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/1CkSngXc78HwSeuaFpuGcCwKXcYAJW1Ax/view?usp=sharing

In [1]:
```
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 [2]:

```
################################################################################
# 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[sp
    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.

In [3]:
```
# 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
```

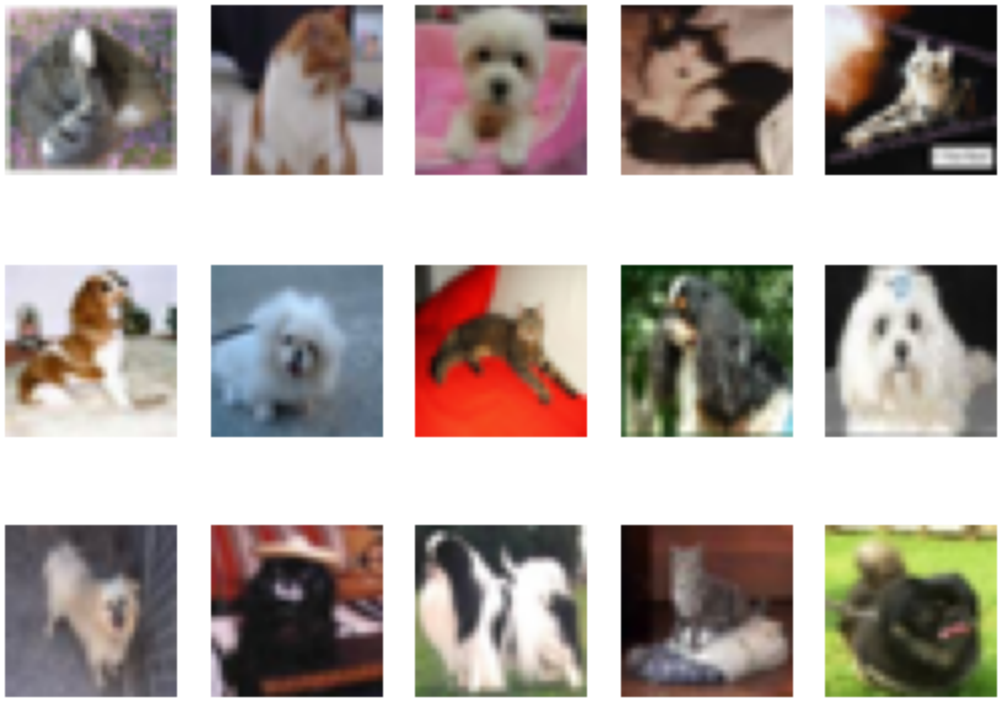
```
Files already downloaded and verified
Files already downloaded and verified
```

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 [4]:
```
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 [5]:
```
# General formula (in this example we know batch_size = 1)
print("Training data size: ", len(train_loader) * train_loader.batch_size)
print("Training data size: ", len(val_loader) * val_loader.batch_size)
print("Training data size: ", len(test_loader) * test_loader.batch_size)
```

```
Training data size:  8000
Training data size:  2000
Training data size:  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 use Training + Test data, the model will optimize towards (overfit to) test data.

Therefore, if we use the validation set with training set for training, we are guaranteed that our results from test data can be trusted

(results that resembles the user using our model for prediction)

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 [6]:
```
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 [7]:
```
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 [8]:
```
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.

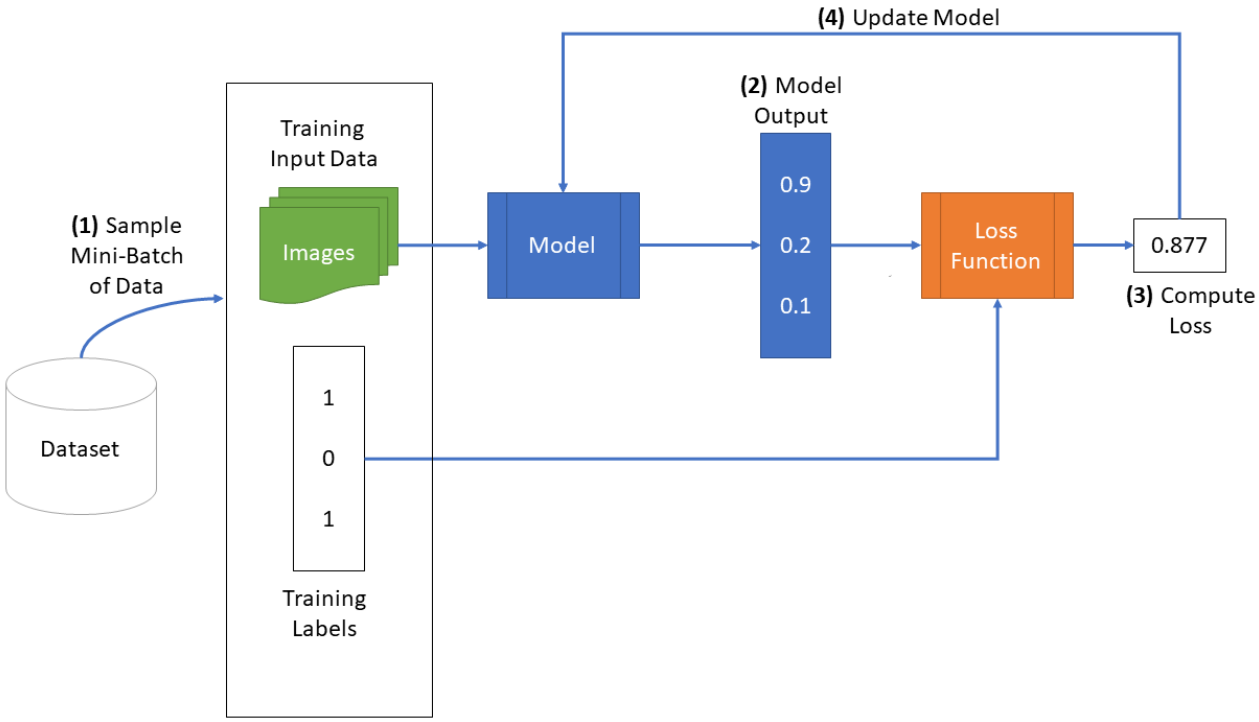
What is the total number of parameters in `small_net` and in `large_net`? (Hint: how many numbers are in each tensor?)

In [9]:
```
small_count = 0
for param in small_net.parameters():
    small_count += param.numel()

large_count = 0
for param in large_net.parameters():
    large_count += param.numel()

print(small_count)
print(large_count)
```

```
386
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 [10]:
```
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`?

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.

model_small_bs64_lr0.01_epoch0
model_small_bs64_lr0.01_epoch1
model_small_bs64_lr0.01_epoch2
model_small_bs64_lr0.01_epoch3
model_small_bs64_lr0.01_epoch4

Each file contains current model's state: net.state_dict() after each epoch. The state_dict() is a Python dictionary that maps each layer to its parameter tensor, containing all the modified weights and biases

model_small_bs64_lr0.01_train_err.csv
model_small_bs64_lr0.01_train_loss.csv

CSV files containing Training error and loss for all 5 epochs during training

model_small_bs64_lr0.01_val_err.csv
model_small_bs64_lr0.01_val_loss.csv

CSV files containing Validation error and loss for all 5 epochs during training

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 [11]:
```
# 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')
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/con
tent/gdrive", force_remount=True).
```

In [12]:

