.5333733639717102 |Validation er
r: 0.3065, Validation loss: 0.5812457604333758
Epoch 13: Train err: 0.266625, Train loss: 0.5203292398452759 |Validation er
r: 0.3015, Validation loss: 0.5749138984829187
Epoch 14: Train err: 0.259, Train loss: 0.5116986064910889 |Validation err:
0.295, Validation loss: 0.5819093231111765
Epoch 15: Train err: 0.253625, Train loss: 0.5032074828147888 |Validation er
r: 0.2995, Validation loss: 0.5973652713000774
Epoch 16: Train err: 0.24475, Train loss: 0.5002501213550568 |Validation err:
0.2995, Validation loss: 0.5842541502788663
Epoch 17: Train err: 0.237375, Train loss: 0.4867300102710724 |Validation er
r: 0.295, Validation loss: 0.5720811374485493
Epoch 18: Train err: 0.226875, Train loss: 0.4705198895931244 |Validation er
r: 0.302, Validation loss: 0.5883396854624152
Epoch 19: Train err: 0.222625, Train loss: 0.4637680356502533 |Validation er
r: 0.323, Validation loss: 0.6188610810786486
Epoch 20: Train err: 0.2175, Train loss: 0.45523372864723205 |Validation err:
0.3035, Validation loss: 0.6151538360863924
Epoch 21: Train err: 0.212625, Train loss: 0.44561918163299563 |Validation er
r: 0.3025, Validation loss: 0.6184177752584219
Epoch 22: Train err: 0.204375, Train loss: 0.43142758345603943 |Validation er
r: 0.315, Validation loss: 0.6207799045369029
Epoch 23: Train err: 0.198625, Train loss: 0.4196669731140137 |Validation er
r: 0.298, Validation loss: 0.6144433505833149
Epoch 24: Train err: 0.189125, Train loss: 0.4101913994550705 |Validation er
r: 0.3095, Validation loss: 0.6123676467686892
```

```
Epoch 25: Train err: 0.1825, Train loss: 0.3924684023857117 |Validation err:
0.3135, Validation loss: 0.6457232059910893
Epoch 26: Train err: 0.177875, Train loss: 0.3822400149106979 |Validation er
r: 0.319, Validation loss: 0.6687311697751284
Epoch 27: Train err: 0.16525, Train loss: 0.3678799693584442 |Validation err:
0.3255, Validation loss: 0.6671360805630684
Epoch 28: Train err: 0.169125, Train loss: 0.3670025726556778 |Validation er
r: 0.311, Validation loss: 0.6988273141905665
Epoch 29: Train err: 0.156625, Train loss: 0.34166969072818754 |Validation er
r: 0.319, Validation loss: 0.7637617951259017
Epoch 30: Train err: 0.13925, Train loss: 0.3183439316749573 |Validation err:
0.3185, Validation loss: 0.7235083160921931
Finished Training
Total time elapsed: 146.09 seconds
```

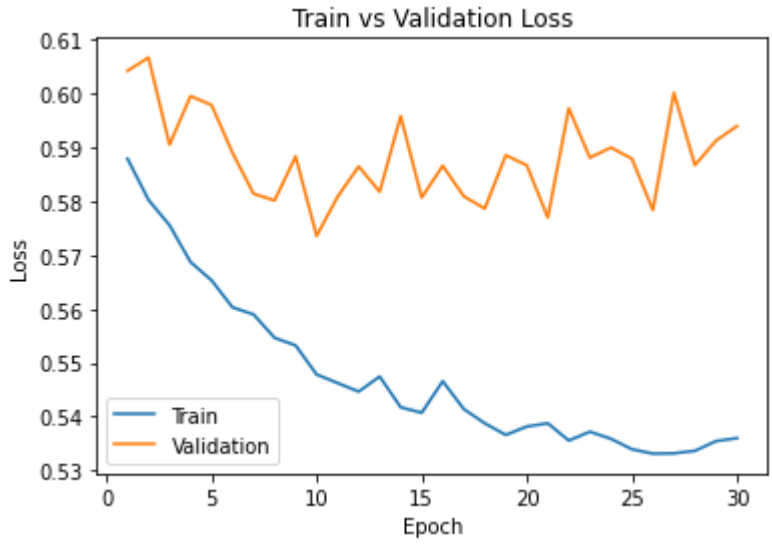
In [ ]:
```
'''
The small_net training took 132.78 seconds.
The large_net training took 146.09 seconds.

It took longer to do the large_net training because it has an additional layer
compared to small_net.
The large_net also has more parameters that need to be updated in the trainin
g. This is why it takes
longer to do the large_net training.
'''
```

Part (e) - 2pt

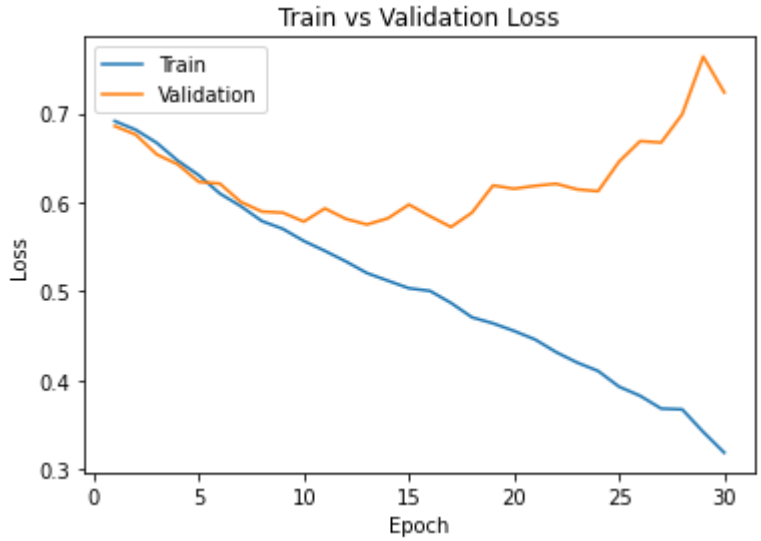
Use the function `plot_training_curve` to display the trajectory of the training/validation error and the training/validation loss. You will need to use the function `get_model_name` to generate the argument to the `plot_training_curve` function.

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

In [100]:
```
#model_path = get_model_name("small", batch_size=??, learning_rate=??, epoch=29)
small_model_path = get_model_name("small", batch_size=64, learning_rate=0.01, epoch=29)
plot_training_curve(small_model_path)
```



In [101]:
```
large_model_path = get_model_name("large", batch_size=64, learning_rate=0.01,
epoch=29)
plot_training_curve(large_model_path)
```



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.

In [ ]:
```
'''
Looking at the Training vs Validation curves for the small_net, I can easily t
ell that the model shows
underfitting at the beginning. However, once the epoch count increases, both c
urves go down for the
error and loss graphs. I can also note that for the small_net curves, that the
validation error and
loss are fluctuating a lot more than they are compared to the training error a
nd loss curves. Overall,
it seems as the number of epochs increases both training and validation error
 and loss decrease, which
is a good sign.

