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://colab.research.google.com/drive/1P-xR_wo-ZZwl1M6pLK0Kg92dcizAM2m5?usp=sharing (https://colab.research.google.com/drive/1P-xR_wo-ZZwl1M6pLK0Kg92dcizAM2m5?usp=sharing)

In [3]:

```
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 [4]:

```
######################################################################
#########
# 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 preproces
sed
    datasets.

    Args:
        target_classes: A list of strings denoting the name of the des
ired
                        classes. Should be a subset of the argument 'c
lasses'
        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 b
atch size
        classes: A list of strings denoting the name of each class
    """

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

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

    # split into training and validation indices
    relevant_train_indices, relevant_val_indices = relevant_indices[:split], relevant_indices[split:]
    train_sampler = SubsetRandomSampler(relevant_train_indices)
    train_loader = torch.utils.data.DataLoader(trainset, batch_size=batch_size,
                                               num_workers=1,
                                               sampler=train_sampler)
    val_sampler = SubsetRandomSampler(relevant_val_indices)
    val_loader = torch.utils.data.DataLoader(trainset, batch_size=batch_size,
                                             num_workers=1,
                                             sampler=val_sampler)
    # Load CIFAR10 testing data
    testset = torchvision.datasets.CIFAR10(root='./data', train=False,
                                           download=True, transform=transform)
    # Get the list of indices to sample from
    relevant_test_indices = get_relevant_indices(testset, classes,
                                                 target_classes)
    test_sampler = SubsetRandomSampler(relevant_test_indices)
    test_loader = torch.utils.data.DataLoader(testset, batch_size=batch_size,
                                             num_workers=1,
                                              sampler=test_sampler)


    return train_loader, val_loader, test_loader, classes

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

    Args:
        config: Configuration object containing the hyperparameters
    Returns:
        path: A string with the hyperparameter name and value concaten
ated
    """
    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 valid
ation set
         loss: A scalar for the average loss function over the validat
ion 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 [6]: # 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
```

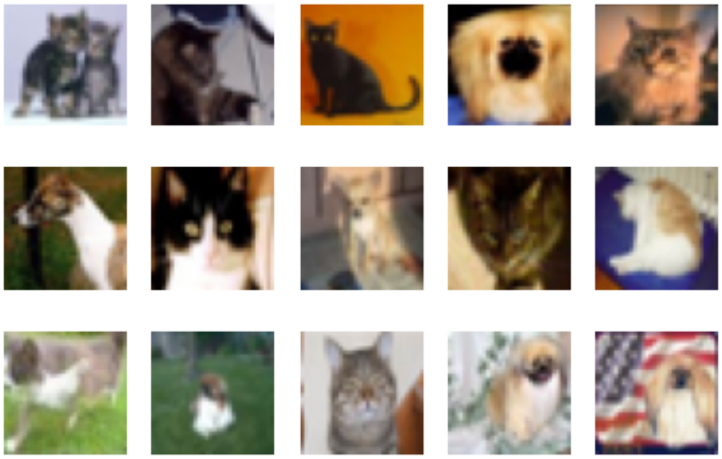
```
Files already downloaded and verified
Files already downloaded and verified
```

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 [7]: 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 [8]:
```
print("The number of training examples", len(train_loader))
print("The number of validation examples",len(val_loader))
print("The number of testing examples",len(test_loader))
```

```
The number of training examples 8000
The number of validation examples 2000
The number of testing examples 2000
```

Part (c) -- 3pt

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

Answer:

Because validation set is a separate data set not used for training. We validation set to track validation accuracy in the training curve and make decision about model hyperparameters and to avoid overfitting.

If we judge the performance of our models using the training set loss/error instead of the validation set loss/error, the model accuracy will be wrong since the model already trained by training set many time, the training set may cause overfit, and the result is unrealiable.

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 [9]:

```
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 [10]:

```
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 [11]:

```
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?)

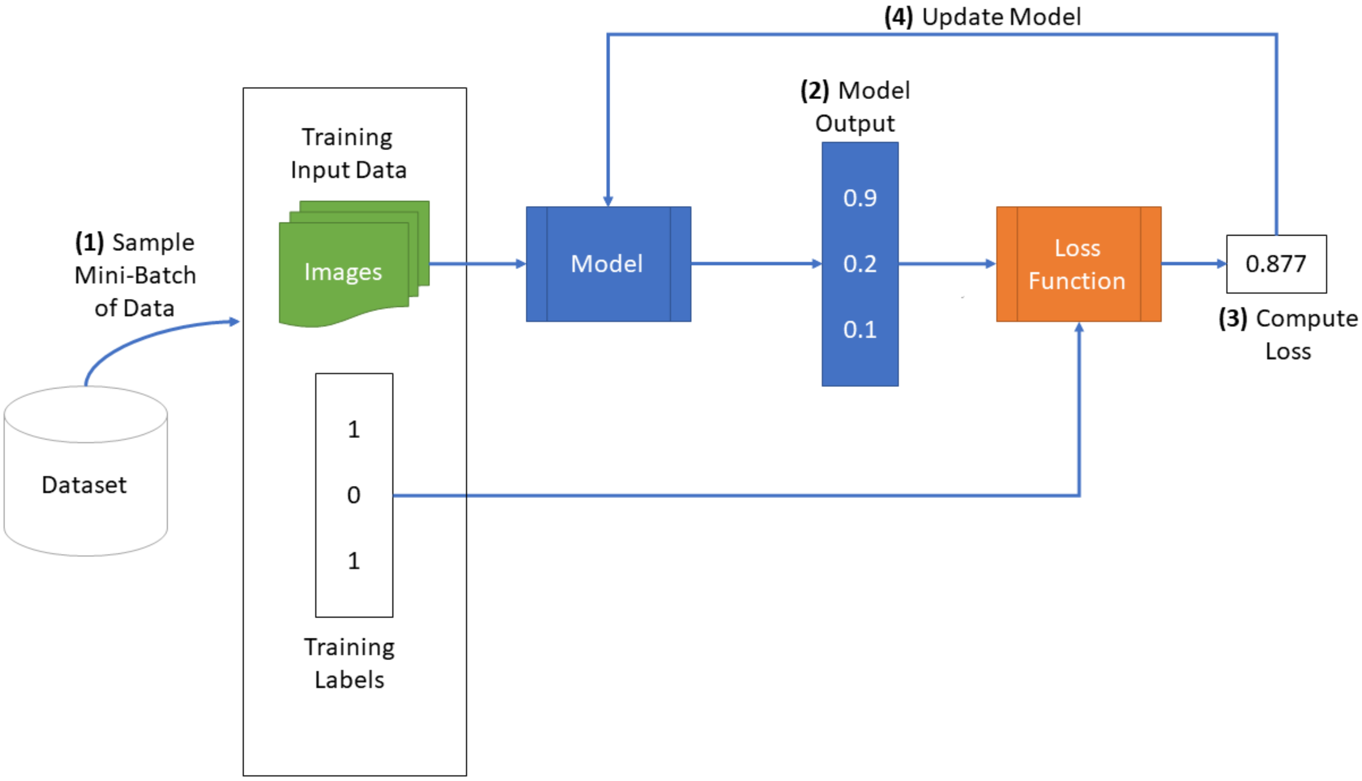
In [12]:

```
num_small_para=0;
for param in small_net.parameters():
    print(param.shape)
    num_small_para+=param.numel()

num_large_para=0;
for param in large_net.parameters():
    print(param.shape)
    num_large_para+=param.numel()

print("The total number of parameters in small_net:",num_small_para)
print("The total number of parameters in large_net:",num_large_para)
```

```
torch.Size([5, 3, 3, 3])
torch.Size([5])
torch.Size([1, 245])
torch.Size([1])
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 small_net: 386
The total number of parameters in large_net: 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 [13]: 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 da
         tasets
             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 ca
se we
    # will use the BCEWithLogitsLoss which takes unnormalized output f
rom
    # 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/errurac
y
    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 d
ata
    # 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, cr
iterion)
```

```
        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_rat
e, 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 [17]:
```
print("The default values: batch_size=64, learning_rate=0.01, num_epoc
hs=30")
```

```
The default values: 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.

In [39]:

```
train_net(small_net, batch_size=64, learning_rate=0.01, num_epochs=5)
print("\nList of all the files written to disk:")
print("model_small_bs64_lr0.01_epoch0")
print("model_small_bs64_lr0.01_epoch1")
print("model_small_bs64_lr0.01_epoch2")
print("model_small_bs64_lr0.01_epoch3")
print("model_small_bs64_lr0.01_epoch4\n")
print("model_small_bs64_lr0.01_epoch4_train_err.csv")
print("model_small_bs64_lr0.01_epoch4_train_loss.csv")

print("model_small_bs64_lr0.01_epoch4_val_err.csv")
print("model_small_bs64_lr0.01_epoch4_val_loss.csv")

print("\nThe files contain train err, train loss, validation err, ")
print("validation loss of each epoch")
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.41575, Train loss: 0.672908959388733 |Validati
on err: 0.3695, Validation loss: 0.6559448726475239
Epoch 2: Train err: 0.357875, Train loss: 0.6406905045509338 |Valida
tion err: 0.3675, Validation loss: 0.6474533956497908
Epoch 3: Train err: 0.340875, Train loss: 0.6181496586799622 |Valida
tion err: 0.334, Validation loss: 0.6119588389992714
Epoch 4: Train err: 0.31925, Train loss: 0.6020886974334717 |Validat
ion err: 0.3325, Validation loss: 0.6103744097054005
Epoch 5: Train err: 0.31025, Train loss: 0.5919639644622803 |Validat
ion err: 0.32, Validation loss: 0.6043383814394474
Finished Training
Total time elapsed: 19.62 seconds

List of all the files written to disk:
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

model_small_bs64_lr0.01_epoch4_train_err.csv
model_small_bs64_lr0.01_epoch4_train_loss.csv
model_small_bs64_lr0.01_epoch4_val_err.csv
model_small_bs64_lr0.01_epoch4_val_loss.csv

The files contain train err, train loss, validation err,
validation loss of each epoch
```

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 [18]:

```
# 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 [19]:

```
#@title Default title text
train_net(small_net,batch_size=64, learning_rate=0.01, num_epochs=30)
train_net(large_net,batch_size=64, learning_rate=0.01, num_epochs=30)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.304125, Train loss: 0.5830500652790069 |Valida
tion err: 0.325, Validation loss: 0.6061604805290699
Epoch 2: Train err: 0.295875, Train loss: 0.5755576927661896 |Valida
tion err: 0.327, Validation loss: 0.6087275417521596
Epoch 3: Train err: 0.299875, Train loss: 0.5708079364299774 |Valida
tion err: 0.3175, Validation loss: 0.5961393099278212
Epoch 4: Train err: 0.289625, Train loss: 0.5654956679344177 |Valida
tion err: 0.317, Validation loss: 0.6030140249058604
Epoch 5: Train err: 0.288875, Train loss: 0.5606383209228516 |Valida
tion err: 0.313, Validation loss: 0.5966839864850044
Epoch 6: Train err: 0.28575, Train loss: 0.5555275466442108 |Validat
ion err: 0.31, Validation loss: 0.5936574498191476
Epoch 7: Train err: 0.287875, Train loss: 0.5556862962245941 |Valida
tion err: 0.301, Validation loss: 0.5864428877830505
Epoch 8: Train err: 0.287, Train loss: 0.5510056962966919 |Validatio
n err: 0.312, Validation loss: 0.5864603249356151
Epoch 9: Train err: 0.283, Train loss: 0.5532685594558716 |Validatio
n err: 0.31, Validation loss: 0.5878112502396107
Epoch 10: Train err: 0.2765, Train loss: 0.5488754229545594 |Validat
```