```
train_net(small_net)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.43175, Train loss: 0.676514790058136 |Validation err: 0.37, Validation los
s: 0.654131855815649
Epoch 2: Train err: 0.36325, Train loss: 0.6481061215400696 |Validation err: 0.3895, Validation
loss: 0.6607548110187054
Epoch 3: Train err: 0.3525, Train loss: 0.6373614816665649 |Validation err: 0.3465, Validation l
oss: 0.6261459644883871
Epoch 4: Train err: 0.34125, Train loss: 0.6226562075614929 |Validation err: 0.3335, Validation
loss: 0.6181808523833752
Epoch 5: Train err: 0.3335, Train loss: 0.6127196784019471 |Validation err: 0.327, Validation lo
ss: 0.6125406548380852
Epoch 6: Train err: 0.3245, Train loss: 0.6012994024753571 |Validation err: 0.323, Validation lo
ss: 0.605414055287838
Epoch 7: Train err: 0.32425, Train loss: 0.5946023759841919 |Validation err: 0.32, Validation lo
ss: 0.5971760079264641
Epoch 8: Train err: 0.31625, Train loss: 0.5870555379390716 |Validation err: 0.316, Validation l
oss: 0.5971901528537273
Epoch 9: Train err: 0.3095, Train loss: 0.5833774945735931 |Validation err: 0.3155, Validation l
oss: 0.5935652973130345
Epoch 10: Train err: 0.303375, Train loss: 0.5774561524391174 |Validation err: 0.307, Validation
loss: 0.5825354047119617
Epoch 11: Train err: 0.296875, Train loss: 0.5730876512527466 |Validation err: 0.305, Validation
loss: 0.5892521496862173
Epoch 12: Train err: 0.297, Train loss: 0.5672329540252685 |Validation err: 0.3115, Validation l
oss: 0.5872620427981019
Epoch 13: Train err: 0.29475, Train loss: 0.5679818923473359 |Validation err: 0.3035, Validation
loss: 0.5832488145679235
Epoch 14: Train err: 0.291875, Train loss: 0.5627726349830627 |Validation err: 0.3015, Validatio
n loss: 0.5868992395699024
Epoch 15: Train err: 0.289125, Train loss: 0.5585299370288849 |Validation err: 0.2995, Validatio
n loss: 0.579656564630568
Epoch 16: Train err: 0.293125, Train loss: 0.5641128072738647 |Validation err: 0.31, Validation
loss: 0.5966846281662583
Epoch 17: Train err: 0.28625, Train loss: 0.5574767847061157 |Validation err: 0.297, Validation
loss: 0.5749756535515189
Epoch 18: Train err: 0.283125, Train loss: 0.5535205278396607 |Validation err: 0.3015, Validatio
n loss: 0.5726198675110936
Epoch 19: Train err: 0.276875, Train loss: 0.5487292559146881 |Validation err: 0.3065, Validatio
n loss: 0.584893542341888
Epoch 20: Train err: 0.283125, Train loss: 0.5494786038398742 |Validation err: 0.297, Validation
loss: 0.5866814386099577
Epoch 21: Train err: 0.280875, Train loss: 0.5523348410129547 |Validation err: 0.2895, Validatio
n loss: 0.5699738468974829
Epoch 22: Train err: 0.281, Train loss: 0.5492669916152955 |Validation err: 0.2935, Validation l
oss: 0.5811173748224974
Epoch 23: Train err: 0.28275, Train loss: 0.5486447350978851 |Validation err: 0.2925, Validation
loss: 0.5700455829501152
Epoch 24: Train err: 0.27775, Train loss: 0.5454678859710693 |Validation err: 0.3035, Validation
loss: 0.5843176441267133
Epoch 25: Train err: 0.27425, Train loss: 0.5451472511291504 |Validation err: 0.298, Validation
loss: 0.5796531271189451
Epoch 26: Train err: 0.272375, Train loss: 0.5428876993656159 |Validation err: 0.284, Validation
loss: 0.5688709989190102
Epoch 27: Train err: 0.270875, Train loss: 0.5407608270645141 |Validation err: 0.29, Validation
loss: 0.578884712420404
Epoch 28: Train err: 0.274625, Train loss: 0.5416526658535004 |Validation err: 0.287, Validation
loss: 0.5680998340249062
Epoch 29: Train err: 0.27325, Train loss: 0.542454687833786 |Validation err: 0.2915, Validation
loss: 0.575818395242095
Epoch 30: Train err: 0.27375, Train loss: 0.5400322697162628 |Validation err: 0.292, Validation
loss: 0.5764603950083256
Finished Training
Total time elapsed: 92.32 seconds
```

In [13]:
```
train_net(large_net)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.4785, Train loss: 0.6912403373718262 |Validation err: 0.4475, Validation l
oss: 0.6879571303725243
Epoch 2: Train err: 0.442625, Train loss: 0.684390170097351 |Validation err: 0.426, Validation l
oss: 0.6833558473736048
Epoch 3: Train err: 0.409625, Train loss: 0.6717938461303711 |Validation err: 0.3885, Validation
loss: 0.659625057131052
Epoch 4: Train err: 0.371375, Train loss: 0.6478279495239258 |Validation err: 0.3695, Validation
loss: 0.6479862276464701
Epoch 5: Train err: 0.35125, Train loss: 0.6294472141265869 |Validation err: 0.347, Validation l
oss: 0.6339765731245279
Epoch 6: Train err: 0.33425, Train loss: 0.6129775280952454 |Validation err: 0.3395, Validation
loss: 0.6241807490587234
Epoch 7: Train err: 0.32925, Train loss: 0.6046936700344085 |Validation err: 0.3225, Validation
loss: 0.6052001565694809
Epoch 8: Train err: 0.313125, Train loss: 0.5851069984436035 |Validation err: 0.323, Validation
loss: 0.597500205039978
Epoch 9: Train err: 0.302375, Train loss: 0.5757834827899933 |Validation err: 0.296, Validation
loss: 0.578717322088778
Epoch 10: Train err: 0.28775, Train loss: 0.5558849699497223 |Validation err: 0.2885, Validation
loss: 0.5699162539094687
Epoch 11: Train err: 0.27825, Train loss: 0.5464601397514344 |Validation err: 0.2935, Validation
loss: 0.576210755854845
Epoch 12: Train err: 0.267125, Train loss: 0.5324718661308289 |Validation err: 0.294, Validation
loss: 0.5670441398397088
Epoch 13: Train err: 0.263, Train loss: 0.526974997997284 |Validation err: 0.297, Validation los
s: 0.5675178710371256
Epoch 14: Train err: 0.254, Train loss: 0.5113218626976013 |Validation err: 0.277, Validation lo
ss: 0.56549240835011
Epoch 15: Train err: 0.252, Train loss: 0.5041717562675476 |Validation err: 0.2755, Validation l
oss: 0.5540695739910007
Epoch 16: Train err: 0.24825, Train loss: 0.5047879762649536 |Validation err: 0.287, Validation
loss: 0.5623108297586441
Epoch 17: Train err: 0.239375, Train loss: 0.4891612873077393 |Validation err: 0.282, Validation
loss: 0.5545862056314945
Epoch 18: Train err: 0.23425, Train loss: 0.47584422659873965 |Validation err: 0.2775, Validatio
n loss: 0.5508657256141305
Epoch 19: Train err: 0.225375, Train loss: 0.4692206320762634 |Validation err: 0.281, Validation
loss: 0.5595511207357049
Epoch 20: Train err: 0.215625, Train loss: 0.4539679839611053 |Validation err: 0.296, Validation
loss: 0.6011877907440066
Epoch 21: Train err: 0.22325, Train loss: 0.4543112599849701 |Validation err: 0.2705, Validation
loss: 0.5769186718389392
Epoch 22: Train err: 0.207625, Train loss: 0.4395176603794098 |Validation err: 0.272, Validation
loss: 0.5685857869684696
Epoch 23: Train err: 0.204375, Train loss: 0.4307711968421936 |Validation err: 0.286, Validation
loss: 0.5900907889008522
Epoch 24: Train err: 0.191, Train loss: 0.4169521610736847 |Validation err: 0.292, Validation lo
ss: 0.5874557020142674
Epoch 25: Train err: 0.191625, Train loss: 0.4088097139596939 |Validation err: 0.275, Validation
loss: 0.5835014199838042
Epoch 26: Train err: 0.188, Train loss: 0.3975340234041214 |Validation err: 0.285, Validation lo
ss: 0.6008838033303618
Epoch 27: Train err: 0.17925, Train loss: 0.3873469865322113 |Validation err: 0.281, Validation
loss: 0.5893719987943769
Epoch 28: Train err: 0.1785, Train loss: 0.3896495385169983 |Validation err: 0.281, Validation l
oss: 0.611377757973969
Epoch 29: Train err: 0.166125, Train loss: 0.36964286589622497 |Validation err: 0.299, Validatio
n loss: 0.6644558114930987
Epoch 30: Train err: 0.162625, Train loss: 0.35503826177120207 |Validation err: 0.283, Validatio
n loss: 0.6316245757043362
Finished Training
Total time elapsed: 97.39 seconds
```

Obviously large_net longer to train, since there are more layers to calculate, more parameters to be updated

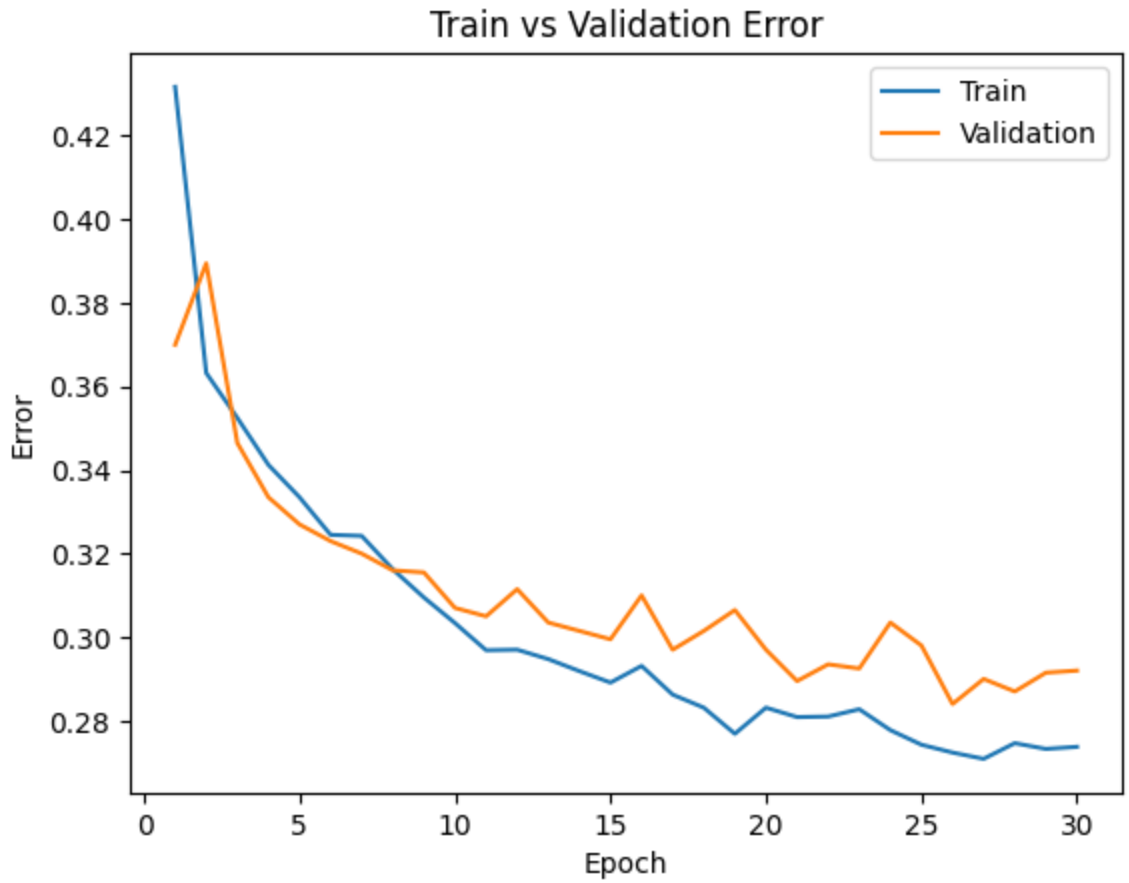
- Small: Total time elapsed: 92.32 seconds
- Large: Total time elapsed: 97.39 seconds

Part (e) - 2pt

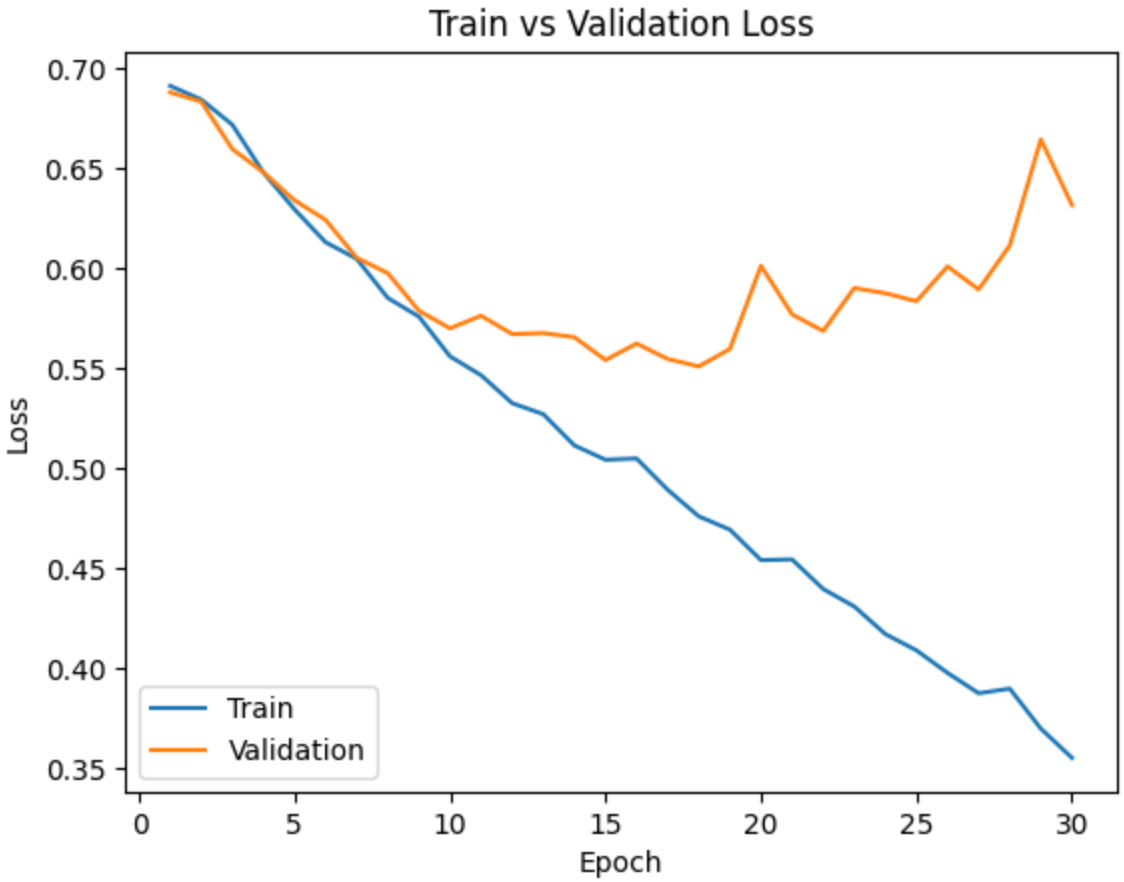
Use the function `plot_training_curve` to display the trajectory of the training/validation error and the training/validation loss. You will need to use the function `get_model_name` to generate the argument to the `plot_training_curve` function.

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

In [14]:
```
model_path = get_model_name("small", batch_size=64, learning_rate=0.01, epoch=29)
plot_training_curve(model_path)
```



In [15]:
```
model_path = get_model_name("large", batch_size=64, learning_rate=0.01, epoch=29)
plot_training_curve(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.