Looking at the Training vs Validation curves for the large_net, it is evident
 that the training error
and loss curve is steadily decreasing as the epoch count increases. However, t
he validation error curve
seems to plateau after the 15 epoch mark, while the validation loss curve seem
s to increase after 15 epochs.
This gives a sign of overfitting occurring in large_net, where after 15 epoch
s, the model is overfitted
to the traininfg data. Additionally, I noticed that compared to the small_net
 curves, there is much
less fluctuation that occurs in large_net curves.
'''
```

Part 3. Optimization Parameters [12 pt]

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

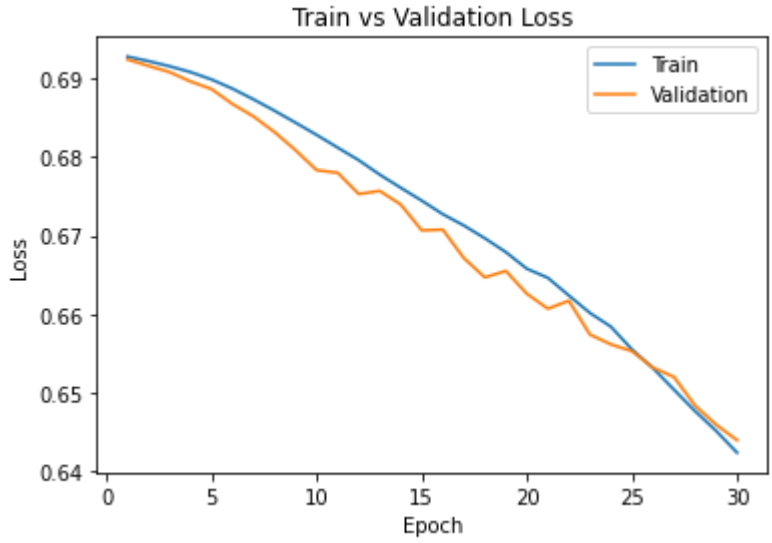
Part (a) - 3pt

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.

In [102]:
```
# 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, learning_rate=0.001)
plot_training_curve(get_model_name("large", batch_size=64, learning_rate=0.001
, epoch=29))
```

```
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: 0.42
4, 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 er
r: 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 er
r: 0.415, Validation loss: 0.6757059413939714
Epoch 14: Train err: 0.412375, Train loss: 0.6761112003326416 |Validation er
r: 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 er
r: 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 er
r: 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 er
r: 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 er
```

```
r: 0.3875, Validation loss: 0.6483502741903067
Epoch 29: Train err: 0.368375, Train loss: 0.6451257643699646 |Validation er
r: 0.3825, Validation loss: 0.6459067314863205
Epoch 30: Train err: 0.362625, Train loss: 0.6423329524993896 |Validation er
r: 0.3785, Validation loss: 0.6439237017184496
Finished Training
Total time elapsed: 147.26 seconds
```



In [ ]:
```
'''
The original large_net training with lr = 0.01 took 146.09 seconds.
This large_net training with lr = 0.001 took 147.26 seconds.
This means it takes roughly the same time to train the model with the learning
rate.

The effects of lowering the learning rate:
- produced the simmilar training and validation curves for both the error and
 loss graphs
- both training and validation error and losses decreased with the increase in
the number of epochs
- there is much less fluctuation in both graphs for both curves
- since lowering the learning rate, it requires more training because only sma
ll changes are made each time
- overall the value of loss and error is higher compared to that of a larger l
earning rate
- overfitting is avoided here

'''
```

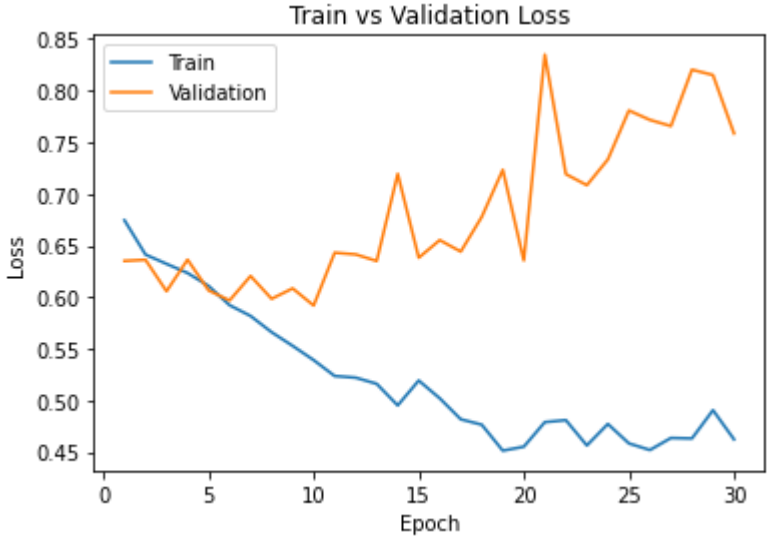
Part (b) - 3pt

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.

In [103]:
```
large_net = LargeNet()
train_net(large_net, learning_rate=0.1)
model_path_large = get_model_name("large", batch_size=64, learning_rate=0.1, e
poch=29)
plot_training_curve(model_path_large)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.4295, Train loss: 0.67437779712677 |Validation err: 0.3
595, 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 er
r: 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 er
r: 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 er
r: 0.3375, Validation loss: 0.6777342790737748
Epoch 19: Train err: 0.218125, Train loss: 0.45134368968009947 |Validation er
r: 0.3445, Validation loss: 0.7232250478118658
Epoch 20: Train err: 0.217875, Train loss: 0.45516350817680357 |Validation er
r: 0.3245, Validation loss: 0.6354950983077288
Epoch 21: Train err: 0.23275, Train loss: 0.47897080445289614 |Validation er
r: 0.3255, Validation loss: 0.8348110988736153
Epoch 22: Train err: 0.234875, Train loss: 0.4808810565471649 |Validation er
r: 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 er
r: 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 er
```

```
r: 0.348, Validation loss: 0.8202023077756166
Epoch 29: Train err: 0.235875, Train loss: 0.49053542733192446 |Validation er
r: 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: 148.05 seconds
```



In [ ]:
```
'''
The original large_net training with lr = 0.01 took 146.09 seconds.
This large_net training with lr = 0.001 took 147.26 seconds.
This large_net training with lr = 0.1 took 148.05 seconds.
This means it takes roughly the same time to train the model with the learning
rate.