```
ion err: 0.309, Validation loss: 0.5808900026604533
Epoch 11: Train err: 0.280375, Train loss: 0.5481656174659729 |Valid
ation err: 0.309, Validation loss: 0.5830764174461365
Epoch 12: Train err: 0.277, Train loss: 0.5436370315551757 |Validati
on err: 0.306, Validation loss: 0.5888135014101863
Epoch 13: Train err: 0.280625, Train loss: 0.5470758447647095 |Valid
ation err: 0.3065, Validation loss: 0.5825403435155749
Epoch 14: Train err: 0.269625, Train loss: 0.542404782295227 |Valida
tion err: 0.2995, Validation loss: 0.5940553247928619
Epoch 15: Train err: 0.27, Train loss: 0.5407245700359344 |Validatio
n err: 0.2995, Validation loss: 0.5767258359119296
Epoch 16: Train err: 0.27575, Train loss: 0.5459235084056854 |Valida
tion err: 0.304, Validation loss: 0.5948715079575777
Epoch 17: Train err: 0.274375, Train loss: 0.544366159439087 |Valida
tion err: 0.2935, Validation loss: 0.5771966995671391
Epoch 18: Train err: 0.2735, Train loss: 0.5375531506538391 |Validat
ion err: 0.2985, Validation loss: 0.5782867232337594
Epoch 19: Train err: 0.2695, Train loss: 0.5355135469436646 |Validat
ion err: 0.302, Validation loss: 0.5841771960258484
Epoch 20: Train err: 0.268625, Train loss: 0.5349176075458527 |Valid
ation err: 0.2875, Validation loss: 0.5824512429535389
Epoch 21: Train err: 0.273625, Train loss: 0.5363162324428559 |Valid
ation err: 0.285, Validation loss: 0.5706301713362336
Epoch 22: Train err: 0.272125, Train loss: 0.5369926867485046 |Valid
ation err: 0.3035, Validation loss: 0.5960945039987564
Epoch 23: Train err: 0.271625, Train loss: 0.5380362043380738 |Valid
ation err: 0.295, Validation loss: 0.581804346293211
Epoch 24: Train err: 0.267, Train loss: 0.5327217171192169 |Validati
on err: 0.296, Validation loss: 0.5835992498323321
Epoch 25: Train err: 0.266, Train loss: 0.5311209700107574 |Validati
on err: 0.287, Validation loss: 0.5770798213779926
Epoch 26: Train err: 0.266625, Train loss: 0.5318643038272858 |Valid
ation err: 0.2885, Validation loss: 0.5768095348030329
Epoch 27: Train err: 0.268875, Train loss: 0.5311221191883088 |Valid
ation err: 0.288, Validation loss: 0.5841397764161229
Epoch 28: Train err: 0.270625, Train loss: 0.534813241481781 |Valida
tion err: 0.287, Validation loss: 0.5734405657276511
Epoch 29: Train err: 0.265375, Train loss: 0.5314156522750855 |Valid
ation err: 0.296, Validation loss: 0.5938139781355858
Epoch 30: Train err: 0.268875, Train loss: 0.5365505158901215 |Valid
ation err: 0.29, Validation loss: 0.5808049160987139
Finished Training
Total time elapsed: 119.24 seconds
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.474, Train loss: 0.6924044041633606 |Validatio
n err: 0.417, Validation loss: 0.6879823822528124
Epoch 2: Train err: 0.427, Train loss: 0.6811256327629089 |Validatio
n err: 0.4355, Validation loss: 0.6823748182505369
Epoch 3: Train err: 0.4015, Train loss: 0.6651031966209412 |Validati
```

```
on err: 0.3675, Validation loss: 0.6487432643771172
Epoch 4: Train err: 0.362, Train loss: 0.640148277759552 |Validation
err: 0.36, Validation loss: 0.6381691172719002
Epoch 5: Train err: 0.34, Train loss: 0.6191210389137268 |Validation
err: 0.334, Validation loss: 0.6243907567113638
Epoch 6: Train err: 0.3185, Train loss: 0.5998364481925964 |Validati
on err: 0.323, Validation loss: 0.6123735681176186
Epoch 7: Train err: 0.305875, Train loss: 0.5856136772632599 |Valida
tion err: 0.315, Validation loss: 0.5995270507410169
Epoch 8: Train err: 0.299, Train loss: 0.5719028851985931 |Validatio
n err: 0.325, Validation loss: 0.6034662080928683
Epoch 9: Train err: 0.290125, Train loss: 0.5626473424434661 |Valida
tion err: 0.3085, Validation loss: 0.5798036847263575
Epoch 10: Train err: 0.281875, Train loss: 0.5478874287605285 |Valid
ation err: 0.2855, Validation loss: 0.5683204298838973
Epoch 11: Train err: 0.26975, Train loss: 0.539211356163025 |Validat
ion err: 0.3055, Validation loss: 0.5867745885625482
Epoch 12: Train err: 0.263125, Train loss: 0.5242250418663025 |Valid
ation err: 0.2905, Validation loss: 0.5690758535638452
Epoch 13: Train err: 0.255375, Train loss: 0.514204537153244 |Valida
tion err: 0.2845, Validation loss: 0.5721875810995698
Epoch 14: Train err: 0.24925, Train loss: 0.5007789981365204 |Valida
tion err: 0.2985, Validation loss: 0.5757667869329453
Epoch 15: Train err: 0.24625, Train loss: 0.50002916431427 |Validati
on err: 0.277, Validation loss: 0.582096628844738
Epoch 16: Train err: 0.240375, Train loss: 0.49279915285110476 |Vali
dation err: 0.295, Validation loss: 0.5699639432132244
Epoch 17: Train err: 0.233625, Train loss: 0.4792389583587646 |Valid
ation err: 0.293, Validation loss: 0.5623635230585933
Epoch 18: Train err: 0.2235, Train loss: 0.4629262320995331 |Validat
ion err: 0.295, Validation loss: 0.5742575116455555
Epoch 19: Train err: 0.2165, Train loss: 0.4566372413635254 |Validat
ion err: 0.284, Validation loss: 0.5722175063565373
Epoch 20: Train err: 0.210875, Train loss: 0.4425299654006958 |Valid
ation err: 0.319, Validation loss: 0.6777711343020201
Epoch 21: Train err: 0.2185, Train loss: 0.4491088275909424 |Validat
ion err: 0.28, Validation loss: 0.5763282468542457
Epoch 22: Train err: 0.1955, Train loss: 0.42160264134407044 |Valida
tion err: 0.2885, Validation loss: 0.6009478718042374
Epoch 23: Train err: 0.195625, Train loss: 0.41811202788352964 |Vali
dation err: 0.297, Validation loss: 0.576689139008522
Epoch 24: Train err: 0.18725, Train loss: 0.402196151971817 |Validat
ion err: 0.297, Validation loss: 0.6180002447217703
Epoch 25: Train err: 0.179625, Train loss: 0.3865300531387329 |Valid
ation err: 0.294, Validation loss: 0.6045780340209603
Epoch 26: Train err: 0.170625, Train loss: 0.377172299861908 |Valida
tion err: 0.2975, Validation loss: 0.6476951222866774
Epoch 27: Train err: 0.164875, Train loss: 0.3652070670127869 |Valid
ation err: 0.292, Validation loss: 0.6792074739933014
Epoch 28: Train err: 0.158375, Train loss: 0.34827687883377073 |Vali
```

```
dation err: 0.2885, Validation loss: 0.670931757427752
Epoch 29: Train err: 0.146375, Train loss: 0.3277204424142838 |Valid
ation err: 0.3075, Validation loss: 0.7807608842849731
Epoch 30: Train err: 0.144, Train loss: 0.3220942703485489 |Validati
on err: 0.308, Validation loss: 0.6827659532427788
Finished Training
Total time elapsed: 128.49 seconds
```

Answer:\ Total time elapsed for training small network is 119.24 seconds.\ Total time elapsed for training large network is 128.49 seconds.\ Large network takes longer to train.\ Because large network has more parameters than small network, such as conv2, fc2, which takes more time to calculate these parameters.

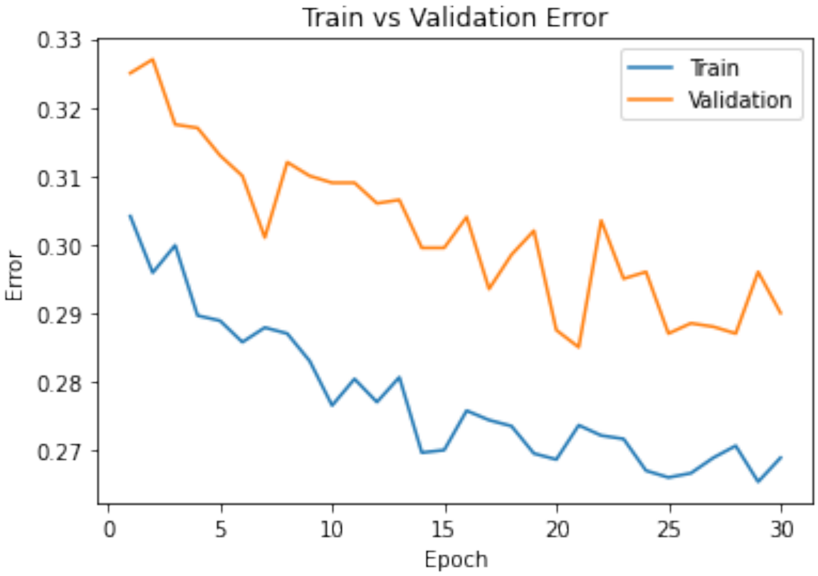
Part (e) - 2pt

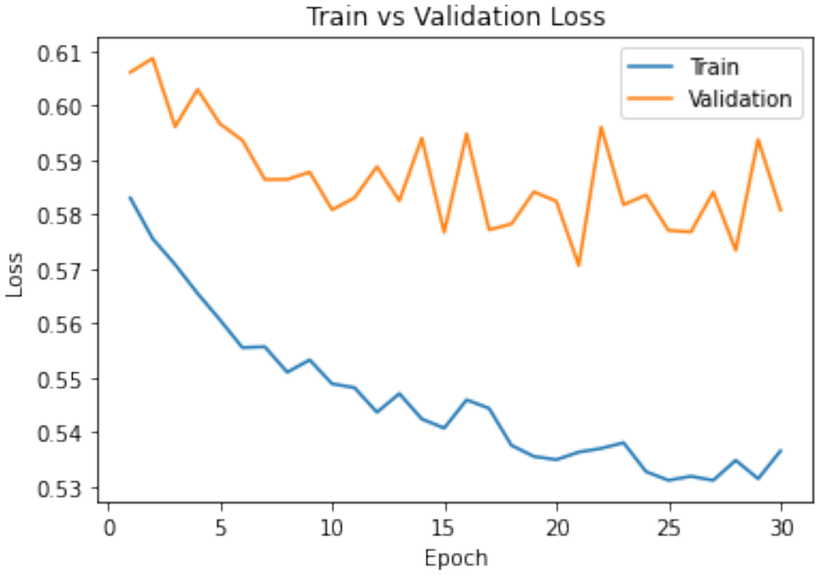
Use the function `plot_training_curve` to display the trajectory of the training/validation error and the training/validation loss. You will need to use the function `get_model_name` to generate the argument to the `plot_training_curve` function.

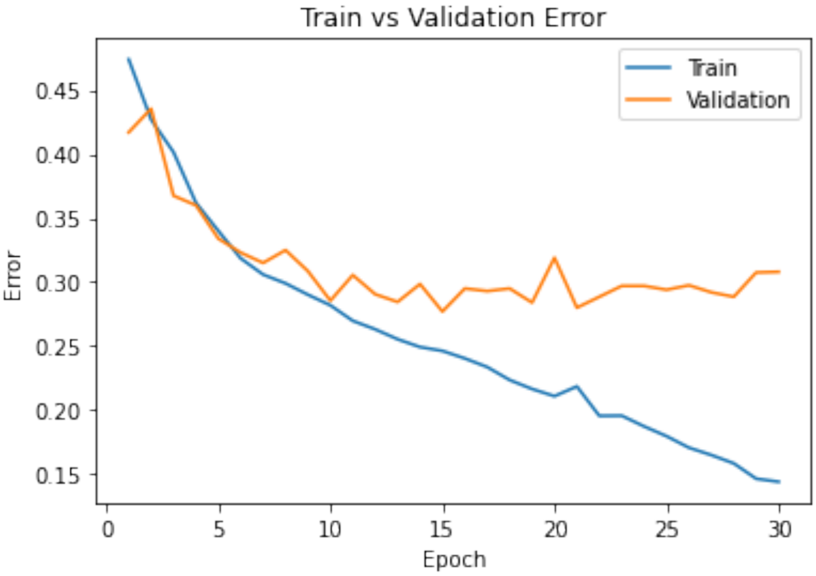
Do this for both the small network and the large network. Include both plots in your writeup.

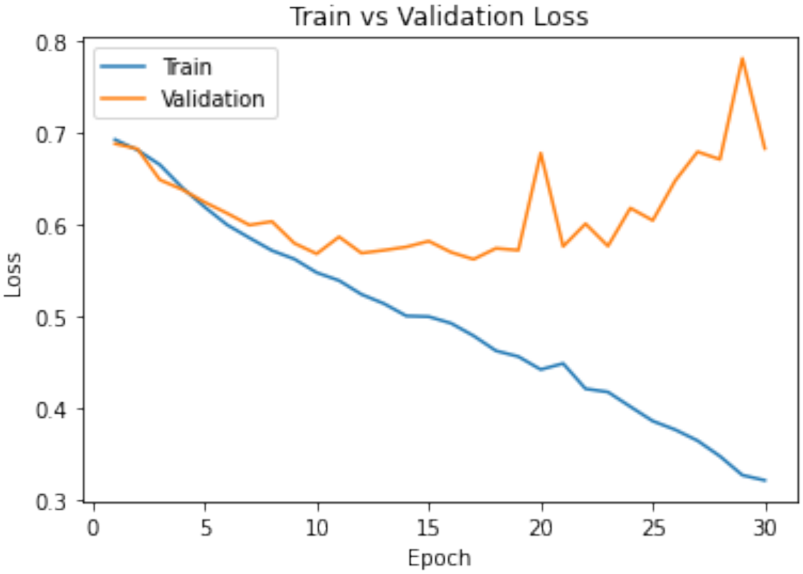
In [21]:

```
#model_path = get_model_name("small", batch_size=??, learning_rate=??,
epoch=29)
small_path_e = get_model_name("small",batch_size=64, learning_rate=0.0
1,
                              epoch=29)
large_path_e = get_model_name("large",batch_size=64, learning_rate=0.0
1,
                              epoch=29)
plot_training_curve(small_path_e)
plot_training_curve(large_path_e)
```


Train vs Validation Error
Error
0.33
0.32
0.31
0.30
0.29
0.28
0.27
Epoch
0
5
10
15
20
25
30
Train
Validation


Train vs Validation Loss
Loss
0.61
0.60
0.59
0.58
0.57
0.56
0.55
0.54
0.53
Epoch
0
5
10
15
20
25
30
Train
Validation





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.

**Answer **

The train error/loss curves for both small_net and large_net shows a downward trend, but for large_net is more smooth. The gap between train andvalidation curves for large_net are much smaller than small_net before 10 epoches. Both error and loss plot for large_net, the curves shows underfitting before 10 epoches, and shows overfitting after 23 epoches. The error plot for small_net shows overfitting after 20 epoches

```
In [ ]:
```

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.

Answer:

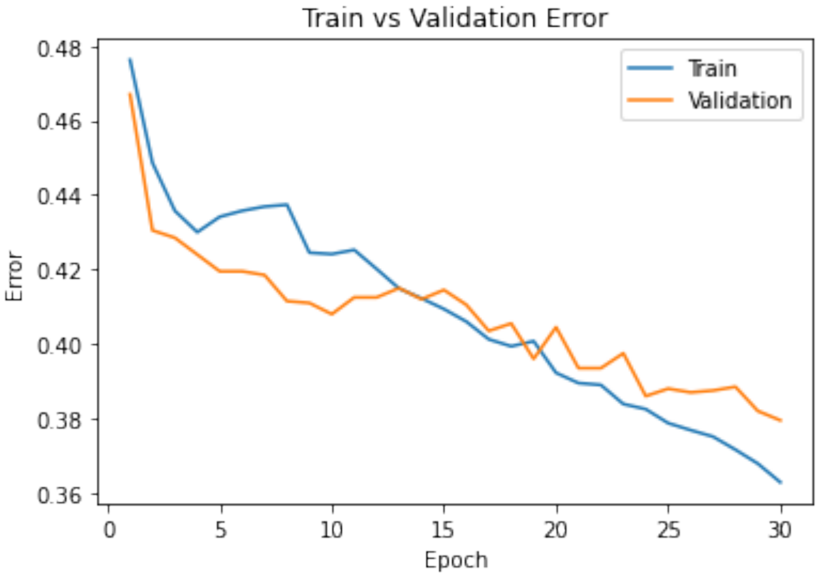
Total time elapsed is 128.92 seconds. The model takes longer to train. The gap between train and validation curve becomes smaller in both error/loss polts, especially over 20 epoches there is no overfitting, which means it is better fitted than the default learning_rate model. Thus, lowering the learning rate increases the traning time and increases validation accuracy.

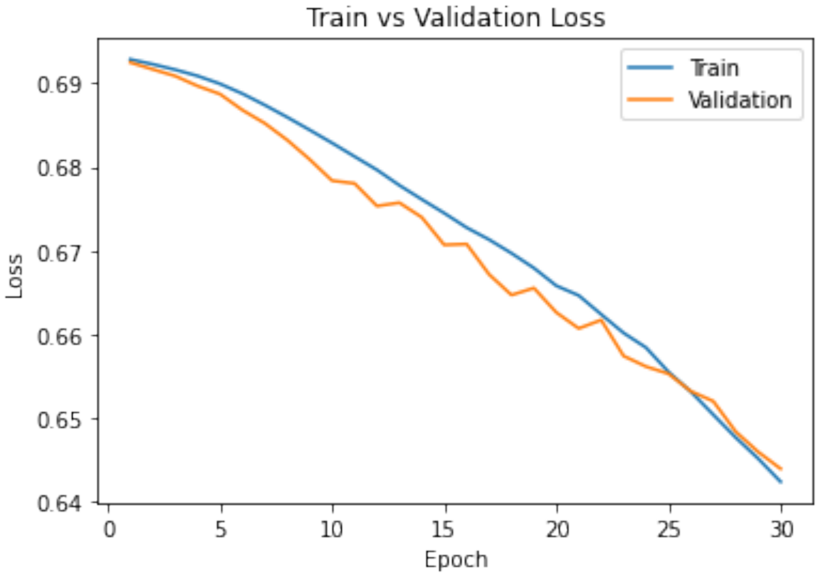
```
In [22]: # Note: When we re-construct the model, we start the training
         # with *random weights*. If we omit this code, the values of
         # the weights will still be the previously trained values.
         large_net = LargeNet()
         train_net(large_net, learning_rate=0.001)
         large_path = get_model_name("large",batch_size=64, learning_rate=0.001
         ,
                                      epoch=29)
         plot_training_curve(large_path)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.47625, Train loss: 0.6928360004425049 |Validat
ion err: 0.467, Validation loss: 0.692468661814928
Epoch 2: Train err: 0.448625, Train loss: 0.6922589693069457 |Valida
```

```
tion err: 0.4305, Validation loss: 0.6916493456810713
Epoch 3: Train err: 0.43575, Train loss: 0.6916067309379578 |Validat
ion err: 0.4285, Validation loss: 0.6908544600009918
Epoch 4: Train err: 0.43, Train loss: 0.6908613452911377 |Validation
err: 0.424, Validation loss: 0.6896595284342766
Epoch 5: Train err: 0.434125, Train loss: 0.6899194993972778 |Valida
tion err: 0.4195, Validation loss: 0.6886935140937567
Epoch 6: Train err: 0.43575, Train loss: 0.6887412719726562 |Validat
ion err: 0.4195, Validation loss: 0.686782693490386
Epoch 7: Train err: 0.436875, Train loss: 0.6873777813911438 |Valida
tion err: 0.4185, Validation loss: 0.6851987447589636
Epoch 8: Train err: 0.437375, Train loss: 0.68592657995224 |Validati
on err: 0.4115, Validation loss: 0.6831984482705593
Epoch 9: Train err: 0.4245, Train loss: 0.6844043183326721 |Validati
on err: 0.411, Validation loss: 0.6808853577822447
Epoch 10: Train err: 0.424125, Train loss: 0.6828489475250245 |Valid
ation err: 0.408, Validation loss: 0.6783470250666142
Epoch 11: Train err: 0.42525, Train loss: 0.6812356414794922 |Valida
tion err: 0.4125, Validation loss: 0.6780184432864189
Epoch 12: Train err: 0.420125, Train loss: 0.6796348176002502 |Valid
ation err: 0.4125, Validation loss: 0.6753157749772072
Epoch 13: Train err: 0.414875, Train loss: 0.6777922549247741 |Valid
ation err: 0.415, Validation loss: 0.675711577758193
Epoch 14: Train err: 0.41225, Train loss: 0.6761121478080749 |Valida
tion err: 0.412, Validation loss: 0.6739676017314196
Epoch 15: Train err: 0.409375, Train loss: 0.6744700741767883 |Valid
ation err: 0.4145, Validation loss: 0.670675216242671
Epoch 16: Train err: 0.406, Train loss: 0.6727381496429443 |Validati
on err: 0.4105, Validation loss: 0.6707698833197355
Epoch 17: Train err: 0.40125, Train loss: 0.6713080592155457 |Valida
tion err: 0.4035, Validation loss: 0.6671544443815947
Epoch 18: Train err: 0.399375, Train loss: 0.6696773543357849 |Valid
ation err: 0.4055, Validation loss: 0.6646812092512846
Epoch 19: Train err: 0.40075, Train loss: 0.6679102511405944 |Valida
tion err: 0.396, Validation loss: 0.6655160505324602
Epoch 20: Train err: 0.39225, Train loss: 0.6657903985977173 |Valida
tion err: 0.4045, Validation loss: 0.6626022979617119
Epoch 21: Train err: 0.3895, Train loss: 0.6646289625167847 |Validat
ion err: 0.3935, Validation loss: 0.6606861352920532
Epoch 22: Train err: 0.389, Train loss: 0.6623730616569519 |Validati
on err: 0.3935, Validation loss: 0.6616983562707901
Epoch 23: Train err: 0.383875, Train loss: 0.6601499290466308 |Valid
ation err: 0.3975, Validation loss: 0.6574018411338329
Epoch 24: Train err: 0.3825, Train loss: 0.6584016590118408 |Validat
ion err: 0.386, Validation loss: 0.6561368461698294
Epoch 25: Train err: 0.37875, Train loss: 0.6555012550354004 |Valida
tion err: 0.388, Validation loss: 0.6552800387144089
Epoch 26: Train err: 0.376875, Train loss: 0.653127950668335 |Valida
tion err: 0.387, Validation loss: 0.6531846430152655
Epoch 27: Train err: 0.375125, Train loss: 0.6503854460716247 |Valid
```