Similar: At the first epochs, Training error and Training loss is high. These decrease during the training process (which shows that the model is indeed learning)

Both small_net and large_net have relatively same fluctuations in error and loss (since they both have the same ammount of batch size = 64)

Validation error and loss decreases until epoch 5-10, then it slows down, starts diverging from Training error/loss (for small_net) or even increase (for large_net Validation loss). This is the sign of **overfitting**

Differences: The Training error & loss for small_net decreases almost exponentially, while that for large_net decreases almost linearly. The large_net achieves much lower Training error & Training loss in the end.

The large_net overfits more (have higher final validation error/loss) compared to small_net. Both models have ~equal lowest Validation error (around 0.28) and Validataion loss (around 0.55)

Part 3. Optimization Parameters [12 pt]

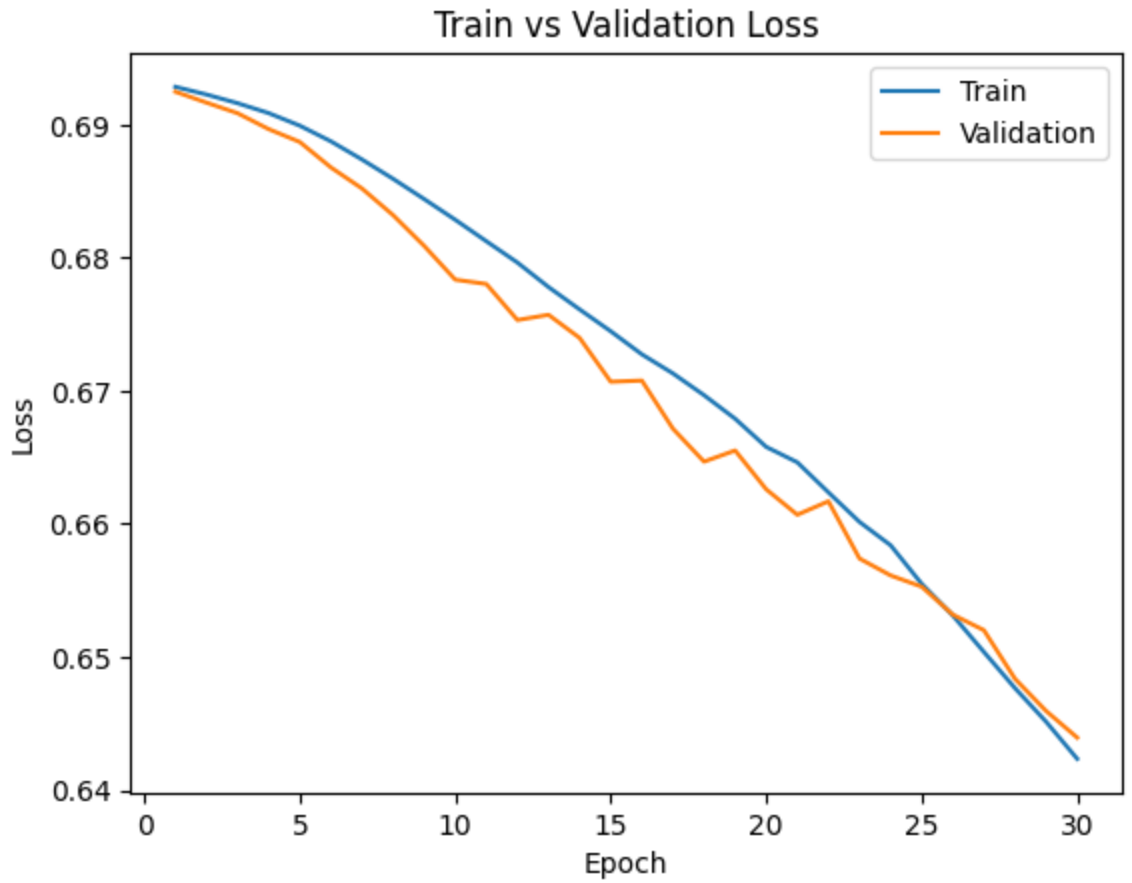
For this section, we will work with `large_net` only.

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 [16]:
```
# 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)
model_path = get_model_name("large", batch_size=64, learning_rate=0.001, epoch=29)
plot_training_curve(model_path)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.47625, Train loss: 0.6928360028266907 |Validation err: 0.467, Validation l
oss: 0.6924686599522829
Epoch 2: Train err: 0.448625, Train loss: 0.6922589740753173 |Validation err: 0.4305, Validation
loss: 0.6916493307799101
Epoch 3: Train err: 0.43575, Train loss: 0.6916067419052124 |Validation err: 0.4285, Validation
loss: 0.6908544395118952
Epoch 4: Train err: 0.430125, Train loss: 0.6908613877296448 |Validation err: 0.424, Validation
loss: 0.6896596923470497
Epoch 5: Train err: 0.434125, Train loss: 0.6899198365211486 |Validation err: 0.4195, Validation
loss: 0.6886942777782679
Epoch 6: Train err: 0.435875, Train loss: 0.6887419753074646 |Validation err: 0.4195, Validation
loss: 0.6867837514728308
Epoch 7: Train err: 0.436625, Train loss: 0.6873781814575195 |Validation err: 0.4185, Validation
loss: 0.6851996649056673
Epoch 8: Train err: 0.43725, Train loss: 0.6859267921447754 |Validation err: 0.4115, Validation
loss: 0.6831991747021675
Epoch 9: Train err: 0.424375, Train loss: 0.6844045443534851 |Validation err: 0.411, Validation
loss: 0.6808868050575256
Epoch 10: Train err: 0.424375, Train loss: 0.682848952293396 |Validation err: 0.408, Validation
loss: 0.6783499345183372
Epoch 11: Train err: 0.425375, Train loss: 0.6812354407310486 |Validation err: 0.4125, Validatio
n loss: 0.6780234910547733
Epoch 12: Train err: 0.419875, Train loss: 0.6796327586174011 |Validation err: 0.4125, Validatio
n loss: 0.6753160189837217
Epoch 13: Train err: 0.414625, Train loss: 0.6777928824424744 |Validation err: 0.415, Validation
loss: 0.6757084671407938
Epoch 14: Train err: 0.412, Train loss: 0.6761129403114319 |Validation err: 0.412, Validation lo
ss: 0.6739710867404938
Epoch 15: Train err: 0.409, Train loss: 0.6744727687835693 |Validation err: 0.415, Validation lo
ss: 0.6706812102347612
Epoch 16: Train err: 0.4065, Train loss: 0.67274143409729 |Validation err: 0.4105, Validation lo
ss: 0.670768965035677
Epoch 17: Train err: 0.40125, Train loss: 0.6713059720993042 |Validation err: 0.404, Validation
loss: 0.66715326346457
Epoch 18: Train err: 0.399375, Train loss: 0.6696724286079406 |Validation err: 0.4055, Validatio
n loss: 0.6646785754710436
Epoch 19: Train err: 0.40075, Train loss: 0.6679023985862732 |Validation err: 0.3955, Validation
loss: 0.6655161324888468
Epoch 20: Train err: 0.3925, Train loss: 0.6657856888771058 |Validation err: 0.405, Validation l
oss: 0.6625996753573418
Epoch 21: Train err: 0.38975, Train loss: 0.6646266269683838 |Validation err: 0.395, Validation
loss: 0.6606872715055943
Epoch 22: Train err: 0.38875, Train loss: 0.6623701963424683 |Validation err: 0.3935, Validation
loss: 0.6617010589689016
Epoch 23: Train err: 0.384125, Train loss: 0.6601490645408631 |Validation err: 0.3975, Validatio
n loss: 0.6573988571763039
Epoch 24: Train err: 0.3825, Train loss: 0.6583953781127929 |Validation err: 0.386, Validation l
oss: 0.6561295725405216
Epoch 25: Train err: 0.3785, Train loss: 0.6554959454536438 |Validation err: 0.3875, Validation
loss: 0.6552845854312181
Epoch 26: Train err: 0.376625, Train loss: 0.6531237239837646 |Validation err: 0.387, Validation
loss: 0.6531836222857237
Epoch 27: Train err: 0.37525, Train loss: 0.6503791484832764 |Validation err: 0.3875, Validation
loss: 0.652014534920454
Epoch 28: Train err: 0.371375, Train loss: 0.6476589822769165 |Validation err: 0.3875, Validatio
n loss: 0.6483639199286699
Epoch 29: Train err: 0.36825, Train loss: 0.6451378240585327 |Validation err: 0.382, Validation
loss: 0.6459472719579935
Epoch 30: Train err: 0.362625, Train loss: 0.6423516173362732 |Validation err: 0.379, Validation
loss: 0.6439454797655344
Finished Training
Total time elapsed: 98.45 seconds
```



Total time elapsed: 98.45 seconds (slightly longer time than learning_rate = 0.01: 97.39 seconds)

We can observe that the Validation error/loss closely follows the Training error/loss.

However, after 30 epoch, the final Training/Validation error only reaches ~0.38 and Training/Validation loss only reaches ~0.64.

Therefore, 30 epochs in this training corresponds to the first 5-10 epochs when the learning rate was 0.01.

This is because decreasing the learning rate makes the model learn and progress slower towards minimum error/loss.

Therefore, the model will take longer time to reach the optimal error/loss, but approximately same time to train (since same batch size = 64, same number of calculations need to be done, etc.)