The effects of increasing the learning rate:
- high fluctuations in both curves in error and loss graphs compared to a lowe
r learning rate
- leads to an exponential decrease in training error and loss curve
- no significant change in validation error as the number of epochs increase
- leads to an increase in validation loss as the number of epochs increase
- model seems to be overfitted as the number of epochs increase
'''
```

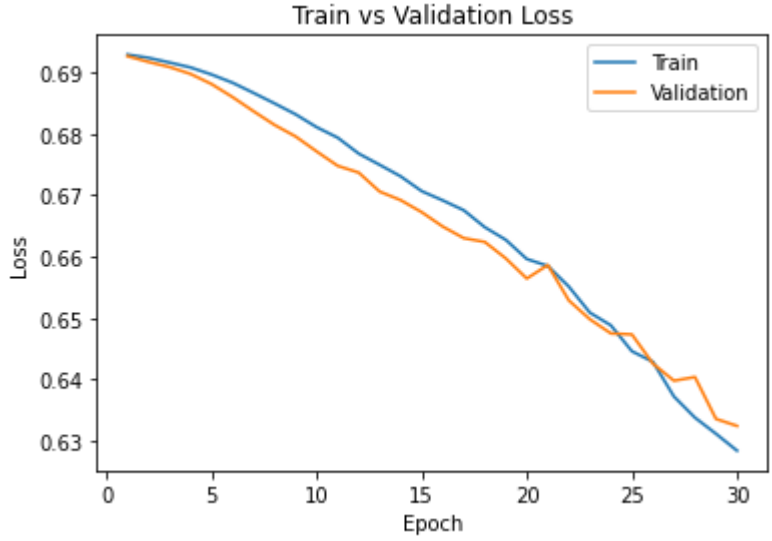
Part (c) - 3pt

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.

In [104]:
```
large_net = LargeNet()
train_net(large_net, batch_size=512)
model_path_large = get_model_name("large", batch_size=512, 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: 0.48175, 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: 0.4
285, 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 er
r: 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 er
r: 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 er
r: 0.405, Validation loss: 0.6649097055196762
Epoch 17: Train err: 0.393875, Train loss: 0.6675694733858109 |Validation er
r: 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 er
r: 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 er
r: 0.382, Validation loss: 0.6473268568515778
Epoch 26: Train err: 0.372625, Train loss: 0.6428566053509712 |Validation er
r: 0.3745, Validation loss: 0.6425703465938568
Epoch 27: Train err: 0.359375, Train loss: 0.6372117549180984 |Validation er
r: 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: 127.29 seconds
```



In [ ]:
```
'''
The original large_net training with bs = 64 took 146.09 seconds.
This large_net training with bs = 512 takes 127.29 seconds
This model with a larger batch size take less time to train. This is because w
ith this increase,
there is less iteration that takes place per epoch.

The effect of increasing the batch size:
- both training and validation error and loss decrease steadily with the incre
ase of number of epochs
- it takes longer to decrease loss curves for training and validation compared
to the original model
- less fluctuations in the curve, steady changes
- the model has access to more data, allowing it to learn ther pattern well ov
er more generalized data

'''
```

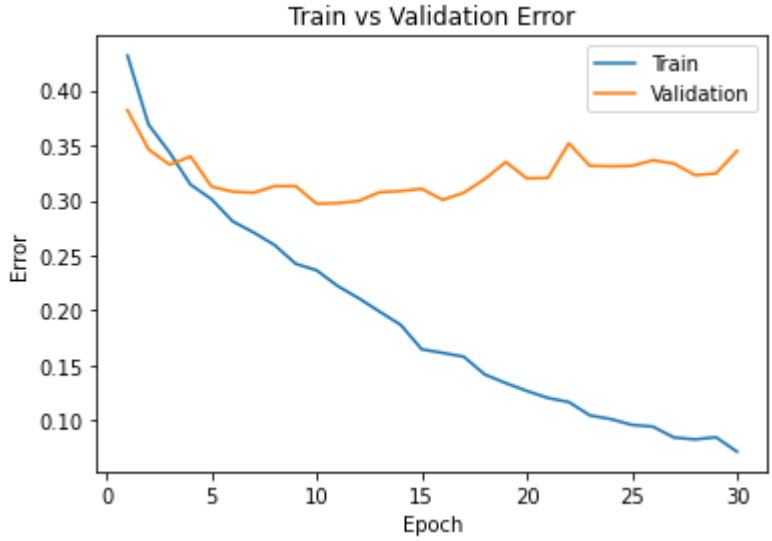
Part (d) - 3pt

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.

In [105]:
```
large_net = LargeNet()
train_net(large_net, batch_size=16)
model_path_large = get_model_name("large", batch_size=16, 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: 0.43175, Train loss: 0.6774994022846222 |Validation err:
0.382, Validation loss: 0.6513170118331909
Epoch 2: Train err: 0.369, Train loss: 0.639639899969101 |Validation err: 0.3
465, 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 er
r: 0.297, Validation loss: 0.5718690166473389
Epoch 11: Train err: 0.222125, Train loss: 0.4599769461452961 |Validation er
r: 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 er
r: 0.3195, Validation loss: 0.7913952842950821
Epoch 19: Train err: 0.13375, Train loss: 0.30618105667084455 |Validation er
r: 0.335, Validation loss: 0.8032052783966065
Epoch 20: Train err: 0.126625, Train loss: 0.3029071792438626 |Validation er
r: 0.32, Validation loss: 0.8106685240268707
Epoch 21: Train err: 0.12025, Train loss: 0.28682796490937473 |Validation er
r: 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 er
r: 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 er
r: 0.3315, Validation loss: 1.1338514368534087
Epoch 26: Train err: 0.094125, Train loss: 0.2325953512713313 |Validation er
r: 0.3365, Validation loss: 1.1414263204336166
Epoch 27: Train err: 0.08425, Train loss: 0.21040759468451142 |Validation er
r: 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 er
r: 0.345, Validation loss: 1.4871552000045776
Finished Training
Total time elapsed: 211.42 seconds
```



In [ ]:
```
'''
The original large_net training with bs = 64 took 146.09 seconds.
This large_net training with bs = 512 takes 127.29 seconds
This large_net training with bs = 16 takes 211.42 seconds
This model with a smaller batch size take a longer time to train. This is beca
use with this decrease
in batch size, there are more iterations that takes place per epoch.

The effect of decreasing the batch size:
- makes the model have very low training error and loss
- leads to overfitting as validation error stays stagnant as the number of epo
chs increase
- leads to overfittiing as validation loss increases as the number of eposchs
 increase
- overall, the model performs well on the training data, but poorly on new uns
een data
- doesn't produce a good model, because the model isn't generalized to correct
ly predict for new data

'''
```

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.

In [ ]:
```
'''
The following are the set of values for the hyperparameters, that I think woul
d help improve
validation accuracy:
- Network: large_net
The reason for this choice is that in part 2e, we saw that large_net provides
 better results
with less fluctuations and felt that it would be better at learning.

- batch_size: 256
The reasons for this choice were that it takes a less amount of time to train
 a larger batch size
and the validation and training error and loss curves decrease steadily as the
number of epochs
increase as we saw in part 3c.

- learning rate: 0.005
The reasons for this choice were that overfitting was avoided and we had stead
y decrease in error and
loss curves for both training an validation set as we saw in part 3c. Addition
ally, with an increasing
number of epochs, the results look promising.