```
ation err: 0.3875, Validation loss: 0.6519918907433748
Epoch 28: Train err: 0.371625, Train loss: 0.6476585249900818 |Valid
ation err: 0.3885, Validation loss: 0.6483596488833427
Epoch 29: Train err: 0.367875, Train loss: 0.6451436839103699 |Valid
ation err: 0.382, Validation loss: 0.645940650254488
Epoch 30: Train err: 0.362875, Train loss: 0.6423715600967407 |Valid
ation err: 0.3795, Validation loss: 0.6439278181642294
Finished Training
Total time elapsed: 128.92 seconds
```





Part (b) - 3pt

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

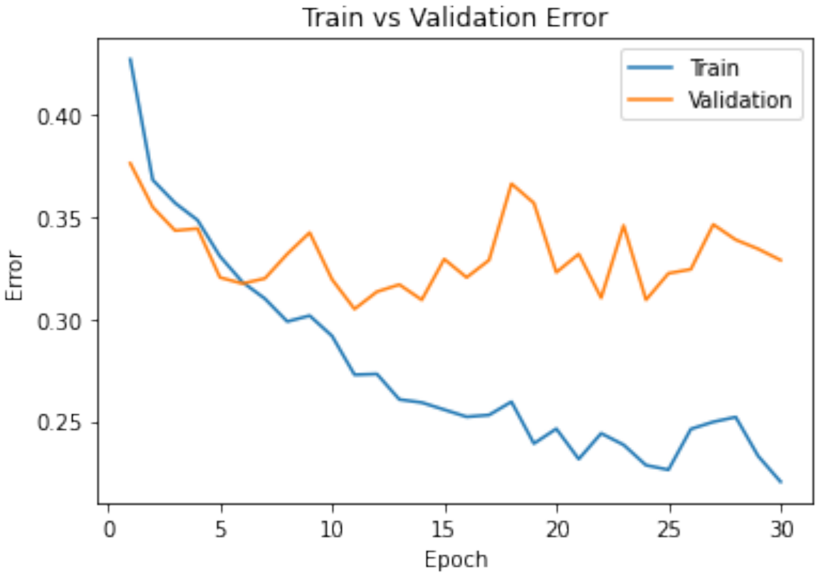
Answer:

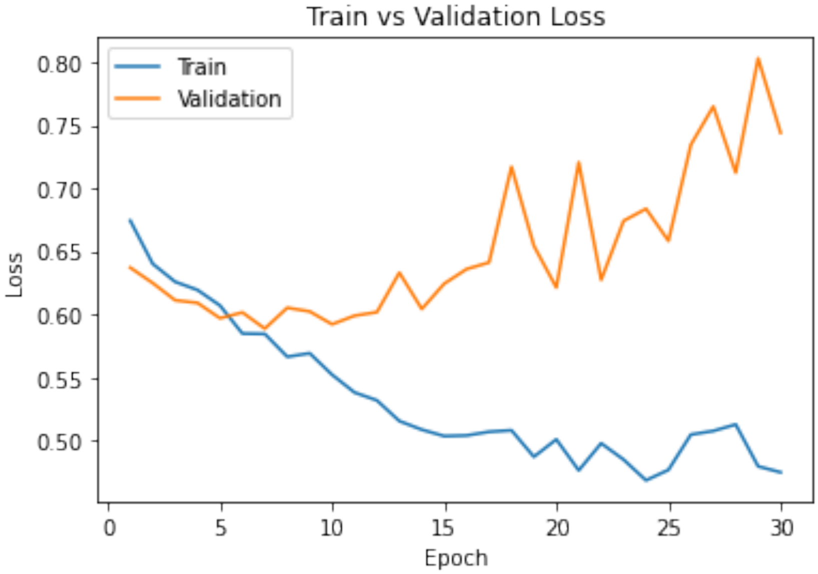
Total time elapsed is 127.77 seconds. The model takes shorter to train. From the plots, train and validation curve becomes overfitting over about 10 epoches in both error/loss polts, which is more early than the default learning_rate plots and result in divergence. Thus, increaseing the learning rate reduces the traning time and makes higher vallidation error/loss.

In [23]:
```
large_net = LargeNet()
train_net(large_net, learning_rate=0.1)
large_path2 = get_model_name("large",batch_size=64, learning_rate=0.1,
                             epoch=29)
plot_training_curve(large_path2)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.4275, Train loss: 0.6742977557182313 |Validati
on err: 0.3765, Validation loss: 0.637031726539135
Epoch 2: Train err: 0.368375, Train loss: 0.6400856471061707 |Valida
tion err: 0.355, Validation loss: 0.6248273365199566
Epoch 3: Train err: 0.357, Train loss: 0.6258103721141816 |Validatio
n err: 0.3435, Validation loss: 0.6111582862213254
Epoch 4: Train err: 0.348625, Train loss: 0.6193035736083984 |Valida
tion err: 0.3445, Validation loss: 0.6090600341558456
Epoch 5: Train err: 0.330875, Train loss: 0.6069964549541473 |Valida
tion err: 0.3205, Validation loss: 0.5968910921365023
Epoch 6: Train err: 0.31825, Train loss: 0.5847644207477569 |Validat
ion err: 0.3175, Validation loss: 0.6014995649456978
Epoch 7: Train err: 0.310125, Train loss: 0.5844504678249359 |Valida
tion err: 0.32, Validation loss: 0.5886365948244929
Epoch 8: Train err: 0.298875, Train loss: 0.5663281772136688 |Valida
tion err: 0.332, Validation loss: 0.6052035503089428
Epoch 9: Train err: 0.30175, Train loss: 0.5691309769153595 |Validat
ion err: 0.3425, Validation loss: 0.6022673770785332
Epoch 10: Train err: 0.29175, Train loss: 0.552112295856475 |Valida
tion err: 0.3195, Validation loss: 0.5920912651345134
Epoch 11: Train err: 0.27275, Train loss: 0.5380442805290222 |Valida
tion err: 0.305, Validation loss: 0.598898870870471
Epoch 12: Train err: 0.273125, Train loss: 0.531749195098877 |Valida
tion err: 0.3135, Validation loss: 0.6017276663333178
Epoch 13: Train err: 0.260625, Train loss: 0.515365345954895 |Valida
```

```
tion err: 0.317, Validation loss: 0.6330568259581923
Epoch 14: Train err: 0.259125, Train loss: 0.5085404133796692 |Valid
ation err: 0.3095, Validation loss: 0.6042338404804468
Epoch 15: Train err: 0.255625, Train loss: 0.5034471051692962 |Valid
ation err: 0.3295, Validation loss: 0.6242936421185732
Epoch 16: Train err: 0.252125, Train loss: 0.503919352531433 |Valida
tion err: 0.3205, Validation loss: 0.6358922934159636
Epoch 17: Train err: 0.253, Train loss: 0.5068532364368439 |Validati
on err: 0.329, Validation loss: 0.641121020540595
Epoch 18: Train err: 0.2595, Train loss: 0.5080235059261322 |Validat
ion err: 0.3665, Validation loss: 0.7169719664379954
Epoch 19: Train err: 0.239, Train loss: 0.48701162552833555 |Validat
ion err: 0.357, Validation loss: 0.6543149184435606
Epoch 20: Train err: 0.24625, Train loss: 0.5008019053936005 |Valida
tion err: 0.323, Validation loss: 0.6212702291086316
Epoch 21: Train err: 0.231375, Train loss: 0.47623249506950377 |Vali
dation err: 0.332, Validation loss: 0.7205164767801762
Epoch 22: Train err: 0.244, Train loss: 0.4977173686027527 |Validati
on err: 0.3105, Validation loss: 0.6273324955254793
Epoch 23: Train err: 0.238375, Train loss: 0.4847790629863739 |Valid
ation err: 0.346, Validation loss: 0.6741146314889193
Epoch 24: Train err: 0.228375, Train loss: 0.4683375310897827 |Valid
ation err: 0.3095, Validation loss: 0.6837004749104381
Epoch 25: Train err: 0.226125, Train loss: 0.47662637662887575 |Vali
dation err: 0.3225, Validation loss: 0.6582820247858763
Epoch 26: Train err: 0.246125, Train loss: 0.5046286690235138 |Valid
ation err: 0.3245, Validation loss: 0.7345256488770247
Epoch 27: Train err: 0.249625, Train loss: 0.5075048182010651 |Valid
ation err: 0.3465, Validation loss: 0.7648427784442902
Epoch 28: Train err: 0.252, Train loss: 0.5126738095283508 |Validati
on err: 0.339, Validation loss: 0.7123841308057308
Epoch 29: Train err: 0.232875, Train loss: 0.4794297907352448 |Valid
ation err: 0.3345, Validation loss: 0.8030239716172218
Epoch 30: Train err: 0.22025, Train loss: 0.4747068645954132 |Valida
tion err: 0.329, Validation loss: 0.743980742059648
Finished Training
Total time elapsed: 127.77 seconds
```





Part (c) - 3pt

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

Answer:

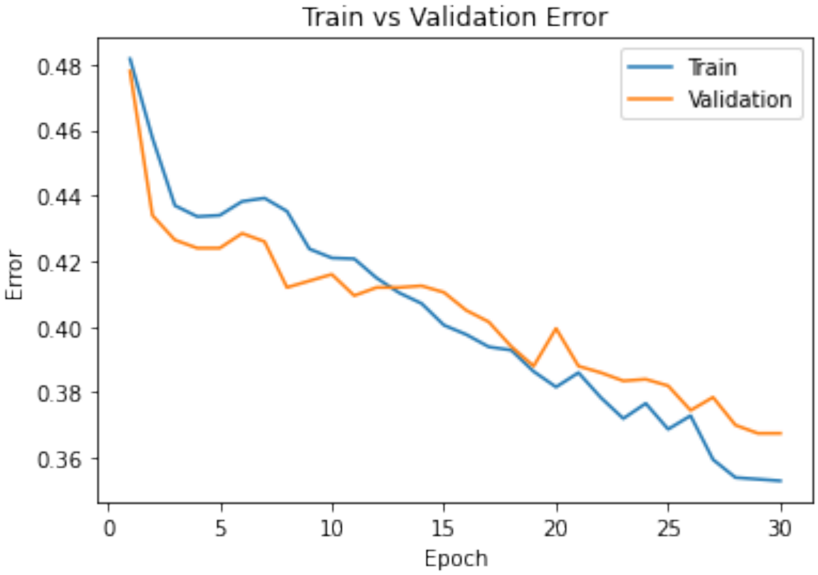
Total time elapsed is 112.38 seconds. The model takes shorter to train. From the plots, the validation error/loss shows decreasing trend,and the gap between train and validation curve becomes smaller in both error/loss polts. There is no overfitting over 20 epoch, which means it is better fitted than the default parameter model. Thus, increasing the batch size reduces the traning time and increases validation accuracy.

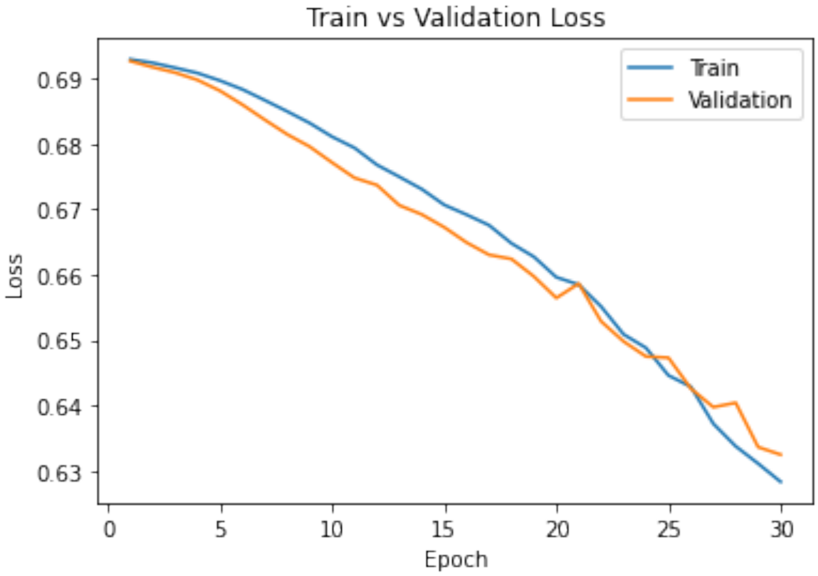
In [24]:

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

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.48175, Train loss: 0.6929379589855671 |Validat
ion err: 0.478, Validation loss: 0.6926824003458023
Epoch 2: Train err: 0.457625, Train loss: 0.6924104019999504 |Valida
tion err: 0.434, Validation loss: 0.6917425245046616
Epoch 3: Train err: 0.437, Train loss: 0.6916500627994537 |Validatio
n err: 0.4265, Validation loss: 0.6909130364656448
Epoch 4: Train err: 0.433625, Train loss: 0.6908450126647949 |Valida
tion err: 0.424, Validation loss: 0.6897871196269989
Epoch 5: Train err: 0.434, Train loss: 0.6896936185657978 |Validatio
n err: 0.424, Validation loss: 0.6881358623504639
Epoch 6: Train err: 0.43825, Train loss: 0.6883535124361515 |Validat
ion err: 0.4285, Validation loss: 0.6860134303569794
Epoch 7: Train err: 0.43925, Train loss: 0.6866881102323532 |Validat
ion err: 0.426, Validation loss: 0.6836976855993271
Epoch 8: Train err: 0.43525, Train loss: 0.6849788911640644 |Validat
ion err: 0.412, Validation loss: 0.681468278169632
Epoch 9: Train err: 0.42375, Train loss: 0.6832026764750481 |Validat
ion err: 0.414, Validation loss: 0.6795943975448608
Epoch 10: Train err: 0.421, Train loss: 0.6811111941933632 |Validati
on err: 0.416, Validation loss: 0.6771571338176727
Epoch 11: Train err: 0.42075, Train loss: 0.6794053874909878 |Valida
tion err: 0.4095, Validation loss: 0.6748155355453491
Epoch 12: Train err: 0.414875, Train loss: 0.6768094897270203 |Valid
ation err: 0.412, Validation loss: 0.6737167537212372
Epoch 13: Train err: 0.410375, Train loss: 0.6749758012592793 |Valid
ation err: 0.412, Validation loss: 0.6706155687570572
Epoch 14: Train err: 0.407125, Train loss: 0.6730947196483612 |Valid
ation err: 0.4125, Validation loss: 0.6692224740982056
Epoch 15: Train err: 0.4005, Train loss: 0.6706876419484615 |Validat
ion err: 0.4105, Validation loss: 0.6672652214765549
```

```
Epoch 16: Train err: 0.397625, Train loss: 0.6691837050020695 |Valid
ation err: 0.405, Validation loss: 0.6649233102798462
Epoch 17: Train err: 0.393875, Train loss: 0.6675742603838444 |Valid
ation err: 0.4015, Validation loss: 0.6630482077598572
Epoch 18: Train err: 0.392875, Train loss: 0.6648024022579193 |Valid
ation err: 0.394, Validation loss: 0.6624124348163605
Epoch 19: Train err: 0.386375, Train loss: 0.6627459488809109 |Valid
ation err: 0.388, Validation loss: 0.6597411781549454
Epoch 20: Train err: 0.381625, Train loss: 0.6596225798130035 |Valid
ation err: 0.3995, Validation loss: 0.6564429700374603
Epoch 21: Train err: 0.386, Train loss: 0.6584866605699062 |Validati
on err: 0.388, Validation loss: 0.6586524397134781
Epoch 22: Train err: 0.378375, Train loss: 0.6551279500126839 |Valid
ation err: 0.386, Validation loss: 0.6528601199388504
Epoch 23: Train err: 0.372, Train loss: 0.6508878879249096 |Validati
on err: 0.3835, Validation loss: 0.649801716208458
Epoch 24: Train err: 0.376625, Train loss: 0.6488120630383492 |Valid
ation err: 0.384, Validation loss: 0.6474965512752533
Epoch 25: Train err: 0.36875, Train loss: 0.6446062549948692 |Valida
tion err: 0.382, Validation loss: 0.6473138779401779
Epoch 26: Train err: 0.372875, Train loss: 0.6428665332496166 |Valid
ation err: 0.3745, Validation loss: 0.642597571015358
Epoch 27: Train err: 0.3595, Train loss: 0.6372309774160385 |Validat
ion err: 0.3785, Validation loss: 0.639750525355339
Epoch 28: Train err: 0.354, Train loss: 0.6337686441838741 |Validati
on err: 0.37, Validation loss: 0.6404179036617279
Epoch 29: Train err: 0.3535, Train loss: 0.6311231106519699 |Validat
ion err: 0.3675, Validation loss: 0.6336427330970764
Epoch 30: Train err: 0.353, Train loss: 0.6283389702439308 |Validati
on err: 0.3675, Validation loss: 0.6324894577264786
Finished Training
Total time elapsed: 112.38 seconds
```





Part (d) - 3pt

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

Answer:

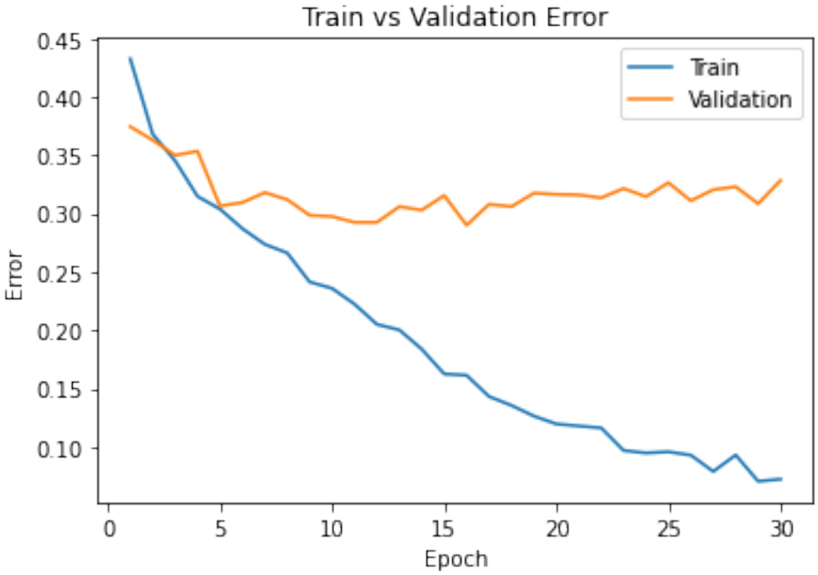
Total time elapsed is 180.60 seconds. The model takes longer to train. From the plots, the gap between train and validation curve becomes larger in both error/loss polts after 5 epoch, which is overfitting earlier than the default parameter model. Thus, decreasing the batch size increase the traning time and decreases validation accuracy.

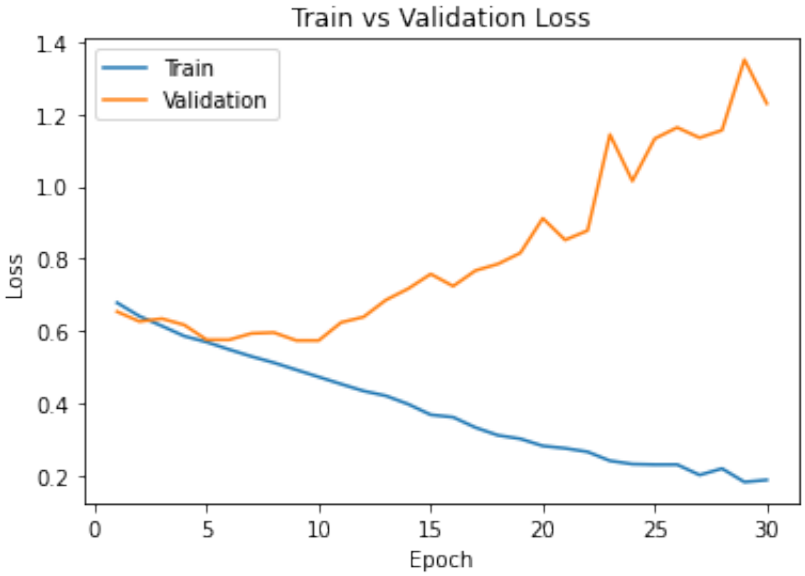
In [25]:
```
large_net = LargeNet()
train_net(large_net, batch_size=16)
large_path_d = get_model_name("large", batch_size = 16, learning_rate
= 0.01,
                                                  epoch = 29)
plot_training_curve(large_path_d)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.432625, Train loss: 0.6776098045110702 |Valida
tion err: 0.3745, Validation loss: 0.6529334635734558
Epoch 2: Train err: 0.368125, Train loss: 0.6402349994182587 |Valida
tion err: 0.363, Validation loss: 0.6257380561828614
Epoch 3: Train err: 0.34525, Train loss: 0.6132398887872695 |Validat
ion err: 0.35, Validation loss: 0.633517231464386
```

```
Epoch 4: Train err: 0.31475, Train loss: 0.5851500154137611 |Validat
ion err: 0.3535, Validation loss: 0.6160770399570465
Epoch 5: Train err: 0.303875, Train loss: 0.5685942940711975 |Valida
tion err: 0.3065, Validation loss: 0.5749539234638215
Epoch 6: Train err: 0.287125, Train loss: 0.5481366667151452 |Valida
tion err: 0.3095, Validation loss: 0.5749097964763641
Epoch 7: Train err: 0.273625, Train loss: 0.5287300577163696 |Valida
tion err: 0.318, Validation loss: 0.5926791634559632
Epoch 8: Train err: 0.26625, Train loss: 0.5114683332741261 |Validat
ion err: 0.312, Validation loss: 0.5949784636497497
Epoch 9: Train err: 0.241375, Train loss: 0.4916866025328636 |Valida
tion err: 0.2985, Validation loss: 0.5729981939792633
Epoch 10: Train err: 0.235875, Train loss: 0.4719826597571373 |Valid
ation err: 0.2975, Validation loss: 0.5729825073480606
Epoch 11: Train err: 0.222375, Train loss: 0.45227504616975783 |Vali
dation err: 0.2925, Validation loss: 0.6230180209875107
Epoch 12: Train err: 0.205, Train loss: 0.4331896264255047 |Validati
on err: 0.2925, Validation loss: 0.6381484514474869
Epoch 13: Train err: 0.20025, Train loss: 0.4197095323652029 |Valida
tion err: 0.306, Validation loss: 0.6852599532604218
Epoch 14: Train err: 0.18375, Train loss: 0.3964645936340094 |Valida
tion err: 0.303, Validation loss: 0.7167223781347275
Epoch 15: Train err: 0.1625, Train loss: 0.36721558406949045 |Valida
tion err: 0.3155, Validation loss: 0.7572290523052215
Epoch 16: Train err: 0.161375, Train loss: 0.36049251943826677 |Vali
dation err: 0.29, Validation loss: 0.7235817478895188
Epoch 17: Train err: 0.143125, Train loss: 0.3317276249304414 |Valid
ation err: 0.308, Validation loss: 0.7668795034885406
Epoch 18: Train err: 0.1355, Train loss: 0.3103339282795787 |Validat
ion err: 0.306, Validation loss: 0.7849839997291564
Epoch 19: Train err: 0.126375, Train loss: 0.30059052174538375 |Vali
dation err: 0.3175, Validation loss: 0.8156078016757965
Epoch 20: Train err: 0.119625, Train loss: 0.28093607498705386 |Vali
dation err: 0.3165, Validation loss: 0.9115923576354981
Epoch 21: Train err: 0.117875, Train loss: 0.2743322209268808 |Valid
ation err: 0.316, Validation loss: 0.8515847996473312
Epoch 22: Train err: 0.11625, Train loss: 0.2645689582154155 |Valida
tion err: 0.3135, Validation loss: 0.8775560821294784
Epoch 23: Train err: 0.096875, Train loss: 0.23964216740056873 |Vali
dation err: 0.3215, Validation loss: 1.1437096375226974
Epoch 24: Train err: 0.09475, Train loss: 0.23061982102319598 |Valid
ation err: 0.3145, Validation loss: 1.0149425619840622
Epoch 25: Train err: 0.09575, Train loss: 0.2290520599540323 |Valida
tion err: 0.3265, Validation loss: 1.132411583542824
Epoch 26: Train err: 0.092875, Train loss: 0.22914073724485934 |Vali
dation err: 0.311, Validation loss: 1.1632333843708038
Epoch 27: Train err: 0.078875, Train loss: 0.20027509773382918 |Vali
dation err: 0.3205, Validation loss: 1.1347101644277573
Epoch 28: Train err: 0.093, Train loss: 0.2181558216291014 |Validati
on err: 0.323, Validation loss: 1.1555380868911742
```

```
Epoch 29: Train err: 0.0705, Train loss: 0.1808979991725646 |Validat
ion err: 0.3085, Validation loss: 1.3511998779922725
Epoch 30: Train err: 0.07225, Train loss: 0.1865640614181757 |Valida
tion err: 0.3285, Validation loss: 1.2295081096887588
Finished Training
Total time elapsed: 180.60 seconds
```





Part 4. Hyperparameter Search [6 pt]

Part (a) - 2pt

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

Answer:

Based on the plots from above, lower learning rate and larger batch size can avoid overfitting and reduce traning time. In the previous part, batch_size=512 and learning rate=0.001 yield better result respectively, so I plan to combine these two and choose large network, batch_size=512, learning rate=0.001.

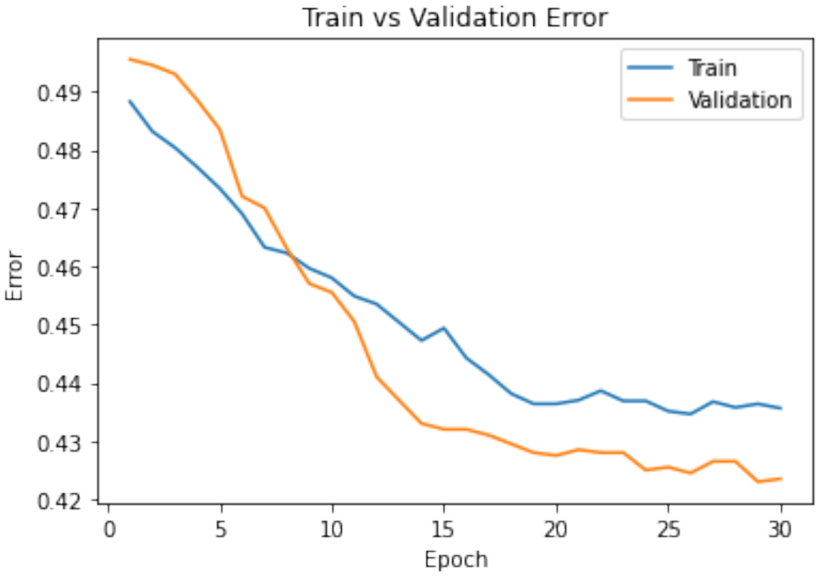
Part (b) - 1pt

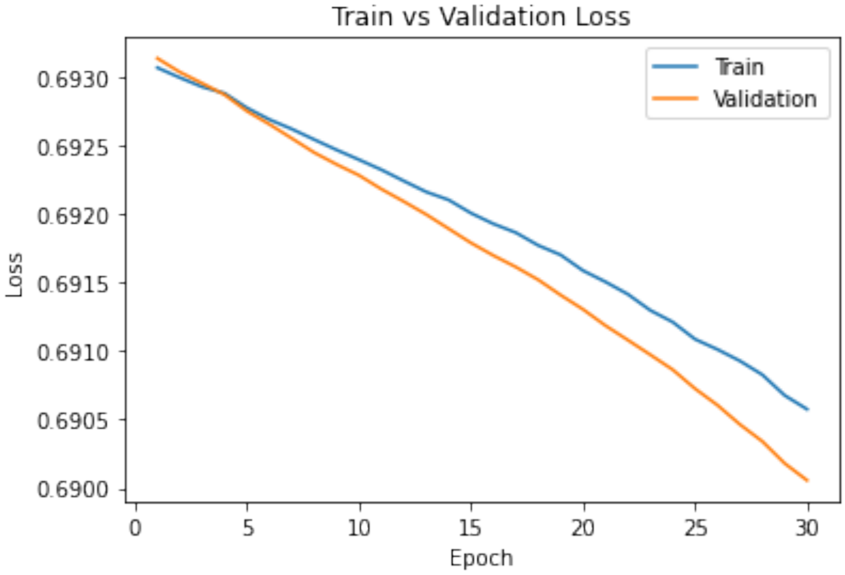
Train the model with the hyperparameters you chose in part(a), and include the training curve.

In [26]:
```
large_net = LargeNet()
train_net(large_net, batch_size=512, learning_rate=0.001, num_epochs=3
0)
large_path_4b = get_model_name("large", batch_size=512, learning_rate=
0.001,
                              epoch=29)
plot_training_curve(large_path_4b)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.48825, Train loss: 0.6930677443742752 |Validat
ion err: 0.4955, Validation loss: 0.6931362152099609
Epoch 2: Train err: 0.483125, Train loss: 0.6929955147206783 |Valida
tion err: 0.4945, Validation loss: 0.6930360496044159
Epoch 3: Train err: 0.480375, Train loss: 0.6929280422627926 |Valida
tion err: 0.493, Validation loss: 0.6929539740085602
Epoch 4: Train err: 0.477, Train loss: 0.6928808465600014 |Validatio
n err: 0.4885, Validation loss: 0.6928706914186478
Epoch 5: Train err: 0.473375, Train loss: 0.692774411290884 |Validat
ion err: 0.4835, Validation loss: 0.692750483751297
Epoch 6: Train err: 0.469, Train loss: 0.692689623683691 |Validation
err: 0.472, Validation loss: 0.6926551908254623
Epoch 7: Train err: 0.46325, Train loss: 0.692620363086462 |Validati
on err: 0.47, Validation loss: 0.6925524473190308
Epoch 8: Train err: 0.46225, Train loss: 0.6925435587763786 |Validat
```

```
ion err: 0.463, Validation loss: 0.6924485266208649
Epoch 9: Train err: 0.459625, Train loss: 0.6924680471420288 |Valida
tion err: 0.457, Validation loss: 0.6923621445894241
Epoch 10: Train err: 0.458, Train loss: 0.6923965588212013 |Validati
on err: 0.4555, Validation loss: 0.6922826170921326
Epoch 11: Train err: 0.454875, Train loss: 0.692323099821806 |Valida
tion err: 0.4505, Validation loss: 0.692181870341301
Epoch 12: Train err: 0.4535, Train loss: 0.6922412402927876 |Validat
ion err: 0.441, Validation loss: 0.6920914500951767
Epoch 13: Train err: 0.450375, Train loss: 0.6921614743769169 |Valid
ation err: 0.437, Validation loss: 0.6919969022274017
Epoch 14: Train err: 0.44725, Train loss: 0.6921032629907131 |Valida
tion err: 0.433, Validation loss: 0.6918932199478149
Epoch 15: Train err: 0.449375, Train loss: 0.6920064613223076 |Valid
ation err: 0.432, Validation loss: 0.6917892098426819
Epoch 16: Train err: 0.44425, Train loss: 0.6919283792376518 |Valida
tion err: 0.432, Validation loss: 0.6916972696781158
Epoch 17: Train err: 0.441375, Train loss: 0.6918644830584526 |Valid
ation err: 0.431, Validation loss: 0.6916135549545288
Epoch 18: Train err: 0.438125, Train loss: 0.6917712613940239 |Valid
ation err: 0.4295, Validation loss: 0.6915201842784882
Epoch 19: Train err: 0.436375, Train loss: 0.6917018331587315 |Valid
ation err: 0.428, Validation loss: 0.6914086788892746
Epoch 20: Train err: 0.436375, Train loss: 0.6915871202945709 |Valid
ation err: 0.4275, Validation loss: 0.6913044303655624
Epoch 21: Train err: 0.437, Train loss: 0.6915052533149719 |Validati
on err: 0.4285, Validation loss: 0.6911861002445221
Epoch 22: Train err: 0.438625, Train loss: 0.6914149858057499 |Valid
ation err: 0.428, Validation loss: 0.6910804063081741
Epoch 23: Train err: 0.436875, Train loss: 0.691297460347414 |Valida
tion err: 0.428, Validation loss: 0.6909734606742859
Epoch 24: Train err: 0.436875, Train loss: 0.6912120655179024 |Valid
ation err: 0.425, Validation loss: 0.6908645182847977
Epoch 25: Train err: 0.435125, Train loss: 0.6910865493118763 |Valid
ation err: 0.4255, Validation loss: 0.6907256990671158
Epoch 26: Train err: 0.434625, Train loss: 0.6910119391977787 |Valid
ation err: 0.4245, Validation loss: 0.6906052231788635
Epoch 27: Train err: 0.43675, Train loss: 0.6909283809363842 |Valida
tion err: 0.4265, Validation loss: 0.6904649585485458
Epoch 28: Train err: 0.43575, Train loss: 0.6908275820314884 |Valida
tion err: 0.4265, Validation loss: 0.6903414130210876
Epoch 29: Train err: 0.436375, Train loss: 0.6906765811145306 |Valid
ation err: 0.423, Validation loss: 0.6901804357767105
Epoch 30: Train err: 0.435625, Train loss: 0.690575547516346 |Valida
tion err: 0.4235, Validation loss: 0.6900565922260284
Finished Training
Total time elapsed: 113.47 seconds
```





Part (c) - 2pt

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

Based on result from Part(a), the taining time is slightly longer than part 3c. I choose small network, batch_size=512, learning rate=0.001. Since small network can reduce training time and has lower error/loss than large network.

Part (d) - 1pt

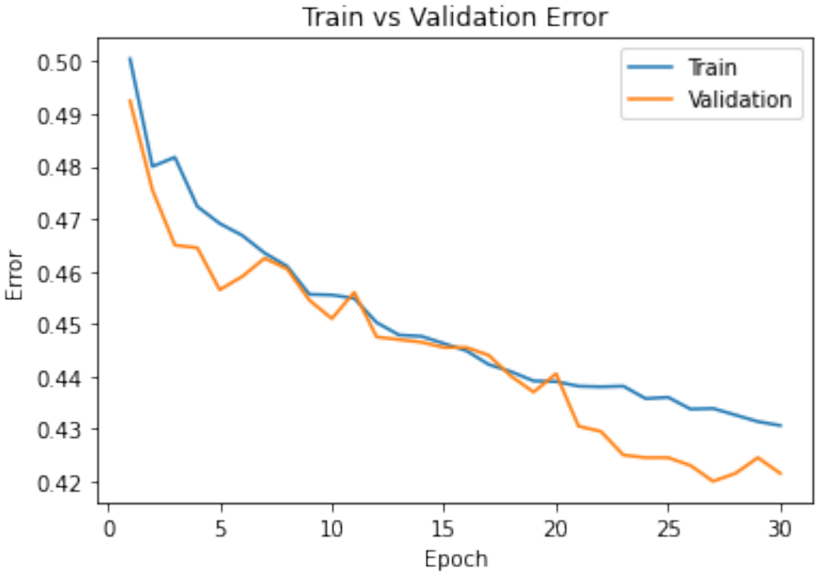
Train the model with the hyperparameters you chose in part(c), and include the training curve.

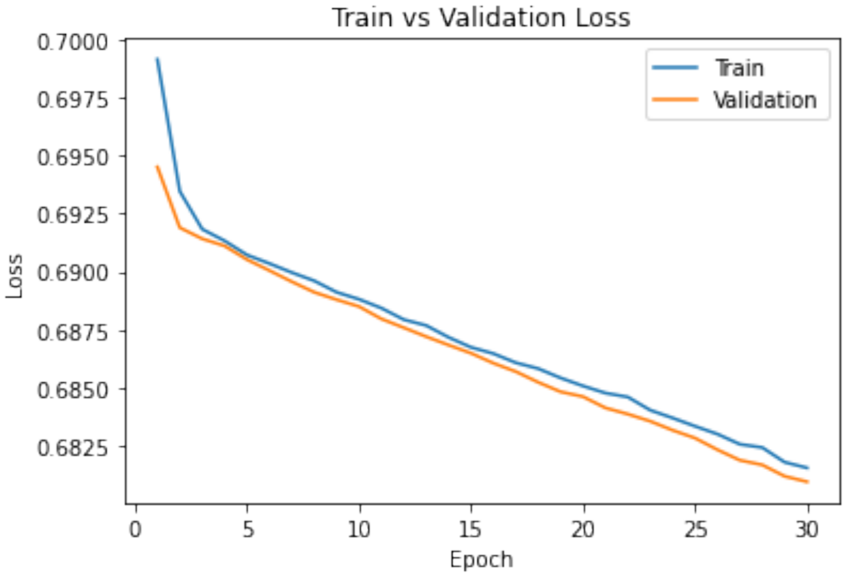
In [27]:

```
small_net = SmallNet()
train_net(small_net, batch_size=512, learning_rate=0.001, num_epochs=3
0)
small_path_4d = get_model_name("small", batch_size=512, learning_rate=
0.001,
                               epoch=29)
plot_training_curve(small_path_4d)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.5005, Train loss: 0.6991336531937122 |Validati
on err: 0.4925, Validation loss: 0.6945076286792755
Epoch 2: Train err: 0.48, Train loss: 0.6934637650847435 |Validation
err: 0.4755, Validation loss: 0.6918978095054626
Epoch 3: Train err: 0.48175, Train loss: 0.6918338537216187 |Validat
ion err: 0.465, Validation loss: 0.6914202272891998
Epoch 4: Train err: 0.472375, Train loss: 0.6913324147462845 |Valida
tion err: 0.4645, Validation loss: 0.6911140233278275
Epoch 5: Train err: 0.469125, Train loss: 0.6907239817082882 |Valida
tion err: 0.4565, Validation loss: 0.690525159239769
Epoch 6: Train err: 0.466875, Train loss: 0.690356221050024 |Validat
ion err: 0.459, Validation loss: 0.6900540739297867
Epoch 7: Train err: 0.4635, Train loss: 0.6899680867791176 |Validati
on err: 0.4625, Validation loss: 0.6895745396614075
Epoch 8: Train err: 0.461, Train loss: 0.6896125487983227 |Validatio
n err: 0.4605, Validation loss: 0.6891213655471802
Epoch 9: Train err: 0.455625, Train loss: 0.6891210973262787 |Valida
tion err: 0.4545, Validation loss: 0.6888016909360886
Epoch 10: Train err: 0.4555, Train loss: 0.6888190060853958 |Validat
ion err: 0.451, Validation loss: 0.6885065883398056
Epoch 11: Train err: 0.454875, Train loss: 0.6884388588368893 |Valid
ation err: 0.456, Validation loss: 0.6879743337631226
Epoch 12: Train err: 0.45025, Train loss: 0.6879428029060364 |Valida
tion err: 0.4475, Validation loss: 0.6875960826873779
Epoch 13: Train err: 0.447875, Train loss: 0.6876913011074066 |Valid
ation err: 0.447, Validation loss: 0.6872121244668961
Epoch 14: Train err: 0.447625, Train loss: 0.6871819645166397 |Valid
ation err: 0.4465, Validation loss: 0.6868490129709244
Epoch 15: Train err: 0.44625, Train loss: 0.6867522932589054 |Valida
tion err: 0.4455, Validation loss: 0.6864986419677734
Epoch 16: Train err: 0.444875, Train loss: 0.686481948941946 |Valida
tion err: 0.4455, Validation loss: 0.6860699504613876
Epoch 17: Train err: 0.44225, Train loss: 0.686088815331459 |Validat
```

```
ion err: 0.444, Validation loss: 0.685708150267601
Epoch 18: Train err: 0.440875, Train loss: 0.6858362071216106 |Valid
ation err: 0.44, Validation loss: 0.6852489411830902
Epoch 19: Train err: 0.439125, Train loss: 0.6854348182678223 |Valid
ation err: 0.437, Validation loss: 0.6848396062850952
Epoch 20: Train err: 0.439, Train loss: 0.6850905865430832 |Validati
on err: 0.4405, Validation loss: 0.6846307516098022
Epoch 21: Train err: 0.438125, Train loss: 0.684784647077322 |Valida
tion err: 0.4305, Validation loss: 0.6841461807489395
Epoch 22: Train err: 0.438, Train loss: 0.684614758938551 |Validatio
n err: 0.4295, Validation loss: 0.6838800311088562
Epoch 23: Train err: 0.438125, Train loss: 0.6840485744178295 |Valid
ation err: 0.425, Validation loss: 0.6835738867521286
Epoch 24: Train err: 0.43575, Train loss: 0.6837127543985844 |Valida
tion err: 0.4245, Validation loss: 0.6831868439912796
Epoch 25: Train err: 0.436, Train loss: 0.6833593137562275 |Validati
on err: 0.4245, Validation loss: 0.6828488856554031
Epoch 26: Train err: 0.43375, Train loss: 0.6830204203724861 |Valida
tion err: 0.423, Validation loss: 0.6823472529649734
Epoch 27: Train err: 0.433875, Train loss: 0.6825836412608624 |Valid
ation err: 0.42, Validation loss: 0.6818965673446655
Epoch 28: Train err: 0.432625, Train loss: 0.682443380355835 |Valida
tion err: 0.4215, Validation loss: 0.6816964596509933
Epoch 29: Train err: 0.431375, Train loss: 0.6818121634423733 |Valid
ation err: 0.4245, Validation loss: 0.6812071800231934
Epoch 30: Train err: 0.430625, Train loss: 0.681567408144474 |Valida
tion err: 0.4215, Validation loss: 0.680970624089241
Finished Training
Total time elapsed: 102.51 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 [28]:
```
net = SmallNet()
model_path = get_model_name(net.name, batch_size=512, learning_rate=0.
001,
                                    epoch=29)
state = torch.load(model_path)
net.load_state_dict(state)
```

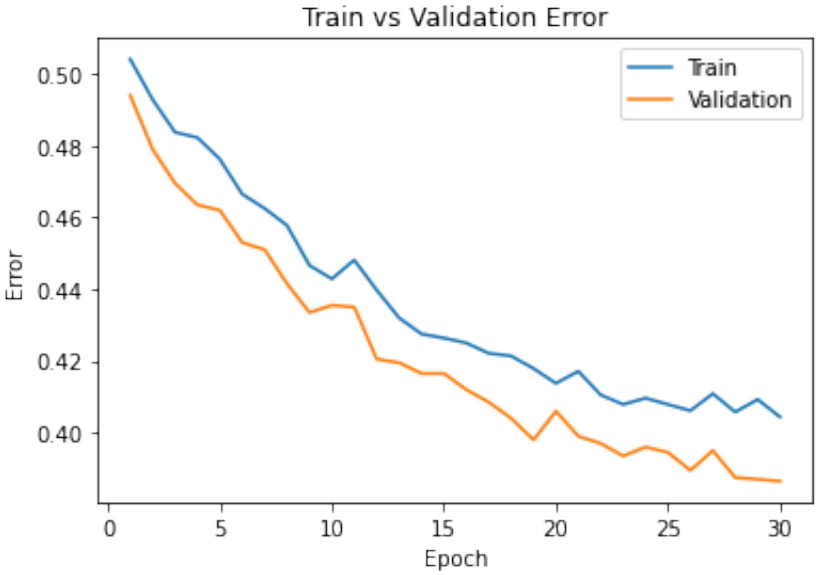
Out[28]:
```
<All keys matched successfully>
```

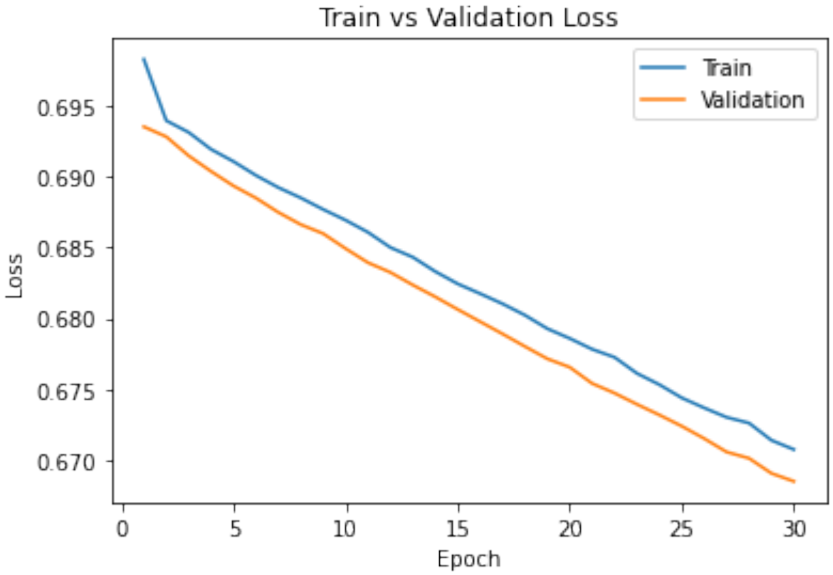
In [29]:
```
net = SmallNet()
train_net(net, batch_size=512, learning_rate=0.001, num_epochs=30)
model_path = get_model_name(net.name, batch_size=512, learning_rate=0.
001, epoch=29)
plot_training_curve(model_path)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.504125, Train loss: 0.6982664950191975 |Valida
```

```
tion err: 0.494, Validation loss: 0.6935129761695862
Epoch 2: Train err: 0.492875, Train loss: 0.6939455606043339 |Valida
tion err: 0.479, Validation loss: 0.6928094327449799
Epoch 3: Train err: 0.48375, Train loss: 0.693117331713438 |Validati
on err: 0.4695, Validation loss: 0.6914655119180679
Epoch 4: Train err: 0.48225, Train loss: 0.6919195018708706 |Validat
ion err: 0.4635, Validation loss: 0.6903645098209381
Epoch 5: Train err: 0.47625, Train loss: 0.6910620555281639 |Validat
ion err: 0.462, Validation loss: 0.6893411129713058
Epoch 6: Train err: 0.4665, Train loss: 0.6900828815996647 |Validati
on err: 0.453, Validation loss: 0.6884862929582596
Epoch 7: Train err: 0.4625, Train loss: 0.6892257295548916 |Validati
on err: 0.451, Validation loss: 0.6874754875898361
Epoch 8: Train err: 0.45775, Train loss: 0.6884971223771572 |Validat
ion err: 0.4415, Validation loss: 0.6866151392459869
Epoch 9: Train err: 0.446625, Train loss: 0.6876824200153351 |Valida
tion err: 0.4335, Validation loss: 0.6859765499830246
Epoch 10: Train err: 0.442875, Train loss: 0.6869241185486317 |Valid
ation err: 0.4355, Validation loss: 0.684920534491539
Epoch 11: Train err: 0.448125, Train loss: 0.686068844050169 |Valida
tion err: 0.435, Validation loss: 0.6839260309934616
Epoch 12: Train err: 0.43975, Train loss: 0.6849946267902851 |Valida
tion err: 0.4205, Validation loss: 0.6832417100667953
Epoch 13: Train err: 0.432, Train loss: 0.6843136325478554 |Validati
on err: 0.4195, Validation loss: 0.6823530197143555
Epoch 14: Train err: 0.4275, Train loss: 0.6833049990236759 |Validat
ion err: 0.4165, Validation loss: 0.681518018245697
Epoch 15: Train err: 0.426375, Train loss: 0.6824350617825985 |Valid
ation err: 0.4165, Validation loss: 0.6806311011314392
Epoch 16: Train err: 0.425, Train loss: 0.6817414425313473 |Validati
on err: 0.412, Validation loss: 0.6797674596309662
Epoch 17: Train err: 0.422125, Train loss: 0.6810277812182903 |Valid
ation err: 0.4085, Validation loss: 0.678903728723526
Epoch 18: Train err: 0.421375, Train loss: 0.6802247911691666 |Valid
ation err: 0.404, Validation loss: 0.6780171692371368
Epoch 19: Train err: 0.417875, Train loss: 0.6792717687785625 |Valid
ation err: 0.398, Validation loss: 0.6771471798419952
Epoch 20: Train err: 0.41375, Train loss: 0.6785835586488247 |Valida
tion err: 0.406, Validation loss: 0.6765521913766861
Epoch 21: Train err: 0.417125, Train loss: 0.6778210513293743 |Valid
ation err: 0.399, Validation loss: 0.6754071712493896
Epoch 22: Train err: 0.4105, Train loss: 0.6772520877420902 |Validat
ion err: 0.397, Validation loss: 0.6747248619794846
Epoch 23: Train err: 0.407875, Train loss: 0.6761172078549862 |Valid
ation err: 0.3935, Validation loss: 0.6739353388547897
Epoch 24: Train err: 0.409625, Train loss: 0.6753240078687668 |Valid
ation err: 0.396, Validation loss: 0.673184722661972
Epoch 25: Train err: 0.407875, Train loss: 0.674388725310564 |Valida
tion err: 0.3945, Validation loss: 0.6723930090665817
Epoch 26: Train err: 0.406125, Train loss: 0.6736827455461025 |Valid
```

```
ation err: 0.3895, Validation loss: 0.6715212613344193
Epoch 27: Train err: 0.410875, Train loss: 0.673019465059042 |Valida
tion err: 0.395, Validation loss: 0.6705719083547592
Epoch 28: Train err: 0.40575, Train loss: 0.6726120188832283 |Valida
tion err: 0.3875, Validation loss: 0.6701292246580124
Epoch 29: Train err: 0.40925, Train loss: 0.6713981702923775 |Valida
tion err: 0.387, Validation loss: 0.6690665632486343
Epoch 30: Train err: 0.404375, Train loss: 0.6707480624318123 |Valid
ation err: 0.3865, Validation loss: 0.6685024201869965
Finished Training
Total time elapsed: 102.19 seconds
```





Part (b) - 2pt

Justify your choice of model from part (a).

Answer: The model from part (a) has lower training time, no outfitting, and the gap between train and validation curves are vary small. This means the validation accuracy is high and is better than the model with default hyperparameters, which the best fit model so far.

In [ ]:

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 [31]:

```
# 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_err,test_loss = evaluate(net, test_loader, criterion)
print("Test error is ", test_err)
print("Test loss is ", test_loss)
```

```
Files already downloaded and verified
Files already downloaded and verified
Test error is  0.3955
Test loss is  0.6704632341861725
```

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.

Answer:

The test classification error is higher than validation error. Because when we tune the hyperparameters, we continuesly use the same validation set to compute error/loss, which makes the model familiar with the data and fit the data. The test data set is used only at the very end. In this case, when test set enters the model, it might be different from validation data, so the test error would be higher than validation error.

In [32]:

```
train_loader, val_loader, test_loader, classes = get_data_loader(
    target_classes=["cat", "dog"],
    batch_size=512)

criterion = nn.BCEWithLogitsLoss()
vali_err,vali_loss = evaluate(net, val_loader, criterion)
print("Validation error is ", vali_err)
print("Validation loss is ", vali_loss)
```

```
Files already downloaded and verified
Files already downloaded and verified
Validation error is  0.3865
Validation loss is  0.668431892991066
```

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?

Answer:

Because test data used to final check the performance of a full trained model and verify the result. If we used the test data many times, the model would fit the test set and causes overfitting. So we use the test data as little as possible can improve test accuracy and avoid overfitting.

In [ ]:

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.

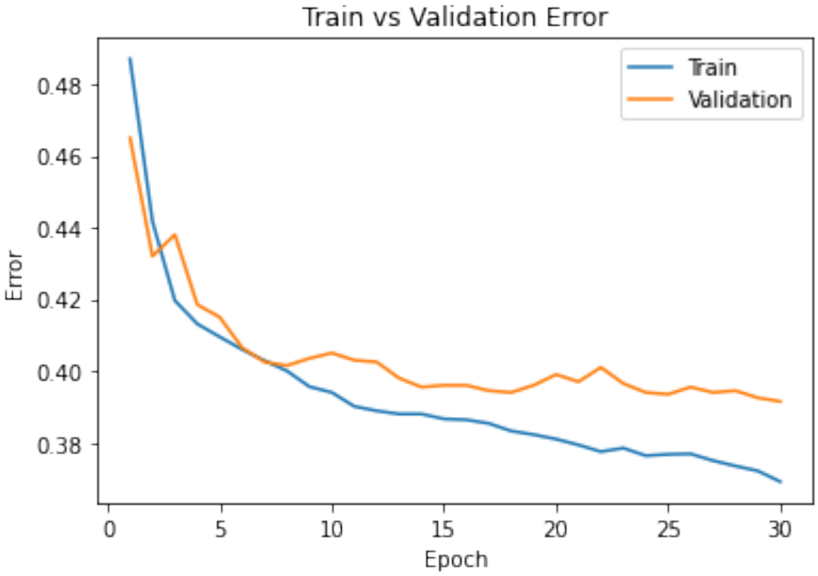
In [36]:

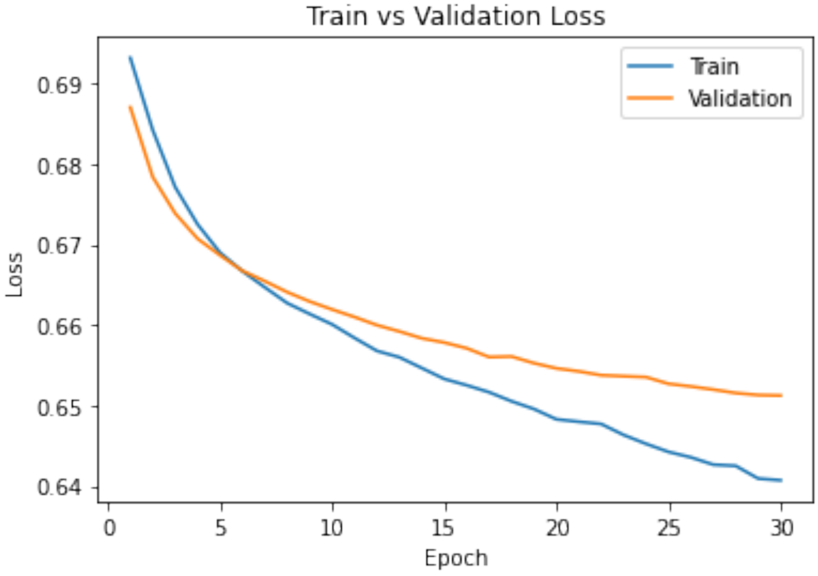
```
import torch
import torch.nn as nn
import torch.nn.functional as F
from torchvision import datasets, transforms
import matplotlib.pyplot as plt # for plotting
import torch.optim as optim

torch.manual_seed(1)# set the random seed

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

pigeon = Pigeon() #sets the NN as train_ANN
#trains and sets a rate of 0.05; determined to be best by my previous experiment
train_net(pigeon, batch_size = 512, learning_rate=0.001, num_epochs = 30)
#run the plotting function on the ANN
path = get_model_name("pigeon", 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: 0.487, Train loss: 0.6932093426585197 |Validatio
n err: 0.465, Validation loss: 0.6870602518320084
Epoch 2: Train err: 0.44175, Train loss: 0.6842432580888271 |Validat
ion err: 0.432, Validation loss: 0.6784038990736008
Epoch 3: Train err: 0.419625, Train loss: 0.6771588250994682 |Valida
tion err: 0.438, Validation loss: 0.6738857179880142
Epoch 4: Train err: 0.413125, Train loss: 0.6725212521851063 |Valida
tion err: 0.4185, Validation loss: 0.6707304120063782
Epoch 5: Train err: 0.4095, Train loss: 0.6690065860748291 |Validati
on err: 0.415, Validation loss: 0.6686770021915436
Epoch 6: Train err: 0.406, Train loss: 0.6666937805712223 |Validatio
n err: 0.4065, Validation loss: 0.6667376905679703
Epoch 7: Train err: 0.402875, Train loss: 0.6646820940077305 |Valida
tion err: 0.4025, Validation loss: 0.6654486507177353
Epoch 8: Train err: 0.4, Train loss: 0.6627189256250858 |Validation
err: 0.4015, Validation loss: 0.6640831679105759
Epoch 9: Train err: 0.395625, Train loss: 0.6613717153668404 |Valida
tion err: 0.4035, Validation loss: 0.6629217267036438
Epoch 10: Train err: 0.394, Train loss: 0.6600756794214249 |Validati
on err: 0.405, Validation loss: 0.6619308888912201
Epoch 11: Train err: 0.390125, Train loss: 0.6583879552781582 |Valid
ation err: 0.403, Validation loss: 0.6609926372766495
Epoch 12: Train err: 0.388875, Train loss: 0.6567776501178741 |Valid
ation err: 0.4025, Validation loss: 0.6599807441234589
Epoch 13: Train err: 0.388, Train loss: 0.6559904254972935 |Validati
on err: 0.398, Validation loss: 0.6591979712247849
Epoch 14: Train err: 0.388, Train loss: 0.6546466015279293 |Validati
on err: 0.3955, Validation loss: 0.6583474427461624
Epoch 15: Train err: 0.386625, Train loss: 0.6532963253557682 |Valid
ation err: 0.396, Validation loss: 0.6578210592269897
Epoch 16: Train err: 0.386375, Train loss: 0.6525077000260353 |Valid
ation err: 0.396, Validation loss: 0.6571104675531387
Epoch 17: Train err: 0.385375, Train loss: 0.6516455113887787 |Valid
ation err: 0.3945, Validation loss: 0.6560133695602417
Epoch 18: Train err: 0.38325, Train loss: 0.6505177244544029 |Valida
tion err: 0.394, Validation loss: 0.6560816168785095
Epoch 19: Train err: 0.38225, Train loss: 0.6495578847825527 |Valida
tion err: 0.396, Validation loss: 0.6552560925483704
Epoch 20: Train err: 0.381, Train loss: 0.648275762796402 |Validatio
n err: 0.399, Validation loss: 0.654608964920044
Epoch 21: Train err: 0.379375, Train loss: 0.6479700952768326 |Valid
ation err: 0.397, Validation loss: 0.6542342752218246
Epoch 22: Train err: 0.3775, Train loss: 0.6476960107684135 |Validat
ion err: 0.401, Validation loss: 0.6537543833255768
Epoch 23: Train err: 0.3785, Train loss: 0.6463310793042183 |Validat
ion err: 0.3965, Validation loss: 0.6536329090595245
Epoch 24: Train err: 0.376375, Train loss: 0.6452262178063393 |Valid
ation err: 0.394, Validation loss: 0.6535203903913498
Epoch 25: Train err: 0.37675, Train loss: 0.6442242860794067 |Valida
tion err: 0.3935, Validation loss: 0.6526893973350525
```

```
Epoch 26: Train err: 0.376875, Train loss: 0.6435430124402046 |Valid
ation err: 0.3955, Validation loss: 0.6523561924695969
Epoch 27: Train err: 0.375, Train loss: 0.6426348350942135 |Validati
on err: 0.394, Validation loss: 0.6519708931446075
Epoch 28: Train err: 0.3735, Train loss: 0.6424877606332302 |Validat
ion err: 0.3945, Validation loss: 0.6515385806560516
Epoch 29: Train err: 0.372125, Train loss: 0.6409070640802383 |Valid
ation err: 0.3925, Validation loss: 0.6512900143861771
Epoch 30: Train err: 0.369125, Train loss: 0.6406939662992954 |Valid
ation err: 0.3915, Validation loss: 0.6512300670146942
Finished Training
Total time elapsed: 79.16 seconds
```





In [38]:

```
train_loader, val_loader, test_loader, classes = get_data_loader(
    target_classes=["cat", "dog"],
    batch_size = 512)

criterion = nn.BCEWithLogitsLoss()
test_err, test_loss = evaluate(pigeon, test_loader, criterion)
print("test classification error: ", test_err)
print("test classification loss:",test_loss)
```

```
Files already downloaded and verified
Files already downloaded and verified
test classification error:  0.3685
test classification loss: 0.6480787694454193
```