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 [17]:
```
large_net = LargeNet()
train_net(large_net, learning_rate=0.1)
model_path = get_model_name("large", batch_size=64, learning_rate=0.1, epoch=29)
plot_training_curve(model_path)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.4295, Train loss: 0.6743808498382569 |Validation err: 0.3565, Validation l
oss: 0.6350402869284153
Epoch 2: Train err: 0.364375, Train loss: 0.6394594783782959 |Validation err: 0.364, Validation
loss: 0.6380155384540558
Epoch 3: Train err: 0.357125, Train loss: 0.6292509632110596 |Validation err: 0.357, Validation
loss: 0.6313505992293358
Epoch 4: Train err: 0.351375, Train loss: 0.6222558770179748 |Validation err: 0.3615, Validation
loss: 0.6581139843910933
Epoch 5: Train err: 0.3425, Train loss: 0.6109966540336609 |Validation err: 0.327, Validation lo
ss: 0.593532383441925
Epoch 6: Train err: 0.31975, Train loss: 0.5888414912223816 |Validation err: 0.3045, Validation
loss: 0.6088876193389297
Epoch 7: Train err: 0.31825, Train loss: 0.5887727000713349 |Validation err: 0.3255, Validation
loss: 0.6114899981766939
Epoch 8: Train err: 0.310125, Train loss: 0.5762689855098725 |Validation err: 0.358, Validation
loss: 0.6087798178195953
Epoch 9: Train err: 0.304625, Train loss: 0.5726798589229584 |Validation err: 0.32, Validation l
oss: 0.5841292049735785
Epoch 10: Train err: 0.297125, Train loss: 0.5647666845321655 |Validation err: 0.3185, Validatio
n loss: 0.5889163985848427
Epoch 11: Train err: 0.282, Train loss: 0.5454324214458466 |Validation err: 0.3085, Validation l
oss: 0.600019222125411
Epoch 12: Train err: 0.278625, Train loss: 0.5384270005226135 |Validation err: 0.2995, Validatio
n loss: 0.5790340006351471
Epoch 13: Train err: 0.274375, Train loss: 0.5309341120719909 |Validation err: 0.308, Validation
loss: 0.6213697791099548
Epoch 14: Train err: 0.26975, Train loss: 0.5229758756160736 |Validation err: 0.308, Validation
loss: 0.6198801156133413
Epoch 15: Train err: 0.257625, Train loss: 0.5075466320514679 |Validation err: 0.312, Validation
loss: 0.6081355484202504
Epoch 16: Train err: 0.25625, Train loss: 0.5148655683994293 |Validation err: 0.327, Validation
loss: 0.6360824098810554
Epoch 17: Train err: 0.25475, Train loss: 0.5065258400440216 |Validation err: 0.3175, Validation
loss: 0.601697607897222
Epoch 18: Train err: 0.24575, Train loss: 0.49124235701560975 |Validation err: 0.3205, Validatio
n loss: 0.6378643093630672
Epoch 19: Train err: 0.259375, Train loss: 0.5141446299552918 |Validation err: 0.3385, Validatio
n loss: 0.6695924252271652
Epoch 20: Train err: 0.24875, Train loss: 0.4952861943244934 |Validation err: 0.3445, Validation
loss: 0.6670713275671005
Epoch 21: Train err: 0.243125, Train loss: 0.4907743418216705 |Validation err: 0.334, Validation
loss: 0.6669150963425636
Epoch 22: Train err: 0.23875, Train loss: 0.48334857559204103 |Validation err: 0.311, Validation
loss: 0.6853150613605976
Epoch 23: Train err: 0.223625, Train loss: 0.4652157464027405 |Validation err: 0.3145, Validatio
n loss: 0.7470605578273535
Epoch 24: Train err: 0.222875, Train loss: 0.4526974799633026 |Validation err: 0.33, Validation
loss: 0.7572380006313324
Epoch 25: Train err: 0.2345, Train loss: 0.4724377233982086 |Validation err: 0.334, Validation l
oss: 0.7064428050071001
Epoch 26: Train err: 0.234625, Train loss: 0.4725676612854004 |Validation err: 0.368, Validation
loss: 0.8023080676794052
Epoch 27: Train err: 0.22175, Train loss: 0.45844222140312196 |Validation err: 0.357, Validation
loss: 0.7886781524866819
Epoch 28: Train err: 0.237375, Train loss: 0.49102354764938355 |Validation err: 0.3375, Validati
on loss: 0.8998023774474859
Epoch 29: Train err: 0.237125, Train loss: 0.4870765597820282 |Validation err: 0.3385, Validatio
n loss: 0.8601288255304098
Epoch 30: Train err: 0.239375, Train loss: 0.4954684257507324 |Validation err: 0.358, Validation
loss: 0.8366707749664783
Finished Training
Total time elapsed: 103.31 seconds
```

Total time elapsed: 103.31 seconds (slightly longer time than learning rate = 0.01)

We can observe that the Validation error/loss diverges quickly from the Training error/loss. (overfit more easily)

Also, after about 5 epochs, both error and loss increases significantly. The minimum error/loss occurred around epoch 5.

Therefore, comparing the graphs, the first ~15 epochs in this training corresponds to the entire 30 epochs when the learning rate was 0.01.

This is because increasing the learning rate makes the model learn and progress faster towards minimum error/loss, and then away from it.

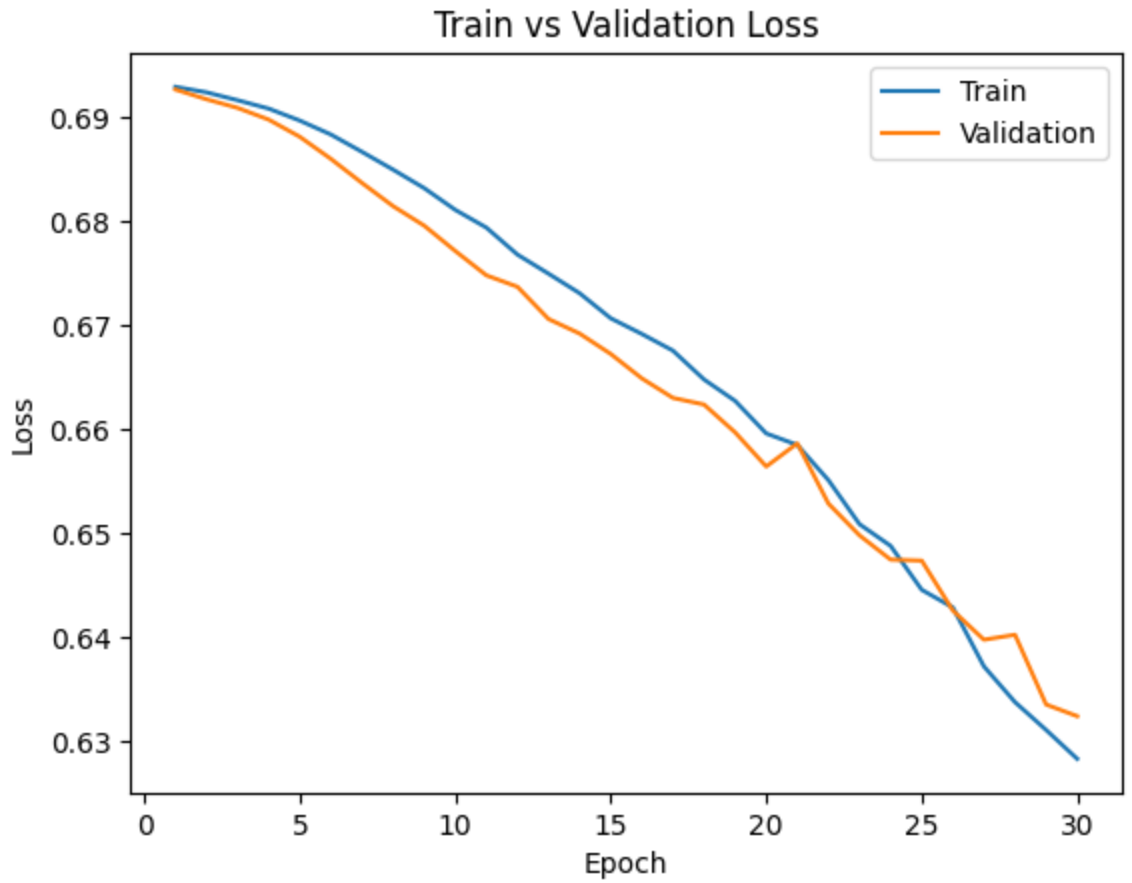
Therefore, the model will take shorter time to reach the optimal error/loss (quickly overfits), but approximately same time to train (since same batch size = 64, same number of calculations need to be done, etc.)

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 [18]:
```
large_net = LargeNet()
train_net(large_net, batch_size=512)
model_path = get_model_name("large", batch_size=512, learning_rate=0.01, epoch=29)
plot_training_curve(model_path)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.48175, Train loss: 0.6929379515349865 |Validation err: 0.478, Validation l
oss: 0.6926823854446411
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 lo
ss: 0.6909129917621613
Epoch 4: Train err: 0.433625, Train loss: 0.6908449940383434 |Validation err: 0.424, Validation
loss: 0.6897870153188705
Epoch 5: Train err: 0.434, Train loss: 0.6896935552358627 |Validation err: 0.424, Validation los
s: 0.6881355047225952
Epoch 6: Train err: 0.438, Train loss: 0.6883531995117664 |Validation err: 0.4285, Validation lo
ss: 0.686011865735054
Epoch 7: Train err: 0.439375, Train loss: 0.6866871826350689 |Validation err: 0.426, Validation
loss: 0.6836968660354614
Epoch 8: Train err: 0.43525, Train loss: 0.6849770732223988 |Validation err: 0.411, Validation l
oss: 0.6814672648906708
Epoch 9: Train err: 0.42375, Train loss: 0.6832008771598339 |Validation err: 0.414, Validation l
oss: 0.6795914322137833
Epoch 10: Train err: 0.421, Train loss: 0.6811087839305401 |Validation err: 0.416, Validation lo
ss: 0.677154079079628
Epoch 11: Train err: 0.42075, Train loss: 0.6794031001627445 |Validation err: 0.4095, Validation
loss: 0.6748126447200775
Epoch 12: Train err: 0.41475, Train loss: 0.6768064312636852 |Validation err: 0.412, Validation
loss: 0.6737105250358582
Epoch 13: Train err: 0.410375, Train loss: 0.6749707795679569 |Validation err: 0.412, Validation
loss: 0.6706106513738632
Epoch 14: Train err: 0.407125, Train loss: 0.6730907596647739 |Validation err: 0.4125, Validatio
n loss: 0.6692123562097549
Epoch 15: Train err: 0.400375, Train loss: 0.6706820242106915 |Validation err: 0.4105, Validatio
n loss: 0.6672529578208923
Epoch 16: Train err: 0.397625, Train loss: 0.6691752374172211 |Validation err: 0.405, Validation
loss: 0.6649040132761002
Epoch 17: Train err: 0.39375, Train loss: 0.6675728745758533 |Validation err: 0.4015, Validation
loss: 0.6630261093378067
Epoch 18: Train err: 0.392625, Train loss: 0.6647983901202679 |Validation err: 0.394, Validation
loss: 0.6623890697956085
Epoch 19: Train err: 0.386, Train loss: 0.6627401672303677 |Validation err: 0.388, Validation lo
ss: 0.6597096621990204
Epoch 20: Train err: 0.381625, Train loss: 0.6596098616719246 |Validation err: 0.4015, Validatio
n loss: 0.6564301252365112
Epoch 21: Train err: 0.386125, Train loss: 0.6584997698664665 |Validation err: 0.3885, Validatio
n loss: 0.6586556434631348
Epoch 22: Train err: 0.378625, Train loss: 0.6551188267767429 |Validation err: 0.385, Validation
loss: 0.6528644561767578
Epoch 23: Train err: 0.371625, Train loss: 0.6508826948702335 |Validation err: 0.3835, Validatio
n loss: 0.649808794260025
Epoch 24: Train err: 0.37675, Train loss: 0.6487980298697948 |Validation err: 0.384, Validation
loss: 0.6474767625331879
Epoch 25: Train err: 0.368375, Train loss: 0.6445756293833256 |Validation err: 0.382, Validation
loss: 0.6473795771598816
Epoch 26: Train err: 0.372875, Train loss: 0.6428777538239956 |Validation err: 0.3755, Validatio
n loss: 0.6425858736038208
Epoch 27: Train err: 0.35925, Train loss: 0.6372104585170746 |Validation err: 0.3785, Validation
loss: 0.6397744864225388
Epoch 28: Train err: 0.35425, Train loss: 0.6337734051048756 |Validation err: 0.37, Validation l
oss: 0.6402499973773956
Epoch 29: Train err: 0.3535, Train loss: 0.6310990080237389 |Validation err: 0.366, Validation l
oss: 0.6335441172122955
Epoch 30: Train err: 0.353, Train loss: 0.6283206455409527 |Validation err: 0.367, Validation lo
ss: 0.6324219256639481
Finished Training
Total time elapsed: 87.37 seconds
```



Total time elapsed: 87.37 seconds (shorter time than batch_size = 64). This model requires less time to train because smaller batch size means less iterations and calculations needed to finish each epoch.