- epoch: 40
The reason for an increase of the epoch count is that with more epochs, we gen
erally saw less error
and loss occuring. I am hoping that with the other chosen hyperparameter value
s, that this is a good
choice to make.
'''
```

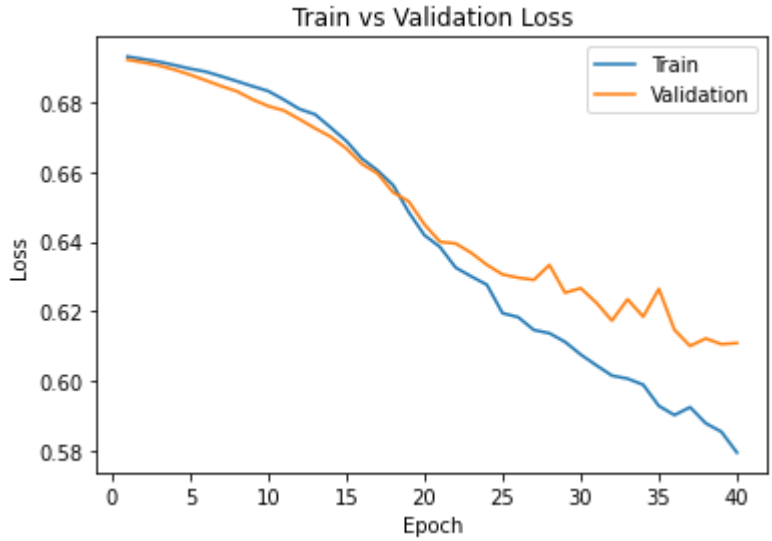
Part (b) - 1pt

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

In [ ]:
```
large_net = LargeNet()
train_net(large_net, batch_size= 256, learning_rate= 0.005, num_epochs=40)
model_path_large = get_model_name("large", batch_size=256, learning_rate=0.005
, epoch=39)
plot_training_curve(model_path_large)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.50225, Train loss: 0.6929918043315411 |Validation err:
0.4895, Validation loss: 0.6920944899320602
Epoch 2: Train err: 0.48725, Train loss: 0.6922785621136427 |Validation err:
0.4675, Validation loss: 0.6913378164172173
Epoch 3: Train err: 0.453125, Train loss: 0.6915246844291687 |Validation err:
0.434, Validation loss: 0.6904729753732681
Epoch 4: Train err: 0.45225, Train loss: 0.690559322014451 |Validation err:
0.432, Validation loss: 0.6892406046390533
Epoch 5: Train err: 0.448375, Train loss: 0.6895249728113413 |Validation err:
0.432, Validation loss: 0.6878118962049484
Epoch 6: Train err: 0.448125, Train loss: 0.6886843405663967 |Validation err:
0.428, Validation loss: 0.6861879155039787
Epoch 7: Train err: 0.44375, Train loss: 0.687343480065465 |Validation err:
0.431, Validation loss: 0.6845433935523033
Epoch 8: Train err: 0.443125, Train loss: 0.6859446428716183 |Validation err:
0.426, Validation loss: 0.6830432936549187
Epoch 9: Train err: 0.443625, Train loss: 0.6844949517399073 |Validation err:
0.4235, Validation loss: 0.6807261854410172
Epoch 10: Train err: 0.439375, Train loss: 0.6831344421952963 |Validation er
r: 0.4245, Validation loss: 0.6788140758872032
Epoch 11: Train err: 0.432125, Train loss: 0.6806647386401892 |Validation er
r: 0.4225, Validation loss: 0.6775638312101364
Epoch 12: Train err: 0.425875, Train loss: 0.6779768317937851 |Validation er
r: 0.417, Validation loss: 0.6750635728240013
Epoch 13: Train err: 0.42, Train loss: 0.6764416564255953 |Validation err: 0.
409, Validation loss: 0.6724516600370407
Epoch 14: Train err: 0.41125, Train loss: 0.6726823877543211 |Validation err:
0.408, Validation loss: 0.6700414717197418
Epoch 15: Train err: 0.40075, Train loss: 0.6688256449997425 |Validation err:
0.3985, Validation loss: 0.6666776314377785
Epoch 16: Train err: 0.38725, Train loss: 0.6637918706983328 |Validation err:
0.3955, Validation loss: 0.6622533053159714
Epoch 17: Train err: 0.3825, Train loss: 0.6603934373706579 |Validation err:
0.3945, Validation loss: 0.6595007106661797
Epoch 18: Train err: 0.37325, Train loss: 0.6561635211110115 |Validation err:
0.379, Validation loss: 0.6540893763303757
Epoch 19: Train err: 0.368875, Train loss: 0.648387698456645 |Validation err:
0.3865, Validation loss: 0.6515041664242744
Epoch 20: Train err: 0.357875, Train loss: 0.6418450940400362 |Validation er
r: 0.378, Validation loss: 0.6449322625994682
Epoch 21: Train err: 0.35525, Train loss: 0.6384237036108971 |Validation err:
0.369, Validation loss: 0.6399170830845833
Epoch 22: Train err: 0.35325, Train loss: 0.6325437482446432 |Validation err:
0.3685, Validation loss: 0.639463908970356
Epoch 23: Train err: 0.349375, Train loss: 0.6300209816545248 |Validation er
r: 0.358, Validation loss: 0.6367627829313278
Epoch 24: Train err: 0.35125, Train loss: 0.6276268921792507 |Validation err:
0.3565, Validation loss: 0.6333432570099831
Epoch 25: Train err: 0.34675, Train loss: 0.6194862555712461 |Validation err:
0.355, Validation loss: 0.6305832713842392
Epoch 26: Train err: 0.343875, Train loss: 0.6183481328189373 |Validation er
r: 0.3485, Validation loss: 0.6296534016728401
Epoch 27: Train err: 0.34025, Train loss: 0.6146794985979795 |Validation err:
0.353, Validation loss: 0.6290242373943329
Epoch 28: Train err: 0.33825, Train loss: 0.6137466821819544 |Validation err:
```

```
0.37, Validation loss: 0.6333177387714386
Epoch 29: Train err: 0.33925, Train loss: 0.611265741288662 |Validation err:
0.352, Validation loss: 0.6253420189023018
Epoch 30: Train err: 0.331625, Train loss: 0.6076459847390652 |Validation er
r: 0.351, Validation loss: 0.6266531646251678
Epoch 31: Train err: 0.328375, Train loss: 0.6045604012906551 |Validation er
r: 0.347, Validation loss: 0.6224947646260262
Epoch 32: Train err: 0.32875, Train loss: 0.6016061473637819 |Validation err:
0.339, Validation loss: 0.6173536255955696
Epoch 33: Train err: 0.32425, Train loss: 0.6007344089448452 |Validation err:
0.3395, Validation loss: 0.6234751045703888
Epoch 34: Train err: 0.320375, Train loss: 0.5989894699305296 |Validation er
r: 0.35, Validation loss: 0.6184636205434799
Epoch 35: Train err: 0.31575, Train loss: 0.5929280035197735 |Validation err:
0.3445, Validation loss: 0.62644212692976
Epoch 36: Train err: 0.31925, Train loss: 0.5902958251535892 |Validation err:
0.3365, Validation loss: 0.6147052049636841
Epoch 37: Train err: 0.319875, Train loss: 0.5925506986677647 |Validation er
r: 0.3345, Validation loss: 0.6100946292281151
Epoch 38: Train err: 0.316375, Train loss: 0.5879574175924063 |Validation er
r: 0.331, Validation loss: 0.6122488006949425
Epoch 39: Train err: 0.311, Train loss: 0.5855193845927715 |Validation err:
0.331, Validation loss: 0.6106091141700745
Epoch 40: Train err: 0.308375, Train loss: 0.5795379020273685 |Validation er
r: 0.3375, Validation loss: 0.6109202951192856
Finished Training
Total time elapsed: 166.54 seconds
```



Part (c) - 2pt

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

In [ ]:
```
'''
Overall, I am seeing good results in ny results from part a. However, I think
 I would be able to decrease
my validation error by reducing the batch size. This is why I will keep all th
e other values of my
hyperparameters the same and I will reduce my batch size to 128.
- Network: large_net
- batch_size: 128
- learning rate: 0.005
- epoch: 40
'''
```

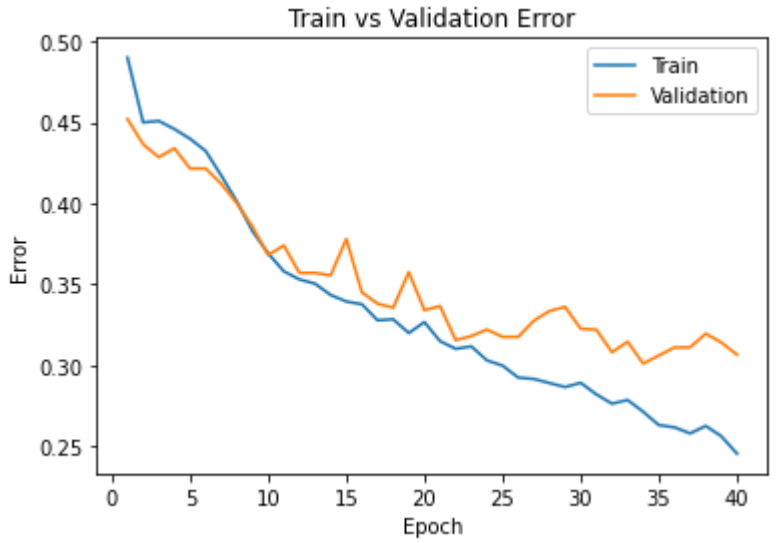
Part (d) - 1pt

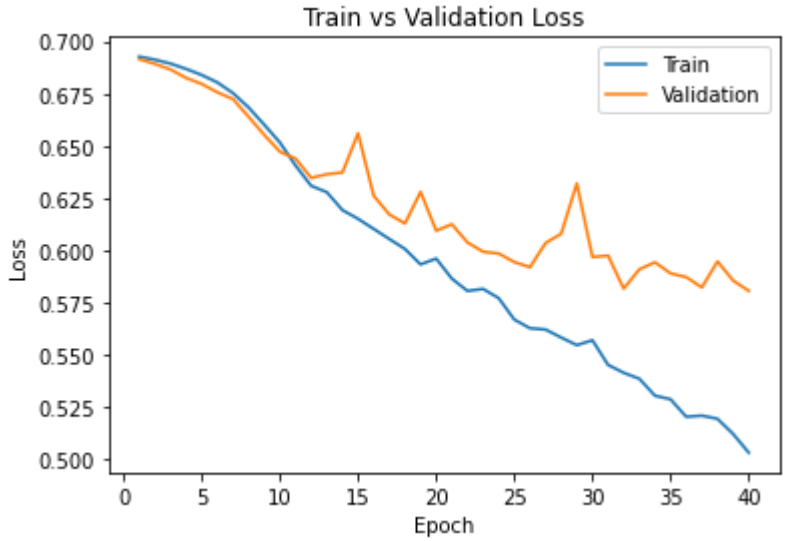
Train the model with the hyperparameters you chose in part(c), and include the training curve.

In [ ]:
```
large_net = LargeNet()
train_net(large_net, batch_size= 128, learning_rate= 0.005, num_epochs=40)
model_path_large = get_model_name("large", batch_size=128, learning_rate=0.005
, epoch=39)
plot_training_curve(model_path_large)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.49, Train loss: 0.6926485489285181 |Validation err: 0.4
52, Validation loss: 0.6914155781269073
Epoch 2: Train err: 0.45, Train loss: 0.6911715988128905 |Validation err: 0.4
365, Validation loss: 0.6891648061573505
Epoch 3: Train err: 0.450875, Train loss: 0.6893004746664138 |Validation err:
0.4285, Validation loss: 0.6864676289260387
Epoch 4: Train err: 0.445875, Train loss: 0.6867584406383453 |Validation err:
0.434, Validation loss: 0.6824363358318806
Epoch 5: Train err: 0.439875, Train loss: 0.6838485920240008 |Validation err:
0.4215, Validation loss: 0.6795621104538441
Epoch 6: Train err: 0.432375, Train loss: 0.6802851529348464 |Validation err:
0.4215, Validation loss: 0.6755634807050228
Epoch 7: Train err: 0.417125, Train loss: 0.6751870399429685 |Validation err:
0.412, Validation loss: 0.6723174154758453
Epoch 8: Train err: 0.401125, Train loss: 0.6683242150715419 |Validation err:
0.4, Validation loss: 0.6638390496373177
Epoch 9: Train err: 0.38275, Train loss: 0.660055140654246 |Validation err:
0.3855, Validation loss: 0.6551868543028831
Epoch 10: Train err: 0.369125, Train loss: 0.6515874749138242 |Validation er
r: 0.368, Validation loss: 0.6471745520830154
Epoch 11: Train err: 0.358, Train loss: 0.6405744770216564 |Validation err:
0.374, Validation loss: 0.6439055502414703
Epoch 12: Train err: 0.353, Train loss: 0.6308747283996098 |Validation err:
0.357, Validation loss: 0.6346632950007915
Epoch 13: Train err: 0.350375, Train loss: 0.6278500859699552 |Validation er
r: 0.357, Validation loss: 0.6364160887897015
Epoch 14: Train err: 0.343375, Train loss: 0.6192907549086071 |Validation er
r: 0.3555, Validation loss: 0.6372502446174622
Epoch 15: Train err: 0.339375, Train loss: 0.6150988293072533 |Validation er
r: 0.378, Validation loss: 0.655949492007494
Epoch 16: Train err: 0.337625, Train loss: 0.6102908953787789 |Validation er
r: 0.345, Validation loss: 0.6261377297341824
Epoch 17: Train err: 0.32775, Train loss: 0.6054614356585911 |Validation err:
0.338, Validation loss: 0.6171704046428204
Epoch 18: Train err: 0.328375, Train loss: 0.6007088082177299 |Validation er
r: 0.3355, Validation loss: 0.6128831617534161
Epoch 19: Train err: 0.319875, Train loss: 0.5933214246280609 |Validation er
r: 0.3575, Validation loss: 0.6279834806919098
Epoch 20: Train err: 0.326625, Train loss: 0.5960910348665147 |Validation er
r: 0.334, Validation loss: 0.6094172224402428
Epoch 21: Train err: 0.314875, Train loss: 0.5865307648976644 |Validation er
r: 0.3365, Validation loss: 0.6124989725649357
Epoch 22: Train err: 0.310125, Train loss: 0.580638745474437 |Validation err:
0.3155, Validation loss: 0.6038050763309002
Epoch 23: Train err: 0.311625, Train loss: 0.5815922154320611 |Validation er
r: 0.318, Validation loss: 0.5993748158216476
Epoch 24: Train err: 0.303, Train loss: 0.5771962853651198 |Validation err:
0.322, Validation loss: 0.5985310822725296
Epoch 25: Train err: 0.29975, Train loss: 0.5669393402243418 |Validation err:
0.3175, Validation loss: 0.5944199673831463
Epoch 26: Train err: 0.292375, Train loss: 0.5628059208393097 |Validation er
r: 0.3175, Validation loss: 0.5919636525213718
Epoch 27: Train err: 0.2915, Train loss: 0.5621800772727482 |Validation err:
0.3275, Validation loss: 0.603543933480978
Epoch 28: Train err: 0.289, Train loss: 0.5583502503614577 |Validation err:
```