The results by increasing the batch_size: the Validation error/loss closely follow that of Training. With higher batch size, the model proceeds less stochastic and steps more slowly towards the minimum error/loss (actually in this case, it have not even reached the minimum error/loss of batch_size = 64)

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 [19]:
```
large_net = LargeNet()
train_net(large_net, batch_size=16)
model_path = get_model_name("large", batch_size=16, learning_rate=0.01, epoch=29)
plot_training_curve(model_path)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.432625, Train loss: 0.6775506158471107 |Validation err: 0.378, Validation loss: 0.6512571625709533
Epoch 2: Train err: 0.3655, Train loss: 0.6395608481168747 |Validation err: 0.353, Validation loss: 0.6191524927616119
Epoch 3: Train err: 0.339875, Train loss: 0.6127814228534698 |Validation err: 0.3495, Validation loss: 0.6442944076061249
Epoch 4: Train err: 0.31475, Train loss: 0.5870258185863495 |Validation err: 0.3385, Validation loss: 0.5978640830516815
Epoch 5: Train err: 0.303, Train loss: 0.5651368154883385 |Validation err: 0.305, Validation loss: 0.5715659594535828
Epoch 6: Train err: 0.29025, Train loss: 0.5493593747615815 |Validation err: 0.302, Validation loss: 0.5801575357913971
Epoch 7: Train err: 0.26925, Train loss: 0.5272505192756652 |Validation err: 0.3005, Validation loss: 0.5818900580406189
Epoch 8: Train err: 0.254375, Train loss: 0.5104012369215488 |Validation err: 0.315, Validation loss: 0.5906455669403076
Epoch 9: Train err: 0.24675, Train loss: 0.4989313832819462 |Validation err: 0.2895, Validation loss: 0.5788275961875915
Epoch 10: Train err: 0.234625, Train loss: 0.4812071683704853 |Validation err: 0.2965, Validation loss: 0.58406099319458
Epoch 11: Train err: 0.229, Train loss: 0.4685571998953819 |Validation err: 0.292, Validation loss: 0.5905215789079666
Epoch 12: Train err: 0.22225, Train loss: 0.44902220405638216 |Validation err: 0.299, Validation loss: 0.6065930938720703
Epoch 13: Train err: 0.200625, Train loss: 0.4229303933084011 |Validation err: 0.2955, Validation loss: 0.6251139466762543
Epoch 14: Train err: 0.190375, Train loss: 0.4043408792465925 |Validation err: 0.291, Validation loss: 0.7069108620882034
Epoch 15: Train err: 0.17075, Train loss: 0.3758661460578442 |Validation err: 0.2985, Validation loss: 0.7291838767528533
Epoch 16: Train err: 0.169, Train loss: 0.36747281277179716 |Validation err: 0.3165, Validation loss: 0.7076600217819213
Epoch 17: Train err: 0.1525, Train loss: 0.34249910655617716 |Validation err: 0.3055, Validation loss: 0.8037151482105255
Epoch 18: Train err: 0.143125, Train loss: 0.3244629814103246 |Validation err: 0.312, Validation loss: 0.7955756268501282
Epoch 19: Train err: 0.14125, Train loss: 0.31422823867201805 |Validation err: 0.313, Validation loss: 0.8098329643011093
Epoch 20: Train err: 0.136875, Train loss: 0.3117204641513526 |Validation err: 0.3125, Validation loss: 0.788938661813736
Epoch 21: Train err: 0.114375, Train loss: 0.2696312564909458 |Validation err: 0.329, Validation loss: 0.9542041051387787
Epoch 22: Train err: 0.111, Train loss: 0.26029947843030093 |Validation err: 0.322, Validation loss: 0.9382650710344315
Epoch 23: Train err: 0.108375, Train loss: 0.2555502322986722 |Validation err: 0.323, Validation loss: 1.0157140600681305
Epoch 24: Train err: 0.1005, Train loss: 0.23835007748007775 |Validation err: 0.315, Validation loss: 1.109861346244812
Epoch 25: Train err: 0.094875, Train loss: 0.2270894316267222 |Validation err: 0.316, Validation loss: 1.1367672670185567
Epoch 26: Train err: 0.09325, Train loss: 0.2228318989351392 |Validation err: 0.318, Validation loss: 1.1843619964122771
Epoch 27: Train err: 0.093, Train loss: 0.23337606593593954 |Validation err: 0.33, Validation loss: 1.1684871008992195
Epoch 28: Train err: 0.08375, Train loss: 0.21251985063403844 |Validation err: 0.328, Validation loss: 1.2360798382759095
Epoch 29: Train err: 0.081625, Train loss: 0.2035546142552048 |Validation err: 0.315, Validation loss: 1.356388526082039
Epoch 30: Train err: 0.075875, Train loss: 0.19755484854709357 |Validation err: 0.3125, Validation loss: 1.4912220520973205
Finished Training
Total time elapsed: 159.68 seconds
```

Total time elapsed: 159.68 seconds. Training time is much longer, compared to batch_size = 64, since more iterations and calculations are needed to finish each epoch

Also, the model proceeds with much faster steps, because smaller batch size leads to more stochastivity, leading to the Validation error/loss easily diverges from that of Training (overfits very quickly), as we can also observe from the graphs

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.

We observed that decreasing learning rate from 0.01 to 0.001 and increasing batch_size from 64 to 512 results in Validation error/loss that closely follows that of Training (meaning the model improves overtime without overfitting to quickly)

Also, increasing batch size also reduces time required to train the model.

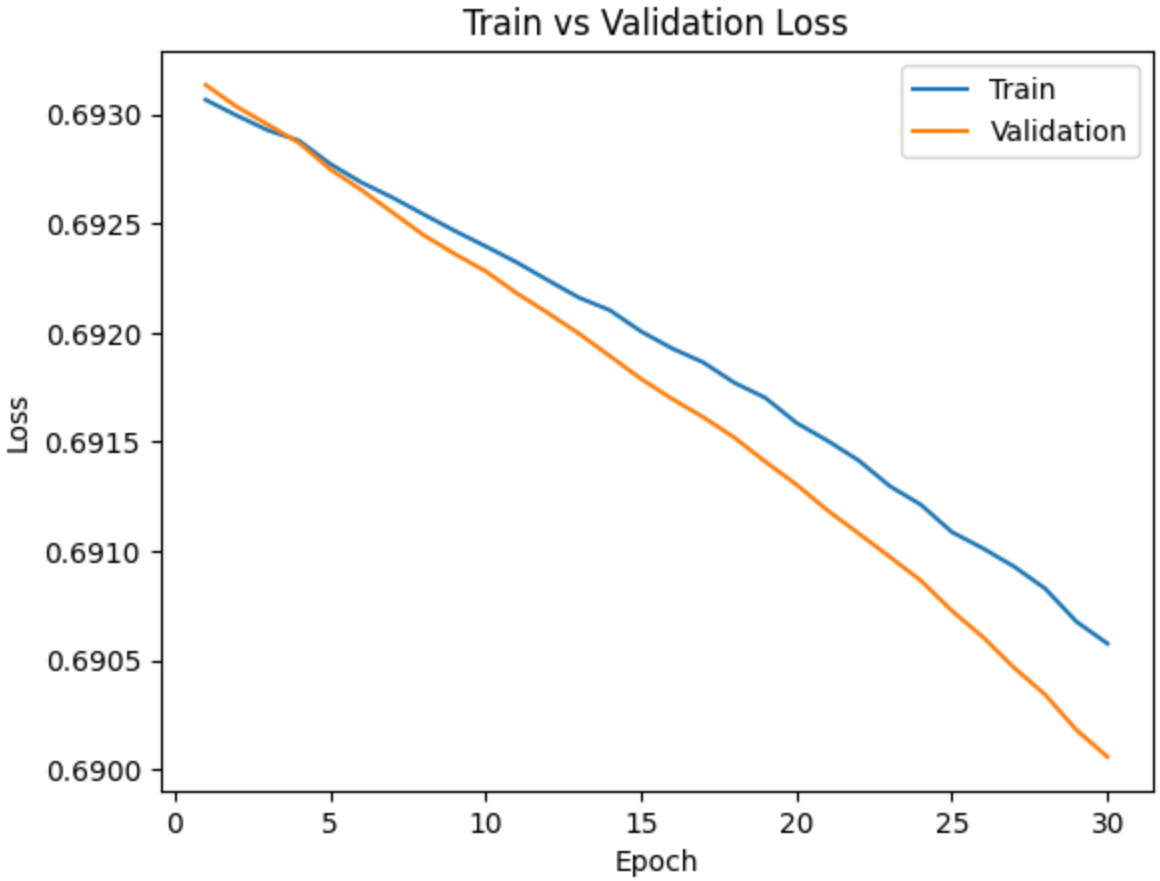
==> I choose (large_net, learning_rate = 0.01, batch_size = 512)