```
0.3335, Validation loss: 0.6079720221459866
Epoch 29: Train err: 0.2865, Train loss: 0.5547232656251817 |Validation err:
0.336, Validation loss: 0.6320360228419304
Epoch 30: Train err: 0.289125, Train loss: 0.5570540130138397 |Validation er
r: 0.3225, Validation loss: 0.5968603566288948
Epoch 31: Train err: 0.282, Train loss: 0.545271236745138 |Validation err: 0.
322, Validation loss: 0.597486911341548
Epoch 32: Train err: 0.27625, Train loss: 0.5415014227231344 |Validation err:
0.308, Validation loss: 0.5817348062992096
Epoch 33: Train err: 0.2785, Train loss: 0.5386569651346358 |Validation err:
0.3145, Validation loss: 0.5909833051264286
Epoch 34: Train err: 0.27125, Train loss: 0.5305579020863488 |Validation err:
0.301, Validation loss: 0.5943150594830513
Epoch 35: Train err: 0.263, Train loss: 0.5288341882682982 |Validation err:
0.306, Validation loss: 0.5890081021934748
Epoch 36: Train err: 0.261625, Train loss: 0.5204828428843665 |Validation er
r: 0.311, Validation loss: 0.5872396472841501
Epoch 37: Train err: 0.257875, Train loss: 0.5210273062425946 |Validation er
r: 0.311, Validation loss: 0.5823167432099581
Epoch 38: Train err: 0.2625, Train loss: 0.5195190915985713 |Validation err:
0.3195, Validation loss: 0.5947305113077164
Epoch 39: Train err: 0.256125, Train loss: 0.5122779626694937 |Validation er
r: 0.314, Validation loss: 0.5856000520288944
Epoch 40: Train err: 0.245375, Train loss: 0.5033376368265303 |Validation er
r: 0.3065, Validation loss: 0.5806790571659803
Finished Training
Total time elapsed: 170.33 seconds
```





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`.

In [ ]:
```
net = LargeNet()
train_net(net, batch_size= 128, learning_rate= 0.005, num_epochs= 45)
model_path = get_model_name(net.name, batch_size=128, learning_rate=0.005, epo
ch=44)
state = torch.load(model_path)
net.load_state_dict(state)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.48, Train loss: 0.6922999232534378 |Validation err: 0.4
65, Validation loss: 0.6917495727539062
Epoch 2: Train err: 0.455125, Train loss: 0.6905293142984784 |Validation err:
0.4565, Validation loss: 0.689816553145647
Epoch 3: Train err: 0.454875, Train loss: 0.68835511283269 |Validation err:
0.441, Validation loss: 0.687373086810112
Epoch 4: Train err: 0.44225, Train loss: 0.6853962833919223 |Validation err:
0.426, Validation loss: 0.6834773197770119
Epoch 5: Train err: 0.43575, Train loss: 0.682204008102417 |Validation err:
0.4265, Validation loss: 0.6803669407963753
Epoch 6: Train err: 0.428, Train loss: 0.6790150167450072 |Validation err: 0.
418, Validation loss: 0.6764487251639366
Epoch 7: Train err: 0.422125, Train loss: 0.6749966239172315 |Validation err:
0.4135, Validation loss: 0.6716253831982613
Epoch 8: Train err: 0.420625, Train loss: 0.6710521020586528 |Validation err:
0.409, Validation loss: 0.6669573038816452
Epoch 9: Train err: 0.39875, Train loss: 0.665031007358006 |Validation err:
0.398, Validation loss: 0.6590025499463081
Epoch 10: Train err: 0.393, Train loss: 0.6608822317350478 |Validation err:
0.3775, Validation loss: 0.654971931129694
Epoch 11: Train err: 0.376625, Train loss: 0.6506085717488849 |Validation er
r: 0.363, Validation loss: 0.6464779451489449
Epoch 12: Train err: 0.368875, Train loss: 0.6413248039427257 |Validation er
r: 0.367, Validation loss: 0.6376986131072044
Epoch 13: Train err: 0.36125, Train loss: 0.6369090884450882 |Validation err:
0.3605, Validation loss: 0.6355678774416447
Epoch 14: Train err: 0.353, Train loss: 0.6270689500702752 |Validation err:
0.3635, Validation loss: 0.6377138905227184
Epoch 15: Train err: 0.344875, Train loss: 0.6231563535947648 |Validation er
r: 0.374, Validation loss: 0.6519850268959999
Epoch 16: Train err: 0.33825, Train loss: 0.6159142955901131 |Validation err:
0.354, Validation loss: 0.6305421628057957
Epoch 17: Train err: 0.329625, Train loss: 0.6092148981397114 |Validation er
r: 0.3385, Validation loss: 0.619023721665144
Epoch 18: Train err: 0.324375, Train loss: 0.6032980585855151 |Validation er
r: 0.3315, Validation loss: 0.6135424897074699
Epoch 19: Train err: 0.316625, Train loss: 0.5951021616421048 |Validation er
r: 0.345, Validation loss: 0.6303162686526775
Epoch 20: Train err: 0.318, Train loss: 0.5922294353681897 |Validation err:
0.3305, Validation loss: 0.6113960221409798
Epoch 21: Train err: 0.316125, Train loss: 0.5935579519423227 |Validation er
r: 0.331, Validation loss: 0.6117657199501991
Epoch 22: Train err: 0.30875, Train loss: 0.5842402227341182 |Validation err:
0.331, Validation loss: 0.61233726516366
Epoch 23: Train err: 0.30275, Train loss: 0.5809815904450795 |Validation err:
0.3205, Validation loss: 0.6025172397494316
Epoch 24: Train err: 0.302125, Train loss: 0.5763545055238027 |Validation er
r: 0.315, Validation loss: 0.6008283160626888
Epoch 25: Train err: 0.293125, Train loss: 0.5700276054087139 |Validation er
r: 0.3245, Validation loss: 0.5998523645102978
Epoch 26: Train err: 0.289875, Train loss: 0.5633928331117781 |Validation er
r: 0.315, Validation loss: 0.5947279576212168
Epoch 27: Train err: 0.2875, Train loss: 0.5634510999634152 |Validation err:
0.324, Validation loss: 0.6118839010596275
Epoch 28: Train err: 0.287, Train loss: 0.5572559630113935 |Validation err:
```

```
0.315, Validation loss: 0.5972356349229813
Epoch 29: Train err: 0.286125, Train loss: 0.5518338122065105 |Validation er
r: 0.3085, Validation loss: 0.5980875566601753
Epoch 30: Train err: 0.279125, Train loss: 0.5510327376070476 |Validation er
r: 0.331, Validation loss: 0.6089434176683426
Epoch 31: Train err: 0.2825, Train loss: 0.5503940667424884 |Validation err:
0.3335, Validation loss: 0.6223097108304501
Epoch 32: Train err: 0.2735, Train loss: 0.5410180579102228 |Validation err:
0.2945, Validation loss: 0.5824554450809956
Epoch 33: Train err: 0.2715, Train loss: 0.5404279042804052 |Validation err:
0.3125, Validation loss: 0.5944762900471687
Epoch 34: Train err: 0.265, Train loss: 0.528831517412549 |Validation err: 0.
3, Validation loss: 0.5896271504461765
Epoch 35: Train err: 0.266125, Train loss: 0.530660351117452 |Validation err:
0.305, Validation loss: 0.5880730636417866
Epoch 36: Train err: 0.262125, Train loss: 0.5242588945797512 |Validation er
r: 0.2965, Validation loss: 0.5854867901653051
Epoch 37: Train err: 0.254625, Train loss: 0.5181891057226393 |Validation er
r: 0.293, Validation loss: 0.5821769461035728
Epoch 38: Train err: 0.2595, Train loss: 0.5170552948164562 |Validation err:
0.2925, Validation loss: 0.5815979689359665
Epoch 39: Train err: 0.25475, Train loss: 0.5122198050930387 |Validation err:
0.307, Validation loss: 0.58584413677454
Epoch 40: Train err: 0.249, Train loss: 0.5066938655717033 |Validation err:
0.29, Validation loss: 0.5815385859459639
Epoch 41: Train err: 0.246875, Train loss: 0.5075293911827935 |Validation er
r: 0.299, Validation loss: 0.5906733199954033
Epoch 42: Train err: 0.24225, Train loss: 0.4997344362357306 |Validation err:
0.294, Validation loss: 0.5822618789970875
Epoch 43: Train err: 0.246375, Train loss: 0.5021141918878707 |Validation er
r: 0.295, Validation loss: 0.5836627073585987
Epoch 44: Train err: 0.245875, Train loss: 0.5015004035972414 |Validation er
r: 0.313, Validation loss: 0.6029088981449604
Epoch 45: Train err: 0.24175, Train loss: 0.49523567964160253 |Validation er
r: 0.286, Validation loss: 0.5757768154144287
Finished Training
Total time elapsed: 192.74 seconds
```

Out[ ]: <All keys matched successfully>

Part (b) - 2pt

Justify your choice of model from part (a).

In [ ]:
```
'''
Overall, this is the best combination of values for the hyperparameters that p
roduced the best results
for me. In epoch 45 in the results above, I achived the lowest validation erro
r rate of 0.286, meaning
a validation accuracy of 71.4%.

- Network: large_net
The reason for this choice is that in part 2e, we saw that large_net provides
 better results
with less fluctuations and felt that it would be better at learning.

- batch_size: 128
Here, I chose a large enough batch_size that would help generalize the algorit
hm, but not too large
that it would lower test accuracy.Also, the validation and training error and
 loss curves decrease
steadily as the number of epochs increase as we saw in part 3c.

- learning rate: 0.005
The reasons for this choice were that overfitting was avoided and we had stead
y decrease in error and
loss curves for both training an validation set as we saw in part 3c. Addition
ally, with an increasing
number of epochs, the results were great.

- epoch: 45
The reason for an increase of the epoch count is that with more epochs, we gen
erally saw less error
and loss occuring. I noticed that this was the perfect number to use, and if w
e were to use any more,
overfitting would began to occur.
'''
```

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.

In [ ]:
```
# 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)
test_err, test_loss = evaluate(net, test_loader, nn.BCEWithLogitsLoss())
print("Test classification error and loss:", test_err, test_loss)
```

```
Files already downloaded and verified
Files already downloaded and verified
Test classification error and loss: 0.299 0.5667464099824429
```

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.

In [ ]:
```
'''
As seen above, the test classification error is 29.9% which is slighly higher
 than the 28.6% validation
error. This suggests that the model performed slightly better on the validatio
n data set, then it did
on the test classification dataset. We would expect this because the model is
 not being updated
after running on the testing data. After getting a validation error, I tried t
o minimize it as much
as possible by adjusting thr hyperparameters that would lead me to the highest
validation accuracy.
However, we are not doing this to optimize testing accuracy. Therefore, valida
tion accuracy will
usually be higher than testing accuracy.
'''
```

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?

In [ ]:
```
'''
We saved the test data for the very end because we must only use it once to te
st the accuracy of our model
at the very end. Only after tuning my hyperparameters to my best abilities, sh
ould I use the test data.
This data must remain unbiased, otherwise we cannot truly evaluate the accurac
y of our model. If the
testing data has been used in the training and validation stage, we may never
 know if the model has
memorized the answers for the testing data, meaning that the testing accuracy
 is unreliable.
'''
```

Part (f) - 5pt

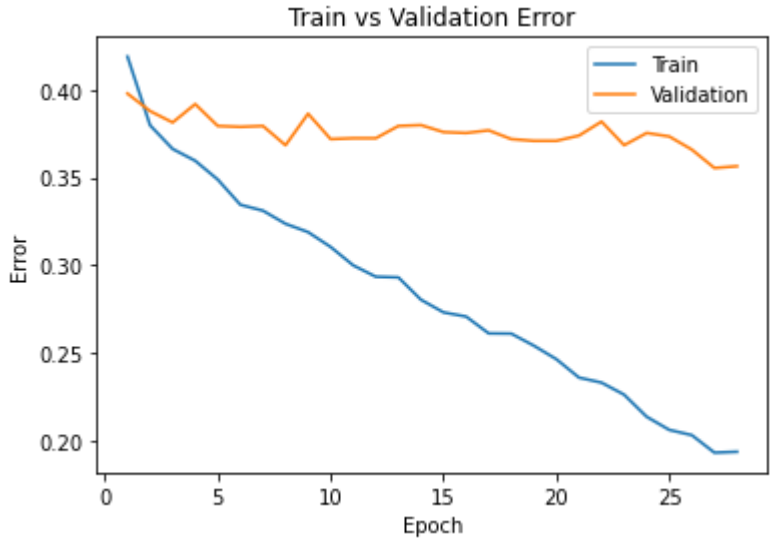
How does the your best CNN model compare with an 2-layer ANN model (no convolutional layers) on classifying cat and dog images. You can use a 2-layer ANN architecture similar to what you used in Lab 1. You should explore different hyperparameter settings to determine how well you can do on the validation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

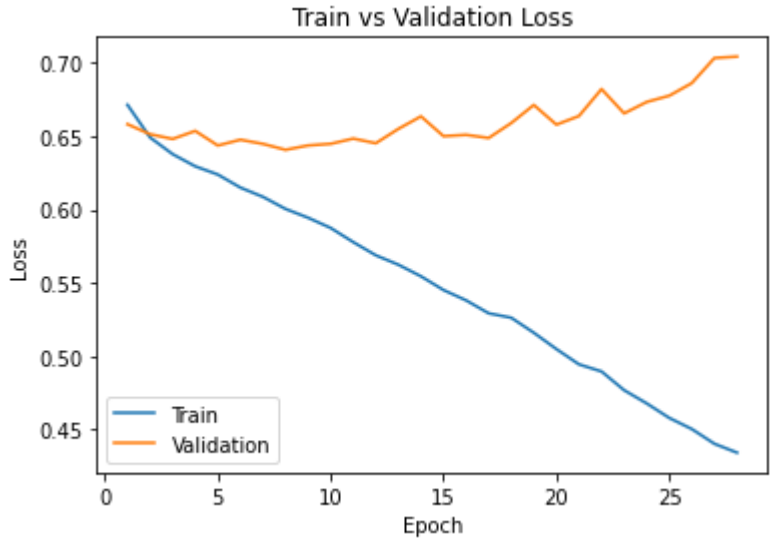
Hint: The ANN in lab 1 was applied on greyscale images. The cat and dog images are colour (RGB) and so you will need to flatted and concatinate all three colour layers before feeding them into an ANN.

In [ ]:
```
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 ANN(nn.Module):
    def __init__(self):
        super(ANN, self).__init__()
        self.layer1 = nn.Linear(32*32*3, 30)
        self.layer2 = nn.Linear(30, 1)
        self.name = "ANN"
    def forward(self, img):
        flattened = img.view(-1, 32*32*3)
        activation1 = self.layer1(flattened)
        activation1 = F.relu(activation1)
        activation2 = self.layer2(activation1)
        return activation2.squeeze()