Part (b) - 1pt

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

In [20]:
```
large_net = LargeNet()
train_net(large_net, learning_rate=0.001, batch_size=512)
model_path = get_model_name("large", batch_size=512, learning_rate=0.001, epoch=29)
plot_training_curve(model_path)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.48825, Train loss: 0.6930677443742752 |Validation err: 0.4955, Validation loss: 0.6931362152099609
Epoch 2: Train err: 0.483125, Train loss: 0.692995510995388 |Validation err: 0.4945, Validation loss: 0.6930360496044159
Epoch 3: Train err: 0.480375, Train loss: 0.6929280534386635 |Validation err: 0.493, Validation loss: 0.6929539889097214
Epoch 4: Train err: 0.477, Train loss: 0.6928808465600014 |Validation err: 0.4885, Validation loss: 0.692870706319809
Epoch 5: Train err: 0.473375, Train loss: 0.692774411290884 |Validation err: 0.4835, Validation loss: 0.6927504986524582
Epoch 6: Train err: 0.469, Train loss: 0.6926896311342716 |Validation err: 0.472, Validation loss: 0.6926551759243011
Epoch 7: Train err: 0.46325, Train loss: 0.6926203593611717 |Validation err: 0.47, Validation loss: 0.6925524920225143
Epoch 8: Train err: 0.46225, Train loss: 0.6925435587763786 |Validation err: 0.463, Validation loss: 0.6924485266208649
Epoch 9: Train err: 0.459625, Train loss: 0.6924680285155773 |Validation err: 0.457, Validation loss: 0.6923621743917465
Epoch 10: Train err: 0.458, Train loss: 0.6923965662717819 |Validation err: 0.4555, Validation loss: 0.6922826170921326
Epoch 11: Train err: 0.454875, Train loss: 0.6923230774700642 |Validation err: 0.4505, Validation loss: 0.6921818554401398
Epoch 12: Train err: 0.4535, Train loss: 0.6922412514686584 |Validation err: 0.441, Validation loss: 0.6920914500951767
Epoch 13: Train err: 0.450375, Train loss: 0.6921614557504654 |Validation err: 0.437, Validation loss: 0.691996842622757
Epoch 14: Train err: 0.44725, Train loss: 0.692103236913681 |Validation err: 0.433, Validation loss: 0.6918932348489761
Epoch 15: Train err: 0.449375, Train loss: 0.6920064575970173 |Validation err: 0.432, Validation loss: 0.6917892247438431
Epoch 16: Train err: 0.44425, Train loss: 0.6919283643364906 |Validation err: 0.432, Validation loss: 0.6916972249746323
Epoch 17: Train err: 0.441375, Train loss: 0.6918644681572914 |Validation err: 0.431, Validation loss: 0.6916135102510452
Epoch 18: Train err: 0.438125, Train loss: 0.6917712353169918 |Validation err: 0.4295, Validation loss: 0.6915201395750046
Epoch 19: Train err: 0.436375, Train loss: 0.6917018294334412 |Validation err: 0.428, Validation loss: 0.6914086788892746
Epoch 20: Train err: 0.436375, Train loss: 0.6915871128439903 |Validation err: 0.4275, Validation loss: 0.69130440056324
Epoch 21: Train err: 0.437, Train loss: 0.6915052197873592 |Validation err: 0.4285, Validation loss: 0.6911860406398773
Epoch 22: Train err: 0.438625, Train loss: 0.6914149634540081 |Validation err: 0.428, Validation loss: 0.6910803616046906
Epoch 23: Train err: 0.436875, Train loss: 0.6912974379956722 |Validation err: 0.428, Validation loss: 0.6909734308719635
Epoch 24: Train err: 0.436875, Train loss: 0.6912120543420315 |Validation err: 0.425, Validation loss: 0.6908644735813141
Epoch 25: Train err: 0.435125, Train loss: 0.6910865269601345 |Validation err: 0.4255, Validation loss: 0.6907256692647934
Epoch 26: Train err: 0.434625, Train loss: 0.6910119280219078 |Validation err: 0.4245, Validation loss: 0.6906051337718964
Epoch 27: Train err: 0.43675, Train loss: 0.6909283213317394 |Validation err: 0.4265, Validation loss: 0.6904648691415787
Epoch 28: Train err: 0.43575, Train loss: 0.690827514976263 |Validation err: 0.4265, Validation loss: 0.6903413087129593
Epoch 29: Train err: 0.436375, Train loss: 0.6906765140593052 |Validation err: 0.423, Validation loss: 0.6901802867650986
Epoch 30: Train err: 0.435625, Train loss: 0.6905755028128624 |Validation err: 0.4235, Validation loss: 0.6900565475225449
Finished Training
Total time elapsed: 90.33 seconds
```



Part (c) - 2pt

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

I have also argued in Part 3 graphs that learning_rate = 0.001 and batch_size = 512 progresses more slowly towards to minimum error/loss, which results in a model that can possibly have not reached the most optimal error/loss. Therefore, we can also increase the number of epochs: epoch = 70 to ensure that we have not missed the optimal error/loss.

Part (d) - 1pt

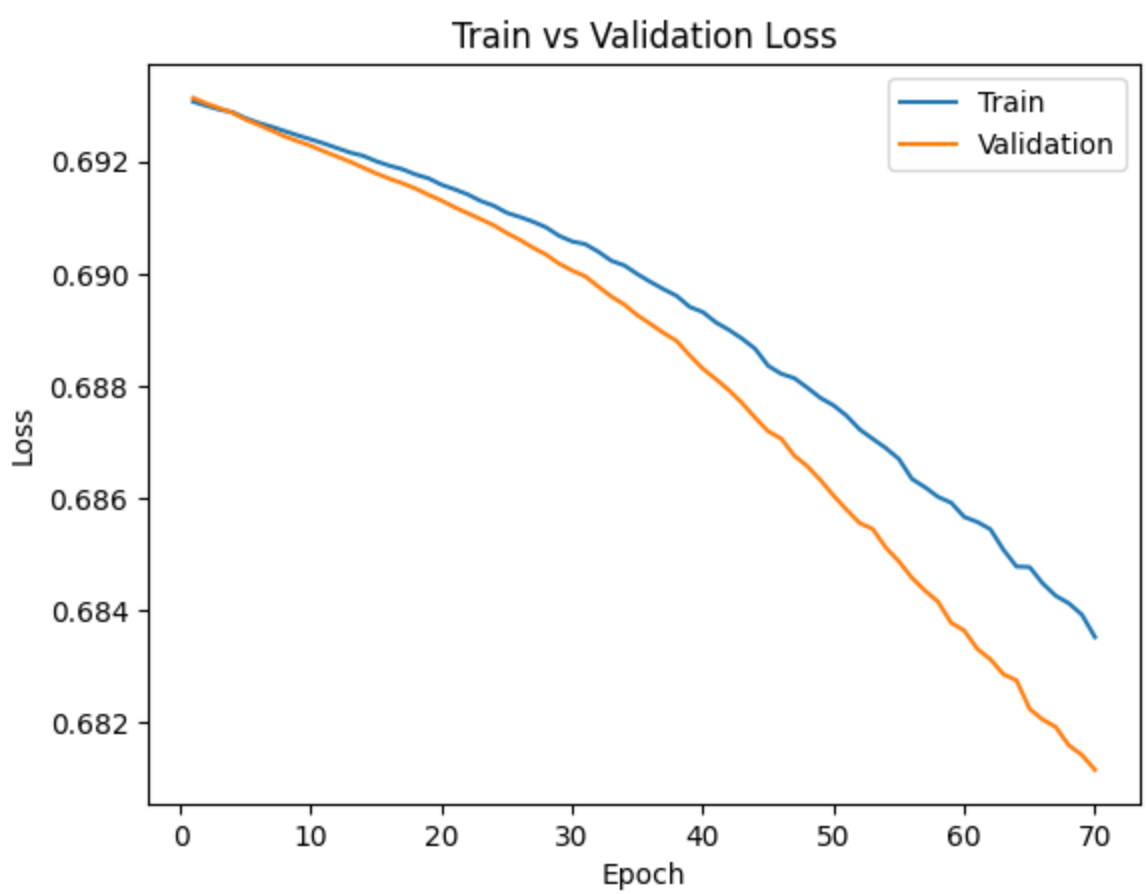
Train the model with the hyperparameters you chose in part(c), and include the training curve.

In [21]:
```
large_net = LargeNet()
train_net(large_net, learning_rate=0.001, batch_size=512, num_epochs=70)
model_path = get_model_name("large", batch_size=512, learning_rate=0.001, epoch=69)
plot_training_curve(model_path)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.48825, Train loss: 0.6930677443742752 |Validation err: 0.4955, Validation loss: 0.6931362152099609
Epoch 2: Train err: 0.483125, Train loss: 0.692995510995388 |Validation err: 0.4945, Validation loss: 0.6930360496044159
Epoch 3: Train err: 0.480375, Train loss: 0.6929280534386635 |Validation err: 0.493, Validation loss: 0.6929539889097214
Epoch 4: Train err: 0.477, Train loss: 0.6928808465600014 |Validation err: 0.4885, Validation loss: 0.692870706319809
Epoch 5: Train err: 0.473375, Train loss: 0.692774411290884 |Validation err: 0.4835, Validation loss: 0.6927504986524582
Epoch 6: Train err: 0.469, Train loss: 0.6926896311342716 |Validation err: 0.472, Validation loss: 0.6926551759243011
Epoch 7: Train err: 0.46325, Train loss: 0.6926203593611717 |Validation err: 0.47, Validation loss: 0.6925524920225143
Epoch 8: Train err: 0.46225, Train loss: 0.6925435587763786 |Validation err: 0.463, Validation loss: 0.6924485266208649
Epoch 9: Train err: 0.459625, Train loss: 0.6924680285155773 |Validation err: 0.457, Validation loss: 0.6923621743917465
Epoch 10: Train err: 0.458, Train loss: 0.6923965662717819 |Validation err: 0.4555, Validation loss: 0.6922826170921326
Epoch 11: Train err: 0.454875, Train loss: 0.6923230774700642 |Validation err: 0.4505, Validation loss: 0.6921818554401398
Epoch 12: Train err: 0.4535, Train loss: 0.6922412514686584 |Validation err: 0.441, Validation loss: 0.6920914500951767
Epoch 13: Train err: 0.450375, Train loss: 0.6921614557504654 |Validation err: 0.437, Validation loss: 0.691996842622757
Epoch 14: Train err: 0.44725, Train loss: 0.692103236913681 |Validation err: 0.433, Validation loss: 0.6918932348489761
Epoch 15: Train err: 0.449375, Train loss: 0.6920064575970173 |Validation err: 0.432, Validation loss: 0.6917892247438431
Epoch 16: Train err: 0.44425, Train loss: 0.6919283643364906 |Validation err: 0.432, Validation loss: 0.6916972249746323
Epoch 17: Train err: 0.441375, Train loss: 0.6918644681572914 |Validation err: 0.431, Validation loss: 0.6916135102510452
Epoch 18: Train err: 0.438125, Train loss: 0.6917712353169918 |Validation err: 0.4295, Validation loss: 0.6915201395750046
Epoch 19: Train err: 0.436375, Train loss: 0.6917018294334412 |Validation err: 0.428, Validation loss: 0.6914086788892746
Epoch 20: Train err: 0.436375, Train loss: 0.6915871128439903 |Validation err: 0.4275, Validation loss: 0.69130440056324
Epoch 21: Train err: 0.437, Train loss: 0.6915052197873592 |Validation err: 0.4285, Validation loss: 0.6911860406398773
Epoch 22: Train err: 0.438625, Train loss: 0.6914149634540081 |Validation err: 0.428, Validation loss: 0.6910803616046906
Epoch 23: Train err: 0.436875, Train loss: 0.6912974379956722 |Validation err: 0.428, Validation loss: 0.6909734308719635
Epoch 24: Train err: 0.436875, Train loss: 0.6912120543420315 |Validation err: 0.425, Validation loss: 0.6908644735813141
Epoch 25: Train err: 0.435125, Train loss: 0.6910865269601345 |Validation err: 0.4255, Validation loss: 0.6907256692647934
Epoch 26: Train err: 0.434625, Train loss: 0.6910119280219078 |Validation err: 0.4245, Validation loss: 0.6906051337718964
Epoch 27: Train err: 0.43675, Train loss: 0.6909283213317394 |Validation err: 0.4265, Validation loss: 0.6904648691415787
Epoch 28: Train err: 0.43575, Train loss: 0.690827514976263 |Validation err: 0.4265, Validation loss: 0.6903413087129593
Epoch 29: Train err: 0.436375, Train loss: 0.6906765140593052 |Validation err: 0.423, Validation loss: 0.6901802867650986
Epoch 30: Train err: 0.435625, Train loss: 0.6905755028128624 |Validation err: 0.4235, Validation loss: 0.6900565475225449
Epoch 31: Train err: 0.435625, Train loss: 0.6905272863805294 |Validation err: 0.424, Validation loss: 0.689955085515976
```

```
Epoch 32: Train err: 0.435375, Train loss: 0.6903932429850101 |Validation err: 0.4245, Validatio
n loss: 0.6897701174020767
Epoch 33: Train err: 0.434375, Train loss: 0.6902320273220539 |Validation err: 0.425, Validation
loss: 0.6895942836999893
Epoch 34: Train err: 0.434, Train loss: 0.6901473924517632 |Validation err: 0.423, Validation lo
ss: 0.6894484460353851
Epoch 35: Train err: 0.434, Train loss: 0.6899915412068367 |Validation err: 0.422, Validation lo
ss: 0.6892576664686203
Epoch 36: Train err: 0.433125, Train loss: 0.6898530721664429 |Validation err: 0.4205, Validatio
n loss: 0.6891040652990341
Epoch 37: Train err: 0.435875, Train loss: 0.6897234618663788 |Validation err: 0.42, Validation
loss: 0.6889433115720749
Epoch 38: Train err: 0.435125, Train loss: 0.6896037980914116 |Validation err: 0.421, Validation
loss: 0.6888010948896408
Epoch 39: Train err: 0.436625, Train loss: 0.6894076056778431 |Validation err: 0.4175, Validatio
n loss: 0.6885437667369843
Epoch 40: Train err: 0.435375, Train loss: 0.6893158853054047 |Validation err: 0.4195, Validatio
n loss: 0.6883071511983871
Epoch 41: Train err: 0.435125, Train loss: 0.6891321539878845 |Validation err: 0.42, Validation
loss: 0.6881202459335327
Epoch 42: Train err: 0.435125, Train loss: 0.6889970898628235 |Validation err: 0.418, Validation
loss: 0.687922477722168
Epoch 43: Train err: 0.4345, Train loss: 0.6888436712324619 |Validation err: 0.418, Validation l
oss: 0.6876961141824722
Epoch 44: Train err: 0.4355, Train loss: 0.6886618919670582 |Validation err: 0.4205, Validation
loss: 0.6874334514141083
Epoch 45: Train err: 0.434375, Train loss: 0.6883573643863201 |Validation err: 0.4185, Validatio
n loss: 0.6871903091669083
Epoch 46: Train err: 0.434625, Train loss: 0.6882153563201427 |Validation err: 0.4185, Validatio
n loss: 0.6870600134134293
Epoch 47: Train err: 0.43625, Train loss: 0.6881375536322594 |Validation err: 0.42, Validation l
oss: 0.686755359172821
Epoch 48: Train err: 0.4365, Train loss: 0.6879665888845921 |Validation err: 0.4205, Validation
loss: 0.6865663975477219
Epoch 49: Train err: 0.43625, Train loss: 0.6877851076424122 |Validation err: 0.419, Validation
loss: 0.686320960521698
Epoch 50: Train err: 0.437, Train loss: 0.6876524724066257 |Validation err: 0.4185, Validation l
oss: 0.6860434263944626
Epoch 51: Train err: 0.437125, Train loss: 0.687467310577631 |Validation err: 0.418, Validation
loss: 0.6857911497354507
Epoch 52: Train err: 0.43625, Train loss: 0.6872233375906944 |Validation err: 0.418, Validation
loss: 0.6855544894933701
Epoch 53: Train err: 0.436625, Train loss: 0.6870557740330696 |Validation err: 0.4185, Validatio
n loss: 0.6854472905397415
Epoch 54: Train err: 0.435625, Train loss: 0.6868905797600746 |Validation err: 0.4185, Validatio
n loss: 0.6851176470518112
Epoch 55: Train err: 0.433625, Train loss: 0.6866971030831337 |Validation err: 0.4165, Validatio
n loss: 0.6848706007003784
Epoch 56: Train err: 0.432125, Train loss: 0.6863408647477627 |Validation err: 0.416, Validation
loss: 0.6845775544643402
Epoch 57: Train err: 0.431875, Train loss: 0.6861926391720772 |Validation err: 0.4145, Validatio
n loss: 0.6843495815992355
Epoch 58: Train err: 0.43125, Train loss: 0.6860211864113808 |Validation err: 0.4145, Validation
loss: 0.6841496527194977
Epoch 59: Train err: 0.431, Train loss: 0.6859164014458656 |Validation err: 0.415, Validation lo
ss: 0.6837750822305679
Epoch 60: Train err: 0.43025, Train loss: 0.6856639347970486 |Validation err: 0.413, Validation
loss: 0.6836382895708084
Epoch 61: Train err: 0.43, Train loss: 0.6855737119913101 |Validation err: 0.4135, Validation lo
ss: 0.6833091527223587
Epoch 62: Train err: 0.4295, Train loss: 0.6854437664151192 |Validation err: 0.414, Validation l
oss: 0.683126226067543
Epoch 63: Train err: 0.428875, Train loss: 0.685077577829361 |Validation err: 0.4115, Validation
loss: 0.6828578859567642
```

```
Epoch 64: Train err: 0.4295, Train loss: 0.6847789660096169 |Validation err: 0.4135, Validation
loss: 0.6827497035264969
Epoch 65: Train err: 0.429875, Train loss: 0.6847690567374229 |Validation err: 0.4135, Validatio
n loss: 0.6822415143251419
Epoch 66: Train err: 0.4295, Train loss: 0.6844780519604683 |Validation err: 0.414, Validation l
oss: 0.6820503026247025
Epoch 67: Train err: 0.42925, Train loss: 0.6842574402689934 |Validation err: 0.4125, Validation
loss: 0.6819190680980682
Epoch 68: Train err: 0.428875, Train loss: 0.6841234676539898 |Validation err: 0.412, Validation
loss: 0.681591585278511
Epoch 69: Train err: 0.428375, Train loss: 0.6839216388761997 |Validation err: 0.41, Validation
loss: 0.6814204305410385
Epoch 70: Train err: 0.427375, Train loss: 0.6835209615528584 |Validation err: 0.4105, Validatio
n loss: 0.6811540722846985
Finished Training
Total time elapsed: 197.73 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 [27]:
```
net = large_net
model_path = get_model_name(net.name, batch_size=512, learning_rate=0.001, epoch=69)
state = torch.load(model_path)
net.load_state_dict(state)
```

Out[27]:
```
<All keys matched successfully>
```

Part (b) - 2pt

Justify your choice of model from part (a).

Based on all of the above arguments, I chose:

- large_net: Less fluctuations (more stable) compared to small_net. Even though it can overfits to higher error/loss, simply limitting learning_rate and adding overfit prevention methods will solve the issue
- batch_size = 512: less stochastic, progresses more slowly, preventing stepping too big steps and miss the potential minimum error/loss. Less likely to overfit.
- Learning_rate = 0.001: same reason as increased batch_size.

- epoch = 69: same argument as in Part3 c, and also by observation (both Validation and Training error and loss are decreasing linearly at that checkpoint). This is also close to the last checkpoint of the training process for this model.

Even though there are some checkpoints that give lower validation error/loss pair, this checkpoint is the most generalized (not overfitting), so it will be better to be used for testing

Accordingly, the Training and Validation error/loss from the best model checkpoint is:

- **Training error: 0.427375**
- **Training loss: 0.6835209615528584**
- **Validation error: 0.4105**
- **Validation loss: 0.6811540722846985**

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 [28]:
```
# 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=512)
criterion = nn.BCEWithLogitsLoss()
test_error, test_loss = evaluate(net, test_loader, criterion)
print("Test error: ", test_error, " and Test loss: ", test_loss)
```

```
Files already downloaded and verified
Files already downloaded and verified
Test error:  0.413  and Test loss:  0.6791730970144272
```

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.

Apparently, the test error (0.413) is **slightly higher than** validation error (0.4105). The difference is not significant because in part b, we intentionally chose a more generalized model checkpoint as the **best** model.

However, in general, I would expect the test error to be higher than the validation error because the model checkpoint chosen was based on the model's performance on validation dataset. The model has never seen (not trained on, and not chosen according to) the test data, so the model is more likely to behave worse to any randomness in the test data.

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?

Because how the model behaves to unseen data like test data is also how the model behaves to user data when it's applied in the real world. Therefore, using this method will better evaluate the model performance.

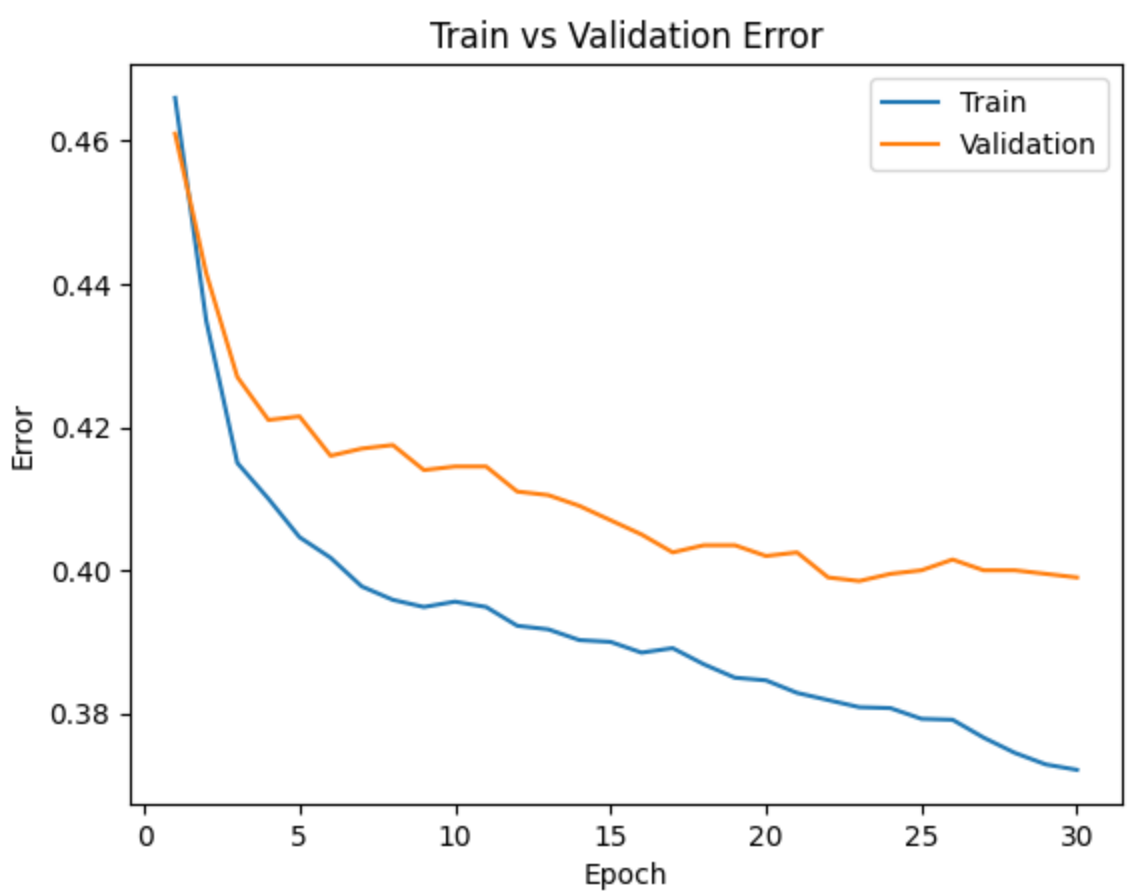
Part (f) - 5pt

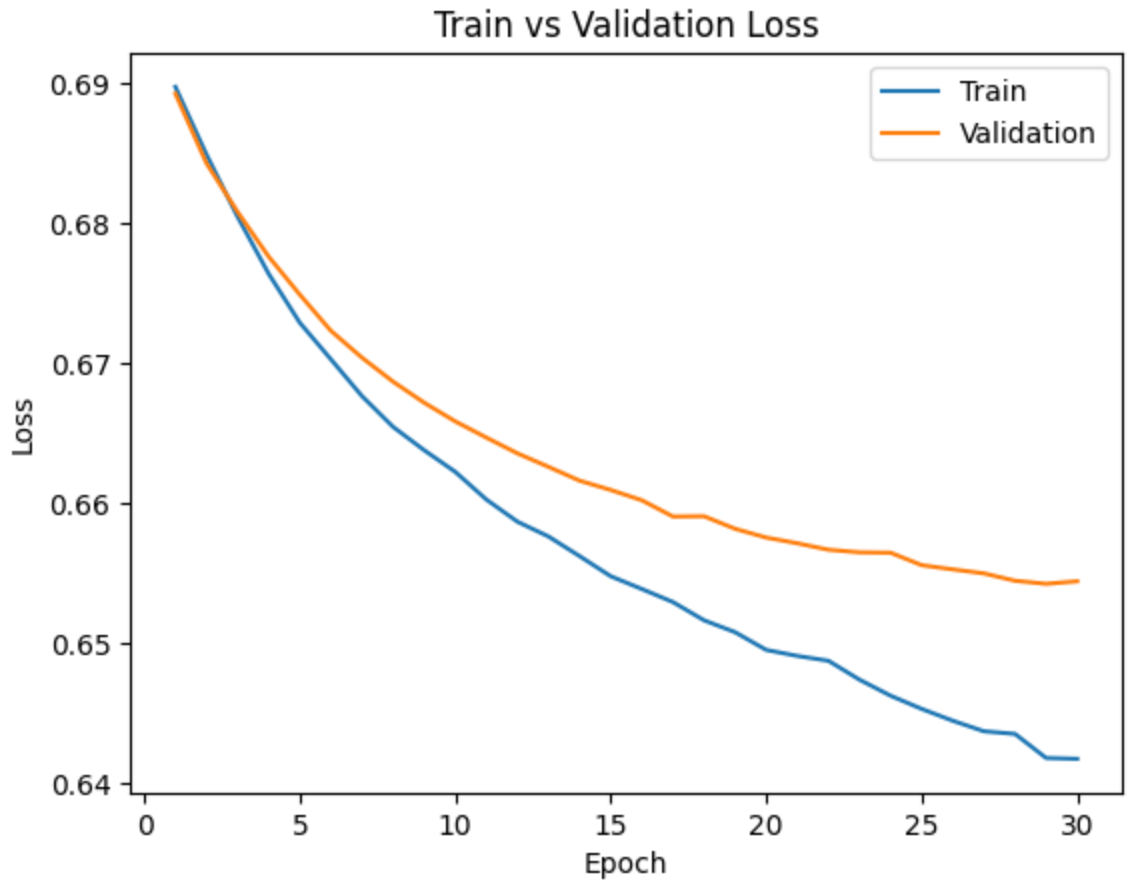
How does the your best CNN model compare with an 2-layer ANN model (no convolutional layers) on classifying cat and dog images. You can use a 2-layer ANN architecture similar to what you used in Lab 1. You should explore different hyperparameter settings to determine how well you can do on the validation 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 [29]:
```
class ANN(nn.Module):
    def __init__(self):
        super(ANN, self).__init__()
        self.name = "ANN"
        self.layer1 = nn.Linear(3 * 32 * 32, 32)
        self.layer2 = nn.Linear(32, 1)
    def forward(self, img):
        flattened = img.view(-1, 3 * 32 * 32)
        activation1 = self.layer1(flattened)
        activation1 = F.relu(activation1)
        activation2 = self.layer2(activation1)
        activation2 = activation2.squeeze(1)
        return activation2

ann = ANN()
train_net(ann, batch_size=512, learning_rate=0.001, num_epochs=30)
path = get_model_name("ANN", batch_size=512, learning_rate=0.001, epoch=29)
plot_training_curve(path)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.466, Train loss: 0.6897092722356319 |Validation err: 0.461, Validation los
s: 0.6892299652099609
Epoch 2: Train err: 0.434875, Train loss: 0.6848809234797955 |Validation err: 0.4415, Validation
loss: 0.684259295463562
Epoch 3: Train err: 0.415, Train loss: 0.680446021258831 |Validation err: 0.427, Validation los
s: 0.680788055062294
Epoch 4: Train err: 0.41, Train loss: 0.6763757988810539 |Validation err: 0.421, Validation los
s: 0.6776017099618912
Epoch 5: Train err: 0.404625, Train loss: 0.6728910319507122 |Validation err: 0.4215, Validation
loss: 0.6748959869146347
Epoch 6: Train err: 0.40175, Train loss: 0.6702827922999859 |Validation err: 0.416, Validation l
oss: 0.6723148375749588
Epoch 7: Train err: 0.39775, Train loss: 0.6676570661365986 |Validation err: 0.417, Validation l
oss: 0.6703968644142151
Epoch 8: Train err: 0.395875, Train loss: 0.6654616594314575 |Validation err: 0.4175, Validation
loss: 0.6686868071556091
Epoch 9: Train err: 0.394875, Train loss: 0.6637987270951271 |Validation err: 0.414, Validation
loss: 0.6671857088804245
Epoch 10: Train err: 0.395625, Train loss: 0.6622541137039661 |Validation err: 0.4145, Validatio
n loss: 0.6658536195755005
Epoch 11: Train err: 0.394875, Train loss: 0.6602800451219082 |Validation err: 0.4145, Validatio
n loss: 0.6646965593099594
Epoch 12: Train err: 0.39225, Train loss: 0.6586988978087902 |Validation err: 0.411, Validation
loss: 0.6635714918375015
Epoch 13: Train err: 0.39175, Train loss: 0.657641101628542 |Validation err: 0.4105, Validation
loss: 0.6626084297895432
Epoch 14: Train err: 0.39025, Train loss: 0.6562410667538643 |Validation err: 0.409, Validation
loss: 0.6616232693195343
Epoch 15: Train err: 0.39, Train loss: 0.6548085547983646 |Validation err: 0.407, Validation los
s: 0.6609607934951782
Epoch 16: Train err: 0.3885, Train loss: 0.653897862881422 |Validation err: 0.405, Validation lo
ss: 0.6602273136377335
Epoch 17: Train err: 0.389125, Train loss: 0.6529676355421543 |Validation err: 0.4025, Validatio
n loss: 0.6590670496225357
Epoch 18: Train err: 0.386875, Train loss: 0.6516681015491486 |Validation err: 0.4035, Validatio
n loss: 0.659086287021637
Epoch 19: Train err: 0.385, Train loss: 0.6508176028728485 |Validation err: 0.4035, Validation l
oss: 0.658190980553627
Epoch 20: Train err: 0.384625, Train loss: 0.6495539620518684 |Validation err: 0.402, Validation
loss: 0.6575670093297958
Epoch 21: Train err: 0.382875, Train loss: 0.6491261869668961 |Validation err: 0.4025, Validatio
n loss: 0.6571594327688217
Epoch 22: Train err: 0.381875, Train loss: 0.6487804986536503 |Validation err: 0.399, Validation
loss: 0.6566989421844482
Epoch 23: Train err: 0.380875, Train loss: 0.6474331356585026 |Validation err: 0.3985, Validatio
n loss: 0.6565130949020386
Epoch 24: Train err: 0.38075, Train loss: 0.64628741517663 |Validation err: 0.3995, Validation l
oss: 0.6564863473176956
Epoch 25: Train err: 0.37925, Train loss: 0.6453480049967766 |Validation err: 0.4, Validation lo
ss: 0.6555974334478378
Epoch 26: Train err: 0.379125, Train loss: 0.6445040851831436 |Validation err: 0.4015, Validatio
n loss: 0.6552984565496445
Epoch 27: Train err: 0.376625, Train loss: 0.6437529996037483 |Validation err: 0.4, Validation l
oss: 0.6550122648477554
Epoch 28: Train err: 0.3745, Train loss: 0.6435665674507618 |Validation err: 0.4, Validation los
s: 0.6544874906539917
Epoch 29: Train err: 0.372875, Train loss: 0.6418486796319485 |Validation err: 0.3995, Validatio
n loss: 0.6542681902647018
Epoch 30: Train err: 0.372125, Train loss: 0.641788013279438 |Validation err: 0.399, Validation
loss: 0.6544489562511444
Finished Training
Total time elapsed: 84.53 seconds
```





In [31]:
```
net = ann
path = get_model_name("ANN", batch_size=512, learning_rate=0.001, epoch=3)  # Chose 3 because la
state = torch.load(path)
net.load_state_dict(state)

train_loader, val_loader, test_loader, classes = get_data_loader(
    target_classes=["cat", "dog"],
```

```
    batch_size=512)
criterion = nn.BCEWithLogitsLoss()
test_error, test_loss = evaluate(net, test_loader, criterion)
print("ANN: Test error: ", test_error, " and Test loss: ", test_loss)
```

```
Files already downloaded and verified
Files already downloaded and verified
ANN: Test error:  0.4075  and Test loss:  0.6746410578489304
```

Even though the ANN test error (0.4075) is smaller than that of the CNN (0.413) given the same batch_size and learning_rate, this ANN checkpoint has to be chosen at epoch = 3 as later epochs overfits quickly. Therefore, the ANN behaves worse than CNN