ann = ANN()
train_net(ann, 128, 0.005, 28)
model_path_ANN = get_model_name("ANN", batch_size=128, learning_rate=0.005, ep
och=27)
plot_training_curve(model_path_ANN)

train_loader, val_loader, test_loader, classes = get_data_loader(
    target_classes=["cat", "dog"],
    batch_size = 128)

test_err, test_loss = evaluate(ann, test_loader, nn.BCEWithLogitsLoss())
print("Test classification error and loss:", test_err, test_loss)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.419125, Train loss: 0.6714658339818319 |Validation err:
0.398, Validation loss: 0.6582800447940826
Epoch 2: Train err: 0.379875, Train loss: 0.6492340470117236 |Validation err:
0.388, Validation loss: 0.6514426842331886
Epoch 3: Train err: 0.366375, Train loss: 0.6378535997299921 |Validation err:
0.3815, Validation loss: 0.6482101641595364
Epoch 4: Train err: 0.359625, Train loss: 0.6295622643970308 |Validation err:
0.392, Validation loss: 0.6536959111690521
Epoch 5: Train err: 0.348875, Train loss: 0.6239601411516704 |Validation err:
0.3795, Validation loss: 0.6437500938773155
Epoch 6: Train err: 0.3345, Train loss: 0.6150485759689694 |Validation err:
0.379, Validation loss: 0.6476834565401077
Epoch 7: Train err: 0.331125, Train loss: 0.6087291429913233 |Validation err:
0.3795, Validation loss: 0.6448529027402401
Epoch 8: Train err: 0.323625, Train loss: 0.6004973082315355 |Validation err:
0.3685, Validation loss: 0.6408060304820538
Epoch 9: Train err: 0.318875, Train loss: 0.5944759902499971 |Validation err:
0.3865, Validation loss: 0.6437809616327286
Epoch 10: Train err: 0.310375, Train loss: 0.5875210572802831 |Validation er
r: 0.372, Validation loss: 0.6448932103812695
Epoch 11: Train err: 0.299875, Train loss: 0.5778527893717327 |Validation er
r: 0.3725, Validation loss: 0.6484746783971786
Epoch 12: Train err: 0.293375, Train loss: 0.5688554091112954 |Validation er
r: 0.3725, Validation loss: 0.6453080251812935
Epoch 13: Train err: 0.293, Train loss: 0.5624305709959969 |Validation err:
0.3795, Validation loss: 0.6550955586135387
Epoch 14: Train err: 0.28025, Train loss: 0.5545295070088099 |Validation err:
0.38, Validation loss: 0.6636285483837128
Epoch 15: Train err: 0.273, Train loss: 0.5451534887154897 |Validation err:
0.376, Validation loss: 0.6501074358820915
Epoch 16: Train err: 0.270625, Train loss: 0.5381186136177608 |Validation er
r: 0.3755, Validation loss: 0.6510150246322155
Epoch 17: Train err: 0.261, Train loss: 0.529287137209423 |Validation err: 0.
377, Validation loss: 0.6488753072917461
Epoch 18: Train err: 0.260875, Train loss: 0.5262640738298022 |Validation er
r: 0.372, Validation loss: 0.6590788662433624
Epoch 19: Train err: 0.254, Train loss: 0.5160183679489863 |Validation err:
0.371, Validation loss: 0.6713938675820827
Epoch 20: Train err: 0.24625, Train loss: 0.5048912418267083 |Validation err:
0.371, Validation loss: 0.6579735763370991
Epoch 21: Train err: 0.23575, Train loss: 0.4944593972629971 |Validation err:
0.374, Validation loss: 0.6638547144830227
Epoch 22: Train err: 0.232875, Train loss: 0.4896033813082983 |Validation er
r: 0.382, Validation loss: 0.6822500266134739
Epoch 23: Train err: 0.226, Train loss: 0.4768485294448005 |Validation err:
0.3685, Validation loss: 0.6656559742987156
Epoch 24: Train err: 0.213375, Train loss: 0.4678313826757764 |Validation er
r: 0.3755, Validation loss: 0.6734554544091225
Epoch 25: Train err: 0.205875, Train loss: 0.4579206312459613 |Validation er
r: 0.3735, Validation loss: 0.6777112111449242
Epoch 26: Train err: 0.202875, Train loss: 0.45032972002786303 |Validation er
r: 0.366, Validation loss: 0.6863138005137444
Epoch 27: Train err: 0.192875, Train loss: 0.4402873170754266 |Validation er
r: 0.3555, Validation loss: 0.7034294679760933
Epoch 28: Train err: 0.193375, Train loss: 0.4342384962808518 |Validation er
```

```
r: 0.3565, Validation loss: 0.7043852843344212
Finished Training
Total time elapsed: 90.22 seconds
```






```
Files already downloaded and verified
Files already downloaded and verified
Test classification error and loss: 0.362 0.7014601714909077
```

In [ ]:
```
# The ANN model has an error of 36.2% and loss of 70.14%
# The CNN model had an error of 29.9% and a loss of 56.67%.
# Therefore, the best CNN model is better than the 2-layered ANN model.
# The CNN model fits better with our data.
```

In [ ]:
```
%%shell
jupyter nbconvert --to html /content/Lab_2_Cats_vs_Dogs.ipynb
```

```
[NbConvertApp] Converting notebook /content/Lab_2_Cats_vs_Dogs.ipynb to html
[NbConvertApp] Writing 992445 bytes to /content/Lab_2_Cats_vs_Dogs.html
```

Out[ ]: