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://colab.research.google.com/drive/15TmB7LkIzWLPWP2is-iuqhFeybsKZOLZ?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[split:]
    train_sampler = SubsetRandomSampler(relevant_train_indices)
    train_loader = torch.utils.data.DataLoader(trainset, batch_size=batch_size,
                                               num_workers=1, sampler=train_sampler)
    val_sampler = SubsetRandomSampler(relevant_val_indices)
    val_loader = torch.utils.data.DataLoader(trainset, batch_size=batch_size,
                                              num_workers=1, sampler=val_sampler)
    # Load CIFAR10 testing data
    testset = torchvision.datasets.CIFAR10(root='./data', train=False,
                                           download=True, transform=transform)
    # Get the list of indices to sample from
    relevant_test_indices = get_relevant_indices(testset, classes, target_classes)
    test_sampler = SubsetRandomSampler(relevant_test_indices)
    test_loader = torch.utils.data.DataLoader(testset, batch_size=batch_size,
                                             num_workers=1, sampler=test_sampler)
    return train_loader, val_loader, test_loader, classes

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

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

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

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

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

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

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

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

Part 1. Visualizing the Data [7 pt]

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

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

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
```

```
100%|██████████| 170M/170M [00:01<00:00, 105MB/s]
```

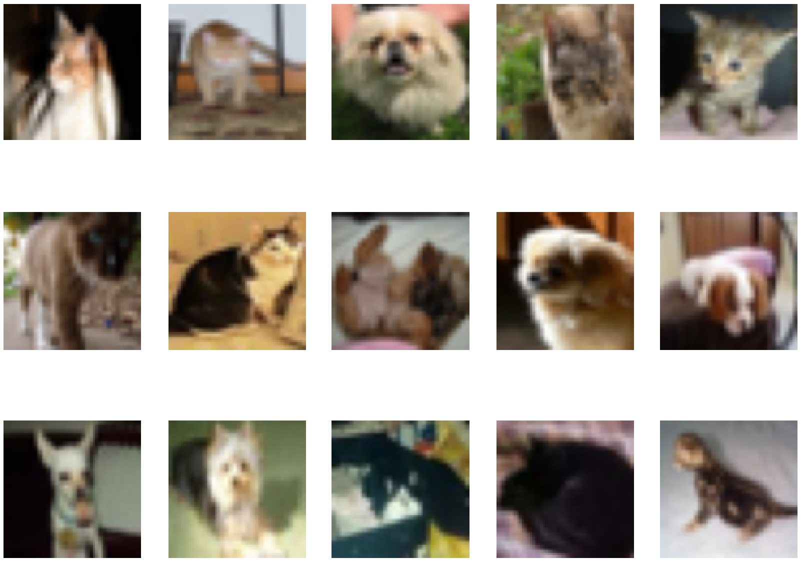
Part (a) -- 1 pt

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]:

```
print("Number of training examples: {}".format(len(train_loader)))
print("Number of validation examples: {}".format(len(val_loader)))
print("Number of test examples: {}".format(len(test_loader)))
```

```
Number of training examples: 8000
Number of validation examples: 2000
Number of test 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:

We need a validation set to get an unbiased evaluation of the model during training and to detect overfitting/underfitting. Using the training set error alone can be misleading as the model will naturally perform well on data it has already seen, potentially leading to overfitting where the model doesn't generalize well to new data. The validation set acts as a proxy for unseen data.

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]:

```
print("small_net:")
for param in small_net.parameters():
    print(param.shape)
print()

print("large_net")
for param in large_net.parameters():
    print(param.shape)
```

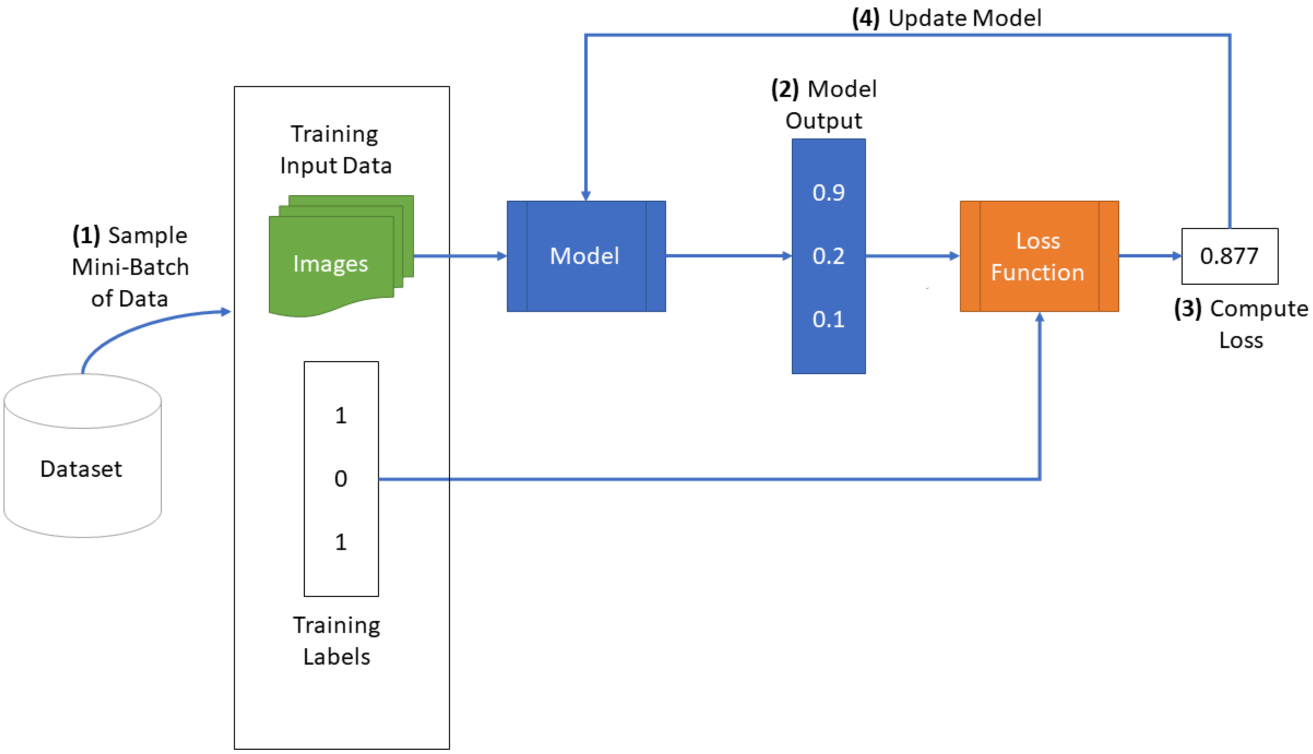
```
small_net:
torch.Size([5, 3, 3, 3])
torch.Size([5])
torch.Size([1, 245])
torch.Size([1])

large_net
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])
```

Answer:

- (5*3*3*3)+(5)+(1*245)+(1)=386
- (5*3*5*5)+(5)+(10*5*5*5)+(10)+(32*250)+(32)+(1*32)+(1)=9705
- There are **386** parameters in the small net, and **9705** parameters in the large net.

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`?

In [11]:

```
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 [12]:
```
train_net(small_net,num_epochs=5)
```

```
Epoch 1: Train err: 0.432375, Train loss: 0.6764058194160462 |Validation err: 0.3785, Validation loss: 0.6579908803
105354
Epoch 2: Train err: 0.370625, Train loss: 0.6472312378883361 |Validation err: 0.384, Validation loss: 0.65966347977
51904
Epoch 3: Train err: 0.355625, Train loss: 0.6366147956848145 |Validation err: 0.353, Validation loss: 0.62876553088
42659
Epoch 4: Train err: 0.341875, Train loss: 0.6221265025138855 |Validation err: 0.3495, Validation loss: 0.6207225974
649191
Epoch 5: Train err: 0.332125, Train loss: 0.6116174993515014 |Validation err: 0.333, Validation loss: 0.61577494069
93389
Finished Training
Total time elapsed: 23.77 seconds
```

Answer:

- checkpoints:
    - model_small_bs64_lr0.01_epoch0: checkpoint at epoch 0
    - model_small_bs64_lr0.01_epoch1: checkpoint at epoch 1
    - model_small_bs64_lr0.01_epoch2: checkpoint at epoch 2
    - model_small_bs64_lr0.01_epoch3: checkpoint at epoch 3
    - model_small_bs64_lr0.01_epoch4: checkpoint at epoch 4
- Train error/loss:
    - model_small_bs64_lr0.01_epoch4_train_err.csv: training error
    - model_small_bs64_lr0.01_epoch4_train_loss.csv: training loss
    - model_small_bs64_lr0.01_epoch4_val_err.csv: validation error
    - model_small_bs64_lr0.01_epoch4_val_loss.csv: validation loss

Part (d) -- 2pt

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

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

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

In [14]:
```
# 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 [15]:
```
print("Training small_net")
small_net = SmallNet()
train_net(small_net)

print("Training large_net")
large_net = LargeNet()
train_net(large_net)
```

```
Training small_net
Epoch 1: Train err: 0.446375, Train loss: 0.6813716740608216 |Validation err: 0.3865, Validation loss: 0.6602997388
69071
Epoch 2: Train err: 0.37325, Train loss: 0.6497629313468933 |Validation err: 0.3845, Validation loss: 0.65759957954
28753
Epoch 3: Train err: 0.360125, Train loss: 0.6389007897377014 |Validation err: 0.3495, Validation loss: 0.6291371881
961823
Epoch 4: Train err: 0.346375, Train loss: 0.6246497564315796 |Validation err: 0.356, Validation loss: 0.62216357700
52671
Epoch 5: Train err: 0.334125, Train loss: 0.6154210748672485 |Validation err: 0.328, Validation loss: 0.61893321759
9988
Epoch 6: Train err: 0.31775, Train loss: 0.6036666023731232 |Validation err: 0.3385, Validation loss: 0.60932970326
39384
Epoch 7: Train err: 0.315875, Train loss: 0.5944248065948486 |Validation err: 0.328, Validation loss: 0.59730244427
91939
Epoch 8: Train err: 0.308125, Train loss: 0.5829446413516999 |Validation err: 0.3085, Validation loss: 0.5884840954
095125
Epoch 9: Train err: 0.302875, Train loss: 0.5804586551189422 |Validation err: 0.312, Validation loss: 0.58448922634
12476
Epoch 10: Train err: 0.29825, Train loss: 0.5729375307559967 |Validation err: 0.3095, Validation loss: 0.5786221697
926521
Epoch 11: Train err: 0.28825, Train loss: 0.5630653114318848 |Validation err: 0.313, Validation loss: 0.58170716278
25499
Epoch 12: Train err: 0.292875, Train loss: 0.5564581718444824 |Validation err: 0.312, Validation loss: 0.5859681870
788336
Epoch 13: Train err: 0.288625, Train loss: 0.5559324417114258 |Validation err: 0.3045, Validation loss: 0.576554805
0403595
Epoch 14: Train err: 0.28, Train loss: 0.5471521301269531 |Validation err: 0.3115, Validation loss: 0.5721538551151
752
Epoch 15: Train err: 0.285125, Train loss: 0.5481183273792267 |Validation err: 0.3045, Validation loss: 0.562364244
8335886
Epoch 16: Train err: 0.29175, Train loss: 0.5540513117313385 |Validation err: 0.31, Validation loss: 0.577261805534
3628
Epoch 17: Train err: 0.28225, Train loss: 0.5475915327072144 |Validation err: 0.3, Validation loss: 0.5682430267333
984
Epoch 18: Train err: 0.2805, Train loss: 0.5438903319835663 |Validation err: 0.3185, Validation loss: 0.57733232248
5745
Epoch 19: Train err: 0.275625, Train loss: 0.5399831240177154 |Validation err: 0.33, Validation loss: 0.60773760639
13107
Epoch 20: Train err: 0.272875, Train loss: 0.5384533171653747 |Validation err: 0.296, Validation loss: 0.5776732238
009572
Epoch 21: Train err: 0.27675, Train loss: 0.5400643782615662 |Validation err: 0.302, Validation loss: 0.56759543437
51073
Epoch 22: Train err: 0.277375, Train loss: 0.5396291089057922 |Validation err: 0.288, Validation loss: 0.5697800805
792212
Epoch 23: Train err: 0.2735, Train loss: 0.5354287037849427 |Validation err: 0.302, Validation loss: 0.567152694799
006
Epoch 24: Train err: 0.272625, Train loss: 0.5356638813018799 |Validation err: 0.2995, Validation loss: 0.587589157
7452421
Epoch 25: Train err: 0.272875, Train loss: 0.534599012374878 |Validation err: 0.2965, Validation loss: 0.5639066351
57764
Epoch 26: Train err: 0.269875, Train loss: 0.5316087663173675 |Validation err: 0.2965, Validation loss: 0.569992721
08078
Epoch 27: Train err: 0.27, Train loss: 0.5297632284164429 |Validation err: 0.301, Validation loss: 0.57880384474992
75
Epoch 28: Train err: 0.270625, Train loss: 0.535400269985199 |Validation err: 0.2995, Validation loss: 0.5656933179
12519
Epoch 29: Train err: 0.272375, Train loss: 0.531760558128357 |Validation err: 0.295, Validation loss: 0.58491497207
43299
Epoch 30: Train err: 0.271, Train loss: 0.537471985578537 |Validation err: 0.315, Validation loss: 0.58150072023272
51
Finished Training
Total time elapsed: 141.55 seconds
Training large_net
Epoch 1: Train err: 0.444625, Train loss: 0.6900211324691773 |Validation err: 0.4305, Validation loss: 0.6807530857
622623
Epoch 2: Train err: 0.418875, Train loss: 0.6781997265815735 |Validation err: 0.4125, Validation loss: 0.6741086076
945066
Epoch 3: Train err: 0.39825, Train loss: 0.6658710074424744 |Validation err: 0.394, Validation loss: 0.651888888329
2675
Epoch 4: Train err: 0.3745, Train loss: 0.6488627505302429 |Validation err: 0.408, Validation loss: 0.6639942992478
609
Epoch 5: Train err: 0.35475, Train loss: 0.633074896812439 |Validation err: 0.352, Validation loss: 0.6291442140936
852
Epoch 6: Train err: 0.340625, Train loss: 0.6164079172611237 |Validation err: 0.3425, Validation loss: 0.6155355125
665665
Epoch 7: Train err: 0.3265, Train loss: 0.6007295575141907 |Validation err: 0.3395, Validation loss: 0.609989359974
```

```
8611
Epoch 8: Train err: 0.313375, Train loss: 0.5828288238048553 |Validation err: 0.327, Validation loss: 0.59817310236
3944
Epoch 9: Train err: 0.3095, Train loss: 0.5759536104202271 |Validation err: 0.3205, Validation loss: 0.595438154414
2962
Epoch 10: Train err: 0.29475, Train loss: 0.5617487483024597 |Validation err: 0.3145, Validation loss: 0.5929642692
20829
Epoch 11: Train err: 0.281, Train loss: 0.5484012789726257 |Validation err: 0.3155, Validation loss: 0.608386868610
9781
Epoch 12: Train err: 0.275875, Train loss: 0.5395474998950959 |Validation err: 0.3175, Validation loss: 0.597383063
2865429
Epoch 13: Train err: 0.271875, Train loss: 0.5274702501296997 |Validation err: 0.298, Validation loss: 0.5836638603
359461
Epoch 14: Train err: 0.260875, Train loss: 0.5171499395370484 |Validation err: 0.297, Validation loss: 0.5946236466
988921
Epoch 15: Train err: 0.25775, Train loss: 0.5102220706939697 |Validation err: 0.2995, Validation loss: 0.5893601188
43615
Epoch 16: Train err: 0.24725, Train loss: 0.5006417303085328 |Validation err: 0.299, Validation loss: 0.59780285693
70508
Epoch 17: Train err: 0.245375, Train loss: 0.4954251596927643 |Validation err: 0.3045, Validation loss: 0.586751978
8444042
Epoch 18: Train err: 0.2335, Train loss: 0.48036268329620363 |Validation err: 0.304, Validation loss: 0.61384835559
87477
Epoch 19: Train err: 0.233, Train loss: 0.47708055830001833 |Validation err: 0.333, Validation loss: 0.624375478364
5272
Epoch 20: Train err: 0.22875, Train loss: 0.46485390305519103 |Validation err: 0.3115, Validation loss: 0.603004706
0921788
Epoch 21: Train err: 0.22, Train loss: 0.45495913982391356 |Validation err: 0.2905, Validation loss: 0.602344034239
6498
Epoch 22: Train err: 0.217, Train loss: 0.4491056790351868 |Validation err: 0.2965, Validation loss: 0.618036177940
6667
Epoch 23: Train err: 0.217125, Train loss: 0.44135768175125123 |Validation err: 0.301, Validation loss: 0.604913904
3316245
Epoch 24: Train err: 0.203125, Train loss: 0.4262930202484131 |Validation err: 0.3165, Validation loss: 0.667187351
7334461
Epoch 25: Train err: 0.198375, Train loss: 0.41211577677726746 |Validation err: 0.312, Validation loss: 0.634801496
7516065
Epoch 26: Train err: 0.1875, Train loss: 0.39792571806907656 |Validation err: 0.295, Validation loss: 0.64646175038
06949
Epoch 27: Train err: 0.184375, Train loss: 0.3930838825702667 |Validation err: 0.31, Validation loss: 0.64248665794
73019
Epoch 28: Train err: 0.176, Train loss: 0.38016000711917874 |Validation err: 0.3105, Validation loss: 0.65657526999
71199
Epoch 29: Train err: 0.166625, Train loss: 0.3643355256319046 |Validation err: 0.33, Validation loss: 0.78958085179
32892
Epoch 30: Train err: 0.158375, Train loss: 0.3545009701251984 |Validation err: 0.303, Validation loss: 0.6914397869
259119
Finished Training
Total time elapsed: 164.74 seconds
```

Answer:

Time for training small network is 141.55s, and for large network is 164.74s. Large network has much more parameters to update so it takes longer.

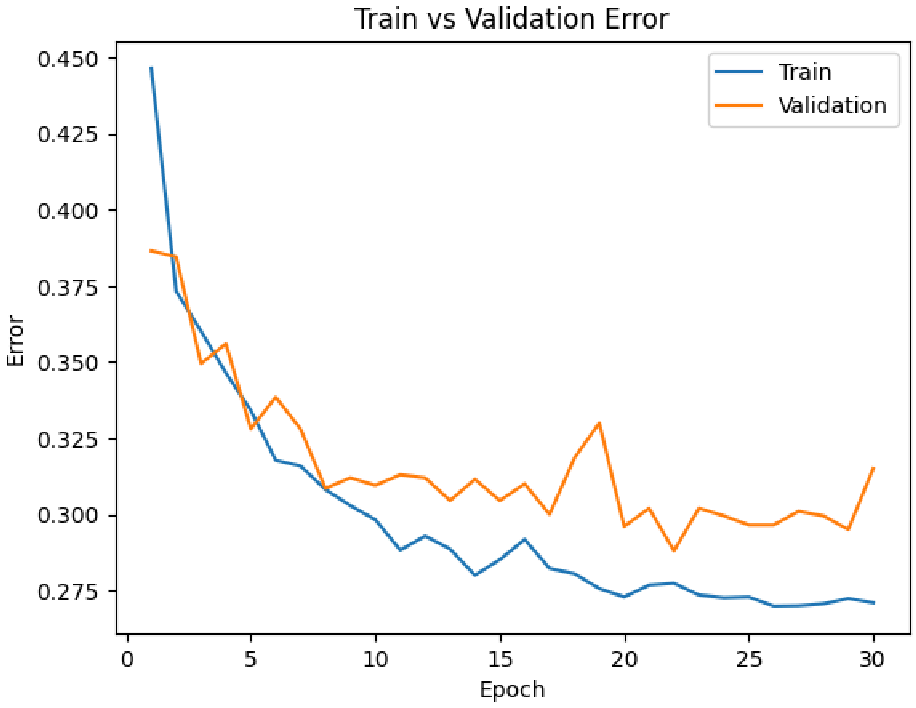
Part (e) - 2pt

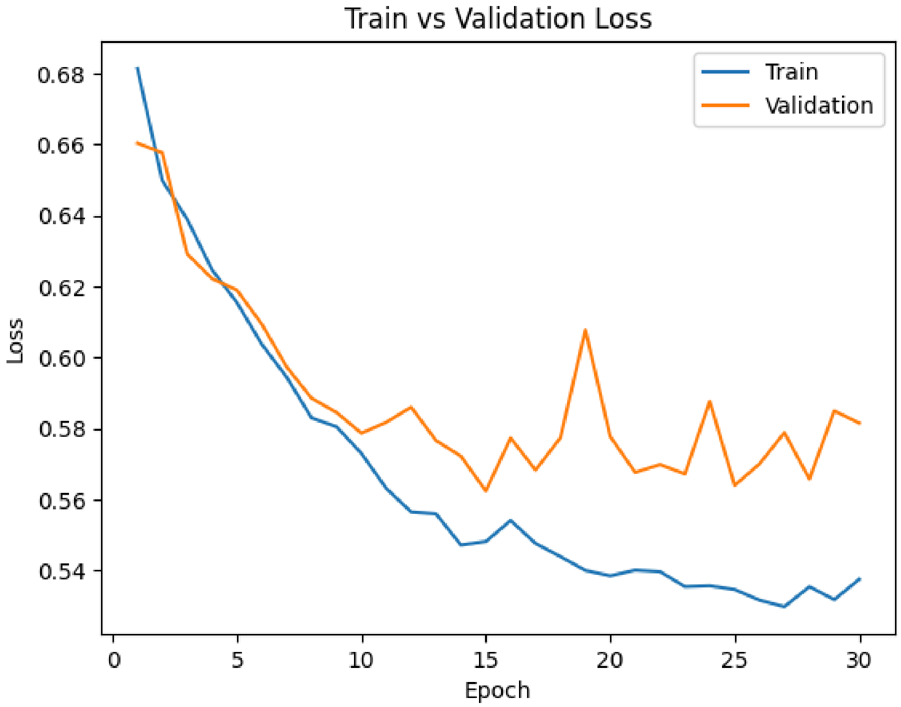
Use the function `plot_training_curve` to display the trajectory of the training/validation error and the training/validation loss. You will need to use the function `get_model_name` to generate the argument to the `plot_training_curve` function.

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

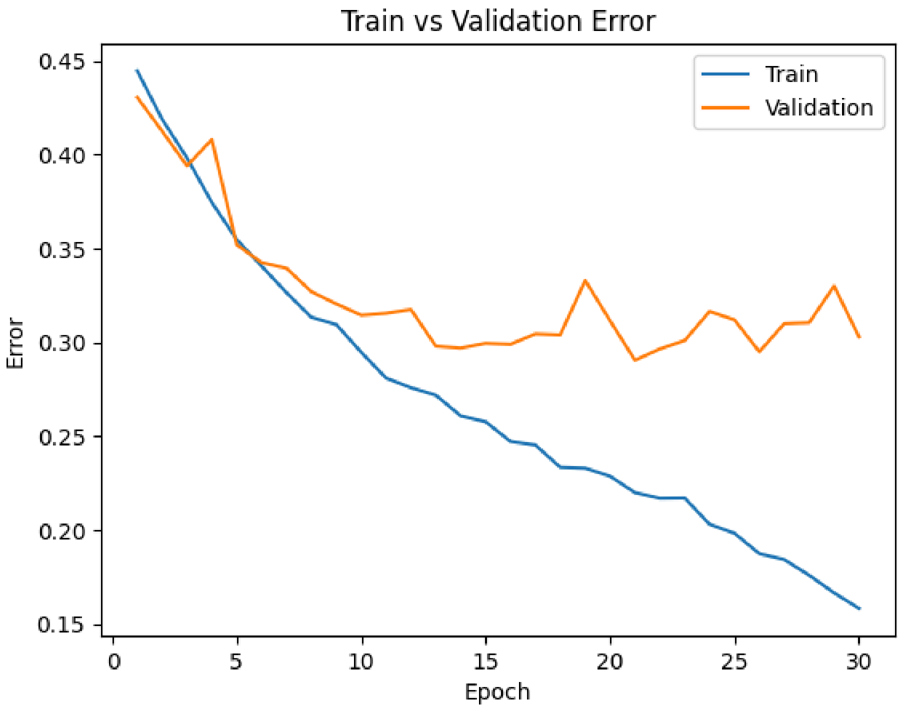
In [16]:
```
#model_path = get_model_name("small", batch_size=??, learning_rate=??, epoch=29)
small_path = get_model_name("small", batch_size=64, learning_rate=0.01, epoch=29)
large_path = get_model_name("large", batch_size=64, learning_rate=0.01, epoch=29)

print("Small Network Plot: ")
plot_training_curve(small_path)
print("Large Network Plot: ")
plot_training_curve(large_path)
```

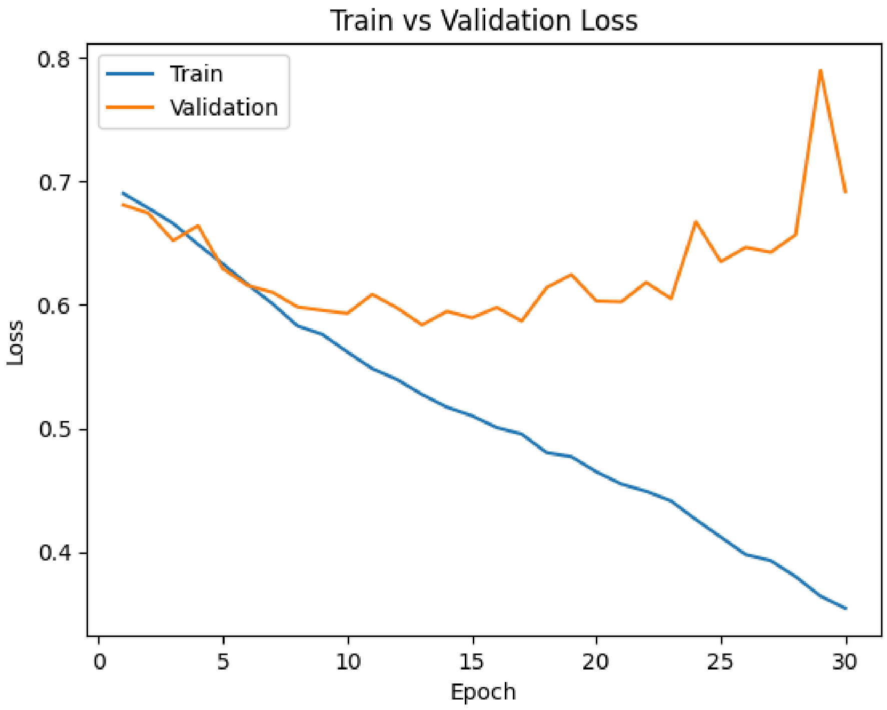
```
Small Network Plot:
```





```
Large Network Plot:
```





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:

At first, both networks' training and validation loss and error decrease steadily as the optimizer refines parameters, but around epoch 5 for the large net and epoch 10 for the small net the validation curves stop improving. The small network's validation error and loss then plateau around the same level—indicating limited capacity and mild underfitting—while the large network's validation loss actually rises even as its training error falls, signaling clear overfitting (i.e., the model memorizes the training set and fails to generalize).

Part 3. Optimization Parameters [12 pt]

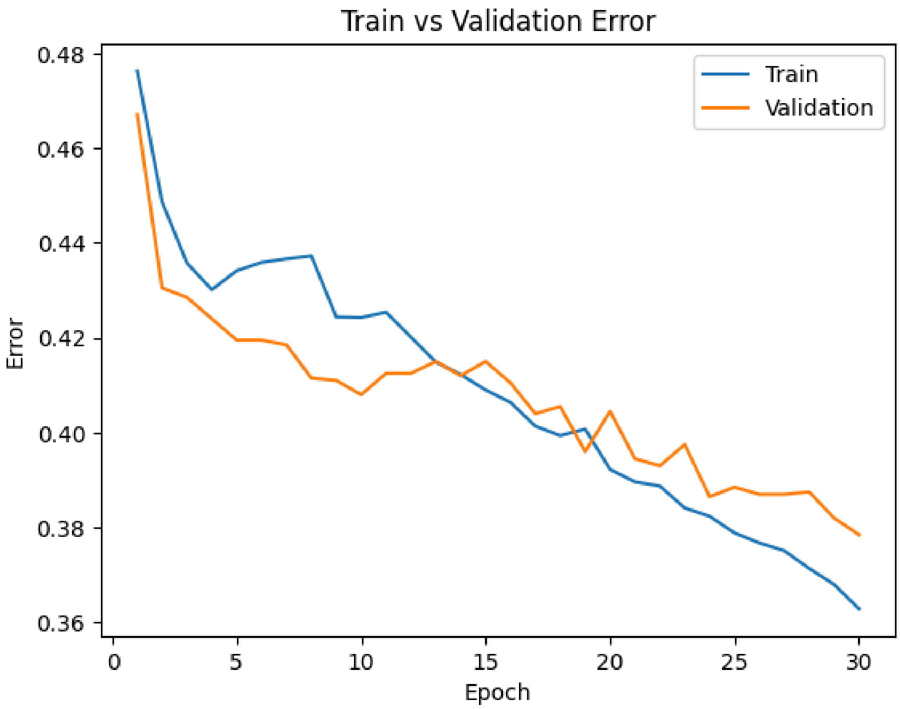
For this section, we will work with `large_net` only.

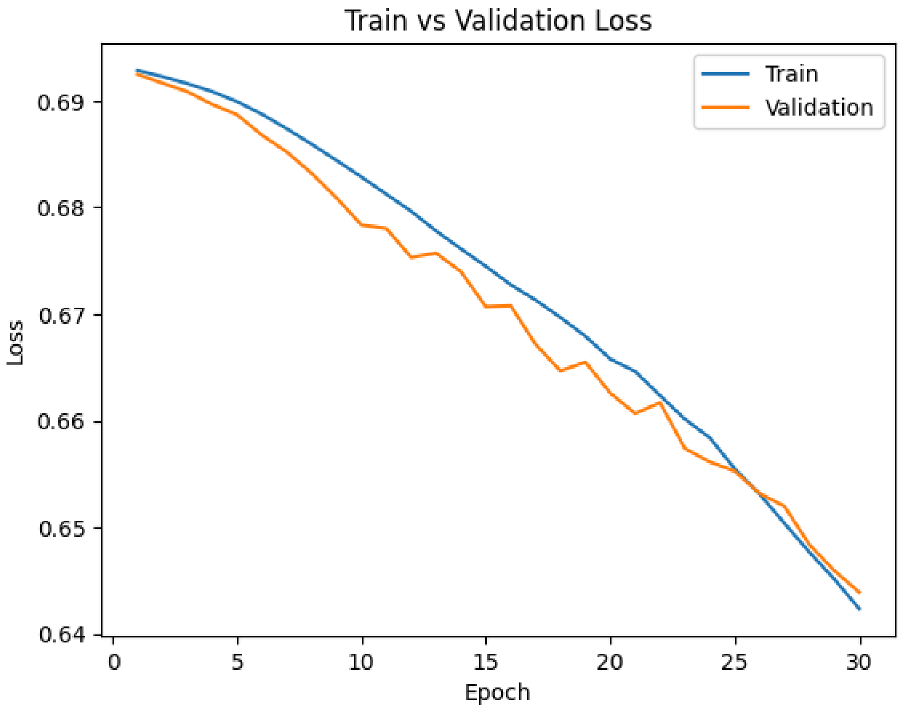
Part (a) - 3pt

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 [17]:
```
# 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)
```

```
Epoch 1: Train err: 0.47625, Train loss: 0.6928360004425049 |Validation err: 0.467, Validation loss: 0.692468658089
6378
Epoch 2: Train err: 0.448625, Train loss: 0.6922589716911316 |Validation err: 0.4305, Validation loss: 0.6916493363
678455
Epoch 3: Train err: 0.43575, Train loss: 0.6916067404747009 |Validation err: 0.4285, Validation loss: 0.69085445255
04112
Epoch 4: Train err: 0.430125, Train loss: 0.6908613877296448 |Validation err: 0.424, Validation loss: 0.68965969607
234
Epoch 5: Train err: 0.434125, Train loss: 0.6899198365211486 |Validation err: 0.4195, Validation loss: 0.6886942796
40913
Epoch 6: Train err: 0.435875, Train loss: 0.6887419748306275 |Validation err: 0.4195, Validation loss: 0.6867837496
101856
Epoch 7: Train err: 0.436625, Train loss: 0.6873781814575195 |Validation err: 0.4185, Validation loss: 0.6851996649
056673
Epoch 8: Train err: 0.43725, Train loss: 0.6859267811775207 |Validation err: 0.4115, Validation loss: 0.68319912441
07485
Epoch 9: Train err: 0.424375, Train loss: 0.6844044809341431 |Validation err: 0.411, Validation loss: 0.68088660389
18495
Epoch 10: Train err: 0.42425, Train loss: 0.6828490204811096 |Validation err: 0.408, Validation loss: 0.67835014313
4594
Epoch 11: Train err: 0.425375, Train loss: 0.6812362918853759 |Validation err: 0.4125, Validation loss: 0.678020710
1255655
Epoch 12: Train err: 0.420125, Train loss: 0.6796324462890625 |Validation err: 0.4125, Validation loss: 0.675314700
2309561
Epoch 13: Train err: 0.414875, Train loss: 0.6777921686172486 |Validation err: 0.415, Validation loss: 0.6757054049
521685
Epoch 14: Train err: 0.41225, Train loss: 0.6761114087104797 |Validation err: 0.412, Validation loss: 0.67397416010
499
Epoch 15: Train err: 0.409, Train loss: 0.6744724254608154 |Validation err: 0.415, Validation loss: 0.6706814523786
306
Epoch 16: Train err: 0.406375, Train loss: 0.6727445869445801 |Validation err: 0.4105, Validation loss: 0.670771589
5026922
Epoch 17: Train err: 0.401375, Train loss: 0.67130890417099 |Validation err: 0.404, Validation loss: 0.667154807597
3988
Epoch 18: Train err: 0.399375, Train loss: 0.6696756863594056 |Validation err: 0.4055, Validation loss: 0.664682278
4096003
Epoch 19: Train err: 0.40075, Train loss: 0.6679100017547608 |Validation err: 0.396, Validation loss: 0.66550736874
34196
Epoch 20: Train err: 0.39225, Train loss: 0.665790048122406 |Validation err: 0.4045, Validation loss: 0.66260510496
79518
Epoch 21: Train err: 0.389625, Train loss: 0.6646309685707092 |Validation err: 0.3945, Validation loss: 0.660683380
4398775
Epoch 22: Train err: 0.38875, Train loss: 0.6623744034767151 |Validation err: 0.393, Validation loss: 0.66170048713
68408
Epoch 23: Train err: 0.384125, Train loss: 0.6601551241874695 |Validation err: 0.3975, Validation loss: 0.657400136
8135214
Epoch 24: Train err: 0.382375, Train loss: 0.6584072341918945 |Validation err: 0.3865, Validation loss: 0.656140377
7450323
Epoch 25: Train err: 0.378875, Train loss: 0.6555052490234375 |Validation err: 0.3885, Validation loss: 0.655285593
1222439
Epoch 26: Train err: 0.37675, Train loss: 0.6531328277587891 |Validation err: 0.387, Validation loss: 0.65318292379
37927
Epoch 27: Train err: 0.375125, Train loss: 0.6503917617797852 |Validation err: 0.387, Validation loss: 0.6519917994
737625
Epoch 28: Train err: 0.371375, Train loss: 0.647677414894104 |Validation err: 0.3875, Validation loss: 0.6483855955
302715
Epoch 29: Train err: 0.368, Train loss: 0.6451585369110108 |Validation err: 0.382, Validation loss: 0.6459537353366
613
Epoch 30: Train err: 0.362875, Train loss: 0.6423822708129883 |Validation err: 0.3785, Validation loss: 0.643928943
2018995
Finished Training
Total time elapsed: 166.52 seconds
```





Answer:

It runs 1.78 seconds longer than the default setting, this overall runtime are very similar. With a lower learning rate, training error and loss both increase, and validation error and loss rise as well. However, the training and validation curves are closer together, indicating that overfitting, as observed when the learning rate is 0.01, is avoided.

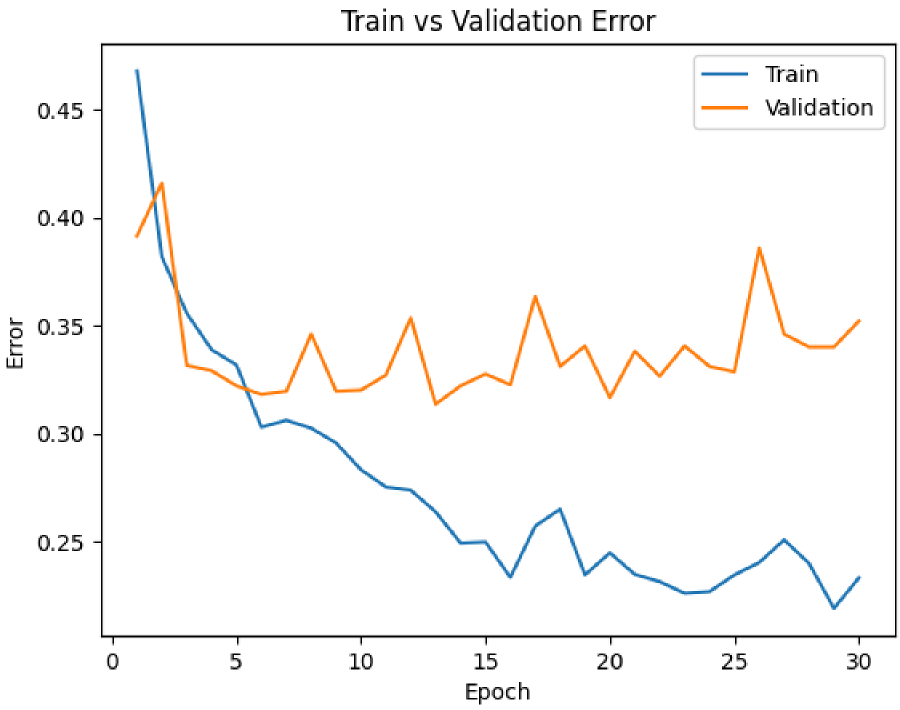
Part (b) - 3pt

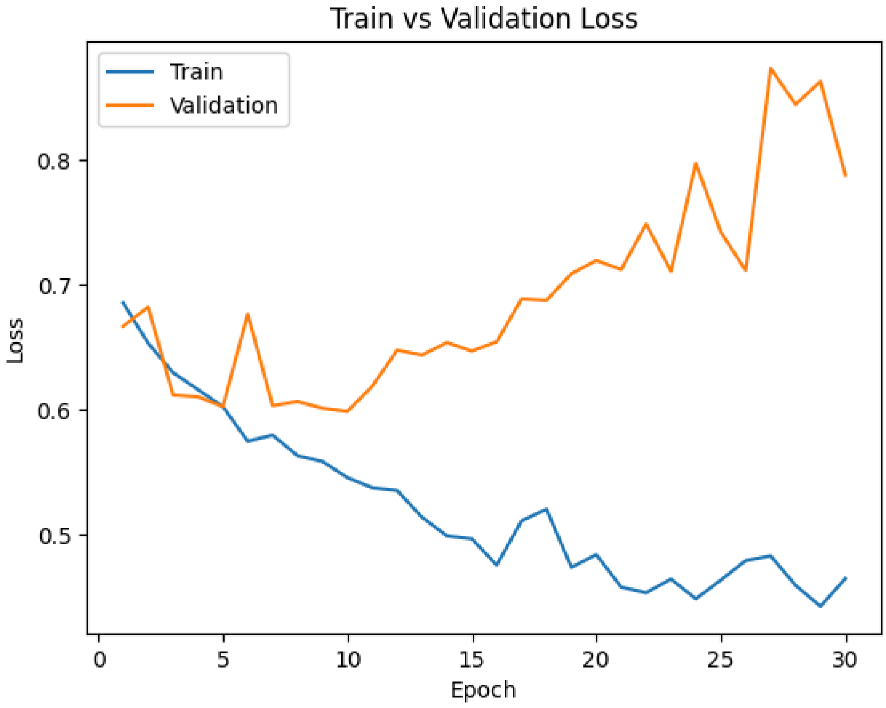
Train `large_net` with all default parameters, except set `learning_rate=0.1`. Does the model take longer/shorter to train? Plot the training curve. Describe the effect of *increasing* the learning rate.

In [19]:

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

```
Epoch 1: Train err: 0.468, Train loss: 0.6859708547592163 |Validation err: 0.3915, Validation loss: 0.6671512890607
119
Epoch 2: Train err: 0.38175, Train loss: 0.6535712971687316 |Validation err: 0.416, Validation loss: 0.682555403560
4
Epoch 3: Train err: 0.355625, Train loss: 0.6300361003875733 |Validation err: 0.3315, Validation loss: 0.6121374126
523733
Epoch 4: Train err: 0.33875, Train loss: 0.6164356069564819 |Validation err: 0.329, Validation loss: 0.610697885975
2417
Epoch 5: Train err: 0.331625, Train loss: 0.6028243894577027 |Validation err: 0.322, Validation loss: 0.60282201413
0652
Epoch 6: Train err: 0.302875, Train loss: 0.5750768322944642 |Validation err: 0.318, Validation loss: 0.67688752338
29021
Epoch 7: Train err: 0.306, Train loss: 0.579937388420105 |Validation err: 0.3195, Validation loss: 0.60362766031175
85
Epoch 8: Train err: 0.302375, Train loss: 0.563445684671402 |Validation err: 0.346, Validation loss: 0.606876859441
3996
Epoch 9: Train err: 0.2955, Train loss: 0.5588727197647094 |Validation err: 0.3195, Validation loss: 0.601520698517
561
Epoch 10: Train err: 0.283125, Train loss: 0.545897435426712 |Validation err: 0.32, Validation loss: 0.598896564915
7763
Epoch 11: Train err: 0.275125, Train loss: 0.5378088033199311 |Validation err: 0.327, Validation loss: 0.6191476769
74535
Epoch 12: Train err: 0.273625, Train loss: 0.535594067811966 |Validation err: 0.3535, Validation loss: 0.6480005066
841841
Epoch 13: Train err: 0.263625, Train loss: 0.5141938621997834 |Validation err: 0.3135, Validation loss: 0.644118250
3476739
Epoch 14: Train err: 0.249125, Train loss: 0.4992824411392212 |Validation err: 0.322, Validation loss: 0.6540486551
821232
Epoch 15: Train err: 0.249625, Train loss: 0.4969462950229645 |Validation err: 0.3275, Validation loss: 0.647324764
1697526
Epoch 16: Train err: 0.23325, Train loss: 0.4758114631175995 |Validation err: 0.3225, Validation loss: 0.6547067314
386368
Epoch 17: Train err: 0.257, Train loss: 0.5113468255996704 |Validation err: 0.3635, Validation loss: 0.689136173576
1166
Epoch 18: Train err: 0.264875, Train loss: 0.5206713597774506 |Validation err: 0.331, Validation loss: 0.6878945203
498006
Epoch 19: Train err: 0.234375, Train loss: 0.47404373025894164 |Validation err: 0.3405, Validation loss: 0.70930768
54944229
Epoch 20: Train err: 0.244625, Train loss: 0.4842326376438141 |Validation err: 0.3165, Validation loss: 0.719858558
8485003
Epoch 21: Train err: 0.234625, Train loss: 0.45815866231918334 |Validation err: 0.338, Validation loss: 0.712642149
9997377
Epoch 22: Train err: 0.23125, Train loss: 0.45371274280548096 |Validation err: 0.3265, Validation loss: 0.749026388
861239
Epoch 23: Train err: 0.225875, Train loss: 0.464591454744339 |Validation err: 0.3405, Validation loss: 0.7113211695
104837
Epoch 24: Train err: 0.226625, Train loss: 0.44882566618919373 |Validation err: 0.331, Validation loss: 0.797717930
7490587
Epoch 25: Train err: 0.234375, Train loss: 0.4637635123729706 |Validation err: 0.3285, Validation loss: 0.742719415
5752659
Epoch 26: Train err: 0.240125, Train loss: 0.4794488083124161 |Validation err: 0.386, Validation loss: 0.7118616551
160812
Epoch 27: Train err: 0.250625, Train loss: 0.48315389394760133 |Validation err: 0.346, Validation loss: 0.873822448
7751722
Epoch 28: Train err: 0.23975, Train loss: 0.4596183142662048 |Validation err: 0.34, Validation loss: 0.844945298507
8096
Epoch 29: Train err: 0.218875, Train loss: 0.4428225445747376 |Validation err: 0.34, Validation loss: 0.86330940388
14306
Epoch 30: Train err: 0.233, Train loss: 0.46504191660881045 |Validation err: 0.352, Validation loss: 0.788372695446
0144
Finished Training
Total time elapsed: 156.76 seconds
```





Answer:

It takes about 7 seconds longer than the default setting, but overall runtime remains similar. Raising the learning rate from 0.01 to 0.1 causes larger gradient-descent steps, so error and loss change more abruptly and overfitting appears earlier compared to the default model. As a result, the curves oscillate more and have sharper angles—indicating that a learning rate of 0.1 is too aggressive for this model.

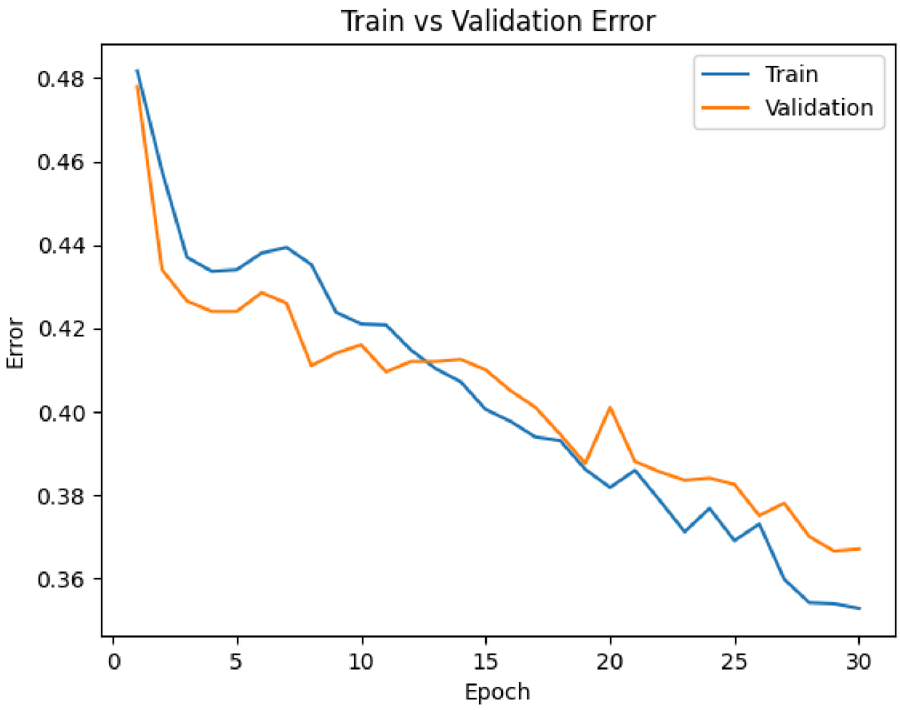
Part (c) - 3pt

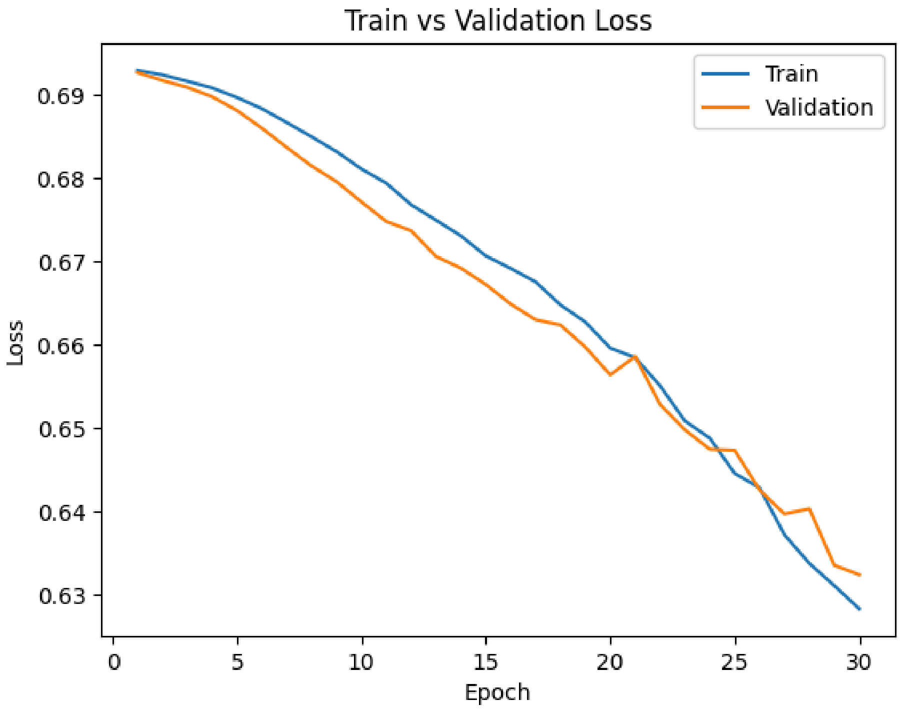
Train `large_net` with all default parameters, including with `learning_rate=0.01`. Now, set `batch_size=512`. Does the model take longer/shorter to train? Plot the training curve. Describe the effect of *increasing* the batch size.

In [20]:

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

```
Epoch 1: Train err: 0.48175, Train loss: 0.6929379552602768 |Validation err: 0.478, Validation loss: 0.692682400345
8023
Epoch 2: Train err: 0.457625, Train loss: 0.692410409450531 |Validation err: 0.434, Validation loss: 0.691742524504
6616
Epoch 3: Train err: 0.437, Train loss: 0.6916500627994537 |Validation err: 0.4265, Validation loss: 0.6909130215644
836
Epoch 4: Train err: 0.433625, Train loss: 0.6908449903130531 |Validation err: 0.424, Validation loss: 0.68978703022
00317
Epoch 5: Train err: 0.434, Train loss: 0.6896935515105724 |Validation err: 0.424, Validation loss: 0.68813550472259
52
Epoch 6: Train err: 0.438, Train loss: 0.6883532106876373 |Validation err: 0.4285, Validation loss: 0.6860118657350
54
Epoch 7: Train err: 0.439375, Train loss: 0.6866871826350689 |Validation err: 0.426, Validation loss: 0.68369686603
54614
Epoch 8: Train err: 0.43525, Train loss: 0.6849770620465279 |Validation err: 0.411, Validation loss: 0.681467264890
6708
Epoch 9: Train err: 0.42375, Train loss: 0.6832008883357048 |Validation err: 0.414, Validation loss: 0.679591417312
6221
Epoch 10: Train err: 0.421, Train loss: 0.6811088025569916 |Validation err: 0.416, Validation loss: 0.6771541684865
952
Epoch 11: Train err: 0.42075, Train loss: 0.6794030703604221 |Validation err: 0.4095, Validation loss: 0.6748128235
340118
Epoch 12: Train err: 0.41475, Train loss: 0.6768062897026539 |Validation err: 0.412, Validation loss: 0.67370671033
85925
Epoch 13: Train err: 0.41025, Train loss: 0.6749698966741562 |Validation err: 0.412, Validation loss: 0.67061218619
34662
Epoch 14: Train err: 0.407125, Train loss: 0.6730909235775471 |Validation err: 0.4125, Validation loss: 0.669210046
5297699
Epoch 15: Train err: 0.4005, Train loss: 0.6706824786961079 |Validation err: 0.41, Validation loss: 0.6672518998384
476
Epoch 16: Train err: 0.397625, Train loss: 0.669177521020174 |Validation err: 0.405, Validation loss: 0.66490471363
06763
Epoch 17: Train err: 0.393875, Train loss: 0.6675704158842564 |Validation err: 0.401, Validation loss: 0.6630221307
277679
Epoch 18: Train err: 0.393, Train loss: 0.6648007929325104 |Validation err: 0.3945, Validation loss: 0.662391513586
0443
Epoch 19: Train err: 0.386125, Train loss: 0.6627416461706161 |Validation err: 0.3875, Validation loss: 0.659724175
9300232
Epoch 20: Train err: 0.38175, Train loss: 0.659615233540535 |Validation err: 0.401, Validation loss: 0.656416386365
8905
Epoch 21: Train err: 0.385875, Train loss: 0.6584861390292645 |Validation err: 0.388, Validation loss: 0.6586255580
186844
Epoch 22: Train err: 0.378625, Train loss: 0.6551117822527885 |Validation err: 0.3855, Validation loss: 0.652862340
2118683
Epoch 23: Train err: 0.371125, Train loss: 0.6508784741163254 |Validation err: 0.3835, Validation loss: 0.649810582
3993683
Epoch 24: Train err: 0.37675, Train loss: 0.6488191038370132 |Validation err: 0.384, Validation loss: 0.64748761057
8537
Epoch 25: Train err: 0.369, Train loss: 0.6445828042924404 |Validation err: 0.3825, Validation loss: 0.647354021668
4341
Epoch 26: Train err: 0.373, Train loss: 0.6428647711873055 |Validation err: 0.375, Validation loss: 0.6425962299108
505
Epoch 27: Train err: 0.35975, Train loss: 0.6372342072427273 |Validation err: 0.378, Validation loss: 0.63973273336
88736
Epoch 28: Train err: 0.354125, Train loss: 0.6337889917194843 |Validation err: 0.37, Validation loss: 0.64034420251
84631
Epoch 29: Train err: 0.353875, Train loss: 0.6311212778091431 |Validation err: 0.3665, Validation loss: 0.633566260
3378296
Epoch 30: Train err: 0.35275, Train loss: 0.6283531747758389 |Validation err: 0.367, Validation loss: 0.63244122266
76941
Finished Training
Total time elapsed: 146.74 seconds
```





Answer:

With a larger batch size, training runs about 18 seconds faster since there are fewer iterations per epoch. Although the training error and loss are higher, the increased batch size reduces overfitting: the loss curves become smoother and the gap between training and validation curves narrows.

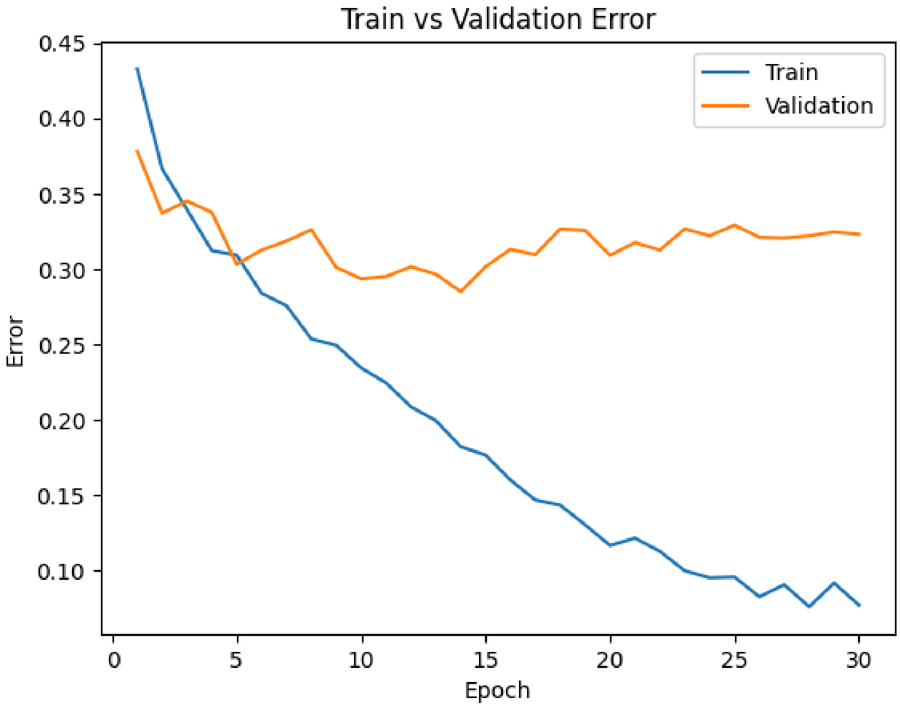
Part (d) - 3pt

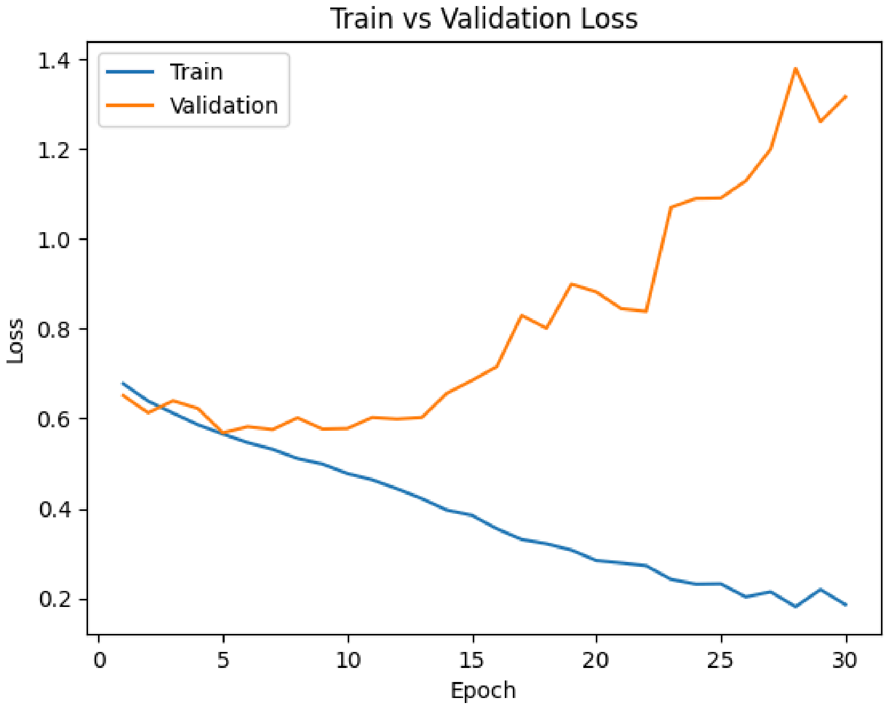
Train `large_net` with all default parameters, including with `learning_rate=0.01`. Now, set `batch_size=16`. Does the model take longer/shorter to train? Plot the training curve. Describe the effect of *decreasing* the batch size.

In [21]:

```
large_net = LargeNet()
train_net(large_net, 16, 0.01, 30)

large_path = get_model_name("large", batch_size=16, learning_rate=0.01, epoch=29)
plot_training_curve(large_path)
```

```
Epoch 1: Train err: 0.432625, Train loss: 0.6775506126880646 |Validation err: 0.378, Validation loss: 0.65125719738
00659
Epoch 2: Train err: 0.366375, Train loss: 0.6387728816270828 |Validation err: 0.337, Validation loss: 0.61274223566
0553
Epoch 3: Train err: 0.339375, Train loss: 0.6119522891640663 |Validation err: 0.345, Validation loss: 0.63963562130
92804
Epoch 4: Train err: 0.312125, Train loss: 0.5861616842746734 |Validation err: 0.3375, Validation loss: 0.6223122742
176056
Epoch 5: Train err: 0.309125, Train loss: 0.5655454085469246 |Validation err: 0.303, Validation loss: 0.56827199125
28992
Epoch 6: Train err: 0.283875, Train loss: 0.546434996843338 |Validation err: 0.3125, Validation loss: 0.58191383647
9187
Epoch 7: Train err: 0.275625, Train loss: 0.5315411986708641 |Validation err: 0.3185, Validation loss: 0.5755203785
896301
Epoch 8: Train err: 0.253625, Train loss: 0.5110043309926987 |Validation err: 0.326, Validation loss: 0.60160018277
16827
Epoch 9: Train err: 0.249375, Train loss: 0.4988122125864029 |Validation err: 0.301, Validation loss: 0.57669455218
31512
Epoch 10: Train err: 0.234375, Train loss: 0.47769227081537247 |Validation err: 0.2935, Validation loss: 0.57782985
68725586
Epoch 11: Train err: 0.2245, Train loss: 0.4637914804518223 |Validation err: 0.295, Validation loss: 0.602375275135
0403
Epoch 12: Train err: 0.208625, Train loss: 0.4436814887523651 |Validation err: 0.3015, Validation loss: 0.598846314
6686553
Epoch 13: Train err: 0.1995, Train loss: 0.42190410897135733 |Validation err: 0.2965, Validation loss: 0.6023776261
806488
Epoch 14: Train err: 0.18225, Train loss: 0.3964607211649418 |Validation err: 0.285, Validation loss: 0.65630778694
15284
Epoch 15: Train err: 0.176625, Train loss: 0.38504845155775547 |Validation err: 0.3015, Validation loss: 0.68457303
7981987
Epoch 16: Train err: 0.160125, Train loss: 0.35515162971615793 |Validation err: 0.313, Validation loss: 0.715396178
2455445
Epoch 17: Train err: 0.146875, Train loss: 0.33127270932495595 |Validation err: 0.3095, Validation loss: 0.82926645
50542831
Epoch 18: Train err: 0.1435, Train loss: 0.3216609486192465 |Validation err: 0.3265, Validation loss: 0.80052021336
55548
Epoch 19: Train err: 0.130375, Train loss: 0.30764051334559916 |Validation err: 0.3255, Validation loss: 0.89872837
56732941
Epoch 20: Train err: 0.11675, Train loss: 0.28437252435833216 |Validation err: 0.309, Validation loss: 0.8813227322
101593
Epoch 21: Train err: 0.121625, Train loss: 0.27901479678601027 |Validation err: 0.3175, Validation loss: 0.84411867
98810959
Epoch 22: Train err: 0.112875, Train loss: 0.2728129132091999 |Validation err: 0.3125, Validation loss: 0.838263761
4011764
Epoch 23: Train err: 0.1, Train loss: 0.24255631332099437 |Validation err: 0.3265, Validation loss: 1.0690370055437
088
Epoch 24: Train err: 0.095375, Train loss: 0.2318725396282971 |Validation err: 0.322, Validation loss: 1.0889509381
055833
Epoch 25: Train err: 0.095875, Train loss: 0.23257503168098628 |Validation err: 0.329, Validation loss: 1.089875996
5896607
Epoch 26: Train err: 0.08275, Train loss: 0.20350570978596808 |Validation err: 0.321, Validation loss: 1.1279444985
38971
Epoch 27: Train err: 0.09075, Train loss: 0.2147584371343255 |Validation err: 0.3205, Validation loss: 1.1985667142
868042
Epoch 28: Train err: 0.076125, Train loss: 0.18171320636058227 |Validation err: 0.322, Validation loss: 1.378227489
9482727
Epoch 29: Train err: 0.091875, Train loss: 0.21968170262500644 |Validation err: 0.3245, Validation loss: 1.26013917
98257828
Epoch 30: Train err: 0.07725, Train loss: 0.1865569370612502 |Validation err: 0.323, Validation loss: 1.31479527735
71014
Finished Training
Total time elapsed: 225.82 seconds
```





Answer:

This takes around 61 seconds longer, decreasing the batch size increases training time since more iterations are needed per epoch. It does, however, lower the training error and loss and the overfitting occurs earlier. For validation, the error and loss is still large.

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:

I would chose large network (with more parameters), large batch size, and small learning rate. So, I choose network = large_net; batch_size = 128, learning_rate = 0.01, and epoch = 30.

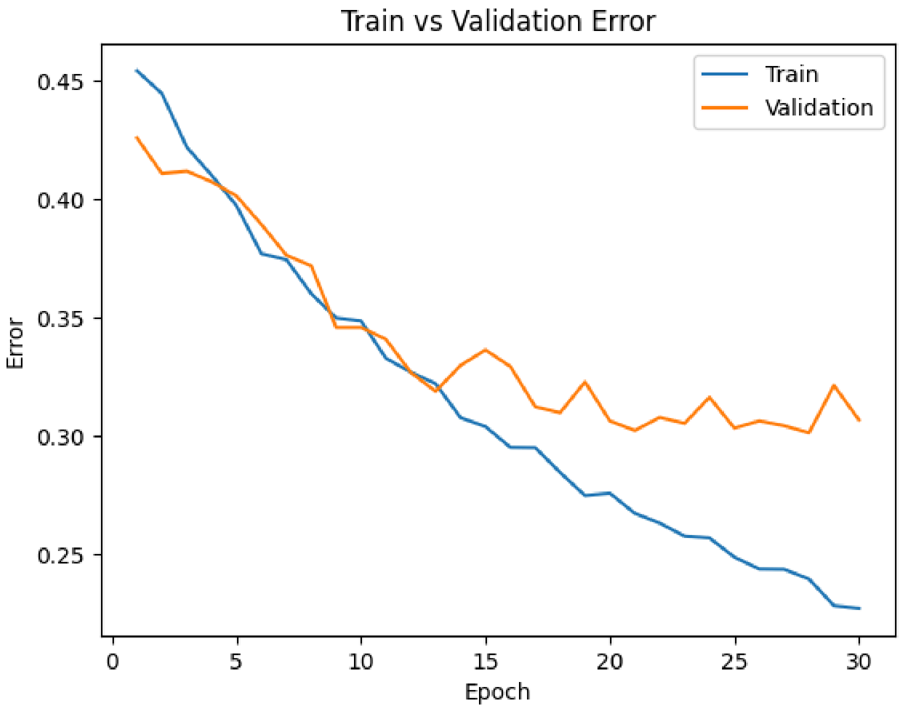
The large network outperforms the small network thanks to its greater parameter capacity. From Part 3, a learning rate of 0.01 proved to be well balanced. Small batch sizes tend to cause overfitting, so using a larger batch improves performance without overfitting. Finally, training for 30 epochs strikes a good balance between underfitting and overfitting.

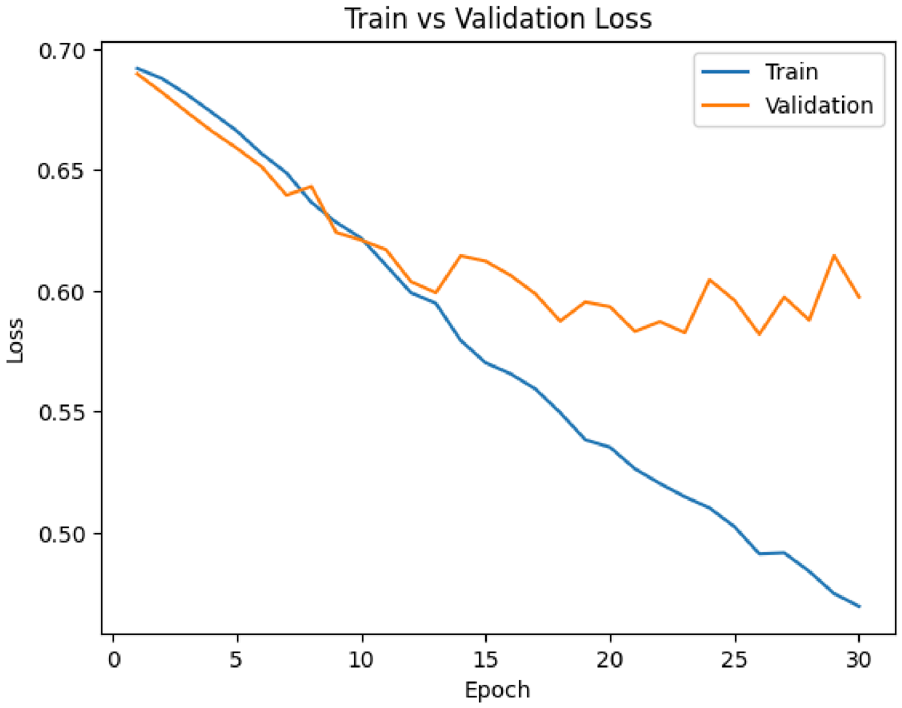
Part (b) - 1pt

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

```
In [22]: large_net = LargeNet()
         train_net(large_net, 128, 0.01, 30)
         model_path_large = get_model_name("large", batch_size=128, learning_rate=0.01, epoch=29)
         plot_training_curve(model_path_large)
```

```
Epoch 1: Train err: 0.454375, Train loss: 0.6920222280517457 |Validation err: 0.426, Validation loss: 0.68975755199
79
Epoch 2: Train err: 0.44475, Train loss: 0.6877915528085496 |Validation err: 0.411, Validation loss: 0.682064440101
3851
Epoch 3: Train err: 0.422125, Train loss: 0.6812269706574697 |Validation err: 0.412, Validation loss: 0.67389298602
93865
Epoch 4: Train err: 0.41025, Train loss: 0.673797849624876 |Validation err: 0.4075, Validation loss: 0.666033994406
4617
Epoch 5: Train err: 0.397375, Train loss: 0.6661591208170331 |Validation err: 0.4015, Validation loss: 0.6590236835
181713
Epoch 6: Train err: 0.377125, Train loss: 0.6567326358386448 |Validation err: 0.3895, Validation loss: 0.6513406559
824944
Epoch 7: Train err: 0.374625, Train loss: 0.6487630000190129 |Validation err: 0.3765, Validation loss: 0.6395393237
47158
Epoch 8: Train err: 0.36025, Train loss: 0.6365784785104176 |Validation err: 0.372, Validation loss: 0.643158335238
6951
Epoch 9: Train err: 0.35, Train loss: 0.6283171810801067 |Validation err: 0.346, Validation loss: 0.624148305505514
1
Epoch 10: Train err: 0.34875, Train loss: 0.6217929465430123 |Validation err: 0.346, Validation loss: 0.62099678441
88213
Epoch 11: Train err: 0.333, Train loss: 0.6105351911650764 |Validation err: 0.341, Validation loss: 0.6170116998255
253
Epoch 12: Train err: 0.327125, Train loss: 0.5992055573160686 |Validation err: 0.327, Validation loss: 0.6038246862
590313
Epoch 13: Train err: 0.32225, Train loss: 0.5948410185556563 |Validation err: 0.319, Validation loss: 0.59919760376
21498
Epoch 14: Train err: 0.307875, Train loss: 0.5794498173017351 |Validation err: 0.33, Validation loss: 0.61448763683
4383
Epoch 15: Train err: 0.304125, Train loss: 0.5702668012134613 |Validation err: 0.3365, Validation loss: 0.612275071
4421272
Epoch 16: Train err: 0.295375, Train loss: 0.5657135503632682 |Validation err: 0.3295, Validation loss: 0.606410942
9717064
Epoch 17: Train err: 0.29525, Train loss: 0.5594405839367519 |Validation err: 0.3125, Validation loss: 0.5987546071
410179
Epoch 18: Train err: 0.28475, Train loss: 0.5496182214646113 |Validation err: 0.31, Validation loss: 0.587442725896
8353
Epoch 19: Train err: 0.275, Train loss: 0.5384441462774125 |Validation err: 0.323, Validation loss: 0.5953904837369
919
Epoch 20: Train err: 0.276, Train loss: 0.5352702755776663 |Validation err: 0.3065, Validation loss: 0.593407429754
734
Epoch 21: Train err: 0.2675, Train loss: 0.5263815578960237 |Validation err: 0.3025, Validation loss: 0.58318453282
11784
Epoch 22: Train err: 0.263375, Train loss: 0.5202986276338971 |Validation err: 0.308, Validation loss: 0.5872550122
439861
Epoch 23: Train err: 0.257875, Train loss: 0.5148504934613667 |Validation err: 0.3055, Validation loss: 0.582683462
6495838
Epoch 24: Train err: 0.257125, Train loss: 0.5102080371644762 |Validation err: 0.3165, Validation loss: 0.604580949
9919415
Epoch 25: Train err: 0.249, Train loss: 0.5024650309767041 |Validation err: 0.3035, Validation loss: 0.596042184159
1597
Epoch 26: Train err: 0.244, Train loss: 0.491200448028625 |Validation err: 0.3065, Validation loss: 0.5820110738277
435
Epoch 27: Train err: 0.243875, Train loss: 0.491646780381127 |Validation err: 0.3045, Validation loss: 0.5974767580
628395
Epoch 28: Train err: 0.23975, Train loss: 0.4839872277918316 |Validation err: 0.3015, Validation loss: 0.5878518670
797348
Epoch 29: Train err: 0.2285, Train loss: 0.4747762259036776 |Validation err: 0.3215, Validation loss: 0.61454588919
87801
Epoch 30: Train err: 0.227375, Train loss: 0.4695242298027826 |Validation err: 0.307, Validation loss: 0.5974648240
953684
Finished Training
Total time elapsed: 152.66 seconds
```





Part (c) - 2pt

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

Answer:

I'll use the large network with a batch size of 128, a learning rate of 0.005, and train for 50 epochs. I increased the epoch count to 50 to get more accurate data with less validation loss and error due to the smaller learning rate. Lowering the learning rate allows the model to make more precise parameter updates.

Part (d) - 1pt

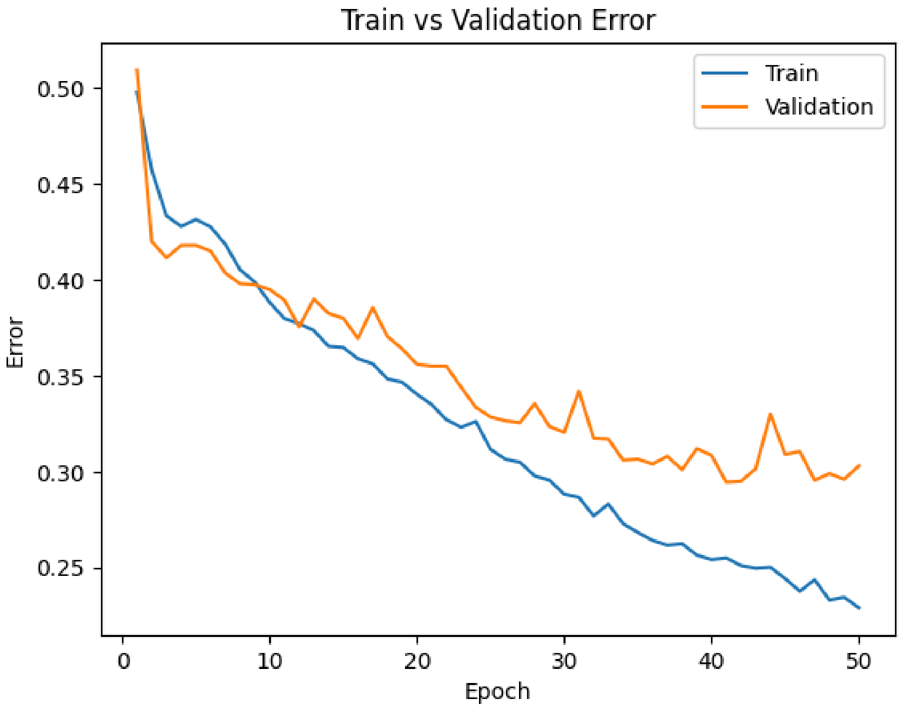
Train the model with the hyperparameters you chose in part(c), and include the training curve.

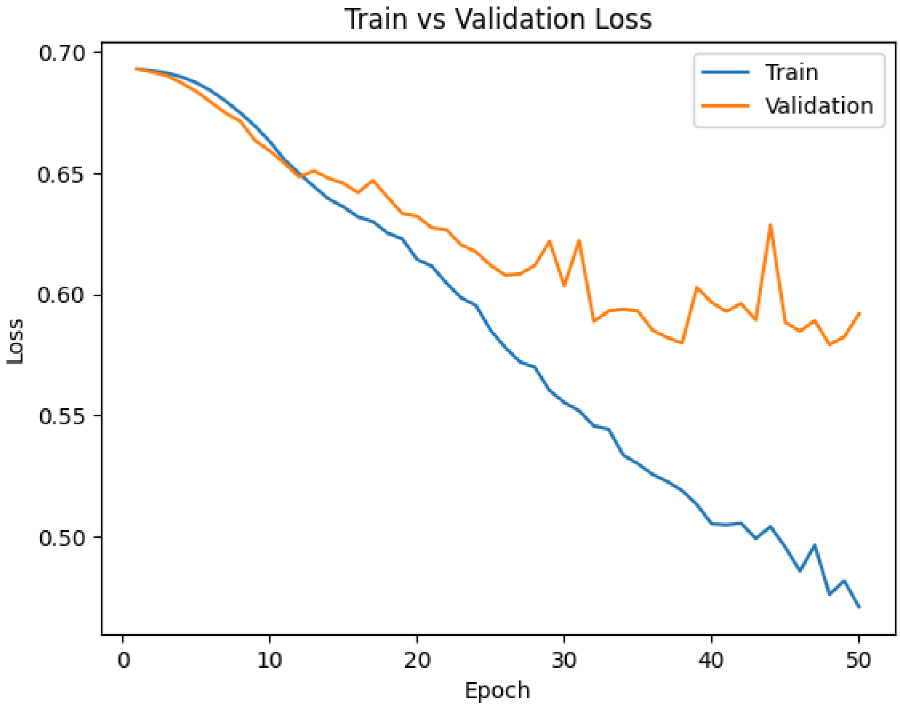
In [26]:

```
large_net = LargeNet()
train_net(large_net, 128, 0.005, 50)
model_path_large = get_model_name("large", batch_size=128, learning_rate=0.005, epoch=49)
plot_training_curve(model_path_large)
```

```
Epoch 1: Train err: 0.49775, Train loss: 0.6930236144671365 |Validation err: 0.5095, Validation loss: 0.69302840903
40137
Epoch 2: Train err: 0.45725, Train loss: 0.6922709601266044 |Validation err: 0.42, Validation loss: 0.6916764788329
601
Epoch 3: Train err: 0.433375, Train loss: 0.6913018368539356 |Validation err: 0.4115, Validation loss: 0.6902153231
203556
Epoch 4: Train err: 0.427875, Train loss: 0.6897292752114553 |Validation err: 0.418, Validation loss: 0.68726608529
68693
Epoch 5: Train err: 0.4315, Train loss: 0.6873460137654864 |Validation err: 0.418, Validation loss: 0.6837649568915
367
Epoch 6: Train err: 0.427625, Train loss: 0.6840077771080865 |Validation err: 0.415, Validation loss: 0.67932287231
08768
Epoch 7: Train err: 0.4185, Train loss: 0.679767336164202 |Validation err: 0.4035, Validation loss: 0.6747710406780
243
Epoch 8: Train err: 0.40525, Train loss: 0.6748903526200188 |Validation err: 0.398, Validation loss: 0.671427298337
2211
Epoch 9: Train err: 0.398875, Train loss: 0.6694405656012278 |Validation err: 0.3975, Validation loss: 0.6635362096
130848
Epoch 10: Train err: 0.38825, Train loss: 0.6631133367144872 |Validation err: 0.395, Validation loss: 0.65923252701
75934
Epoch 11: Train err: 0.379875, Train loss: 0.6555238801335531 |Validation err: 0.3895, Validation loss: 0.654037024
8258114
Epoch 12: Train err: 0.377125, Train loss: 0.6498739918073019 |Validation err: 0.3755, Validation loss: 0.648442260
9210014
Epoch 13: Train err: 0.373625, Train loss: 0.6444921550296602 |Validation err: 0.39, Validation loss: 0.65095221996
30737
Epoch 14: Train err: 0.365375, Train loss: 0.6393705587538462 |Validation err: 0.3825, Validation loss: 0.647908397
0189095
Epoch 15: Train err: 0.36475, Train loss: 0.6360937640780494 |Validation err: 0.38, Validation loss: 0.645784851163
6257
Epoch 16: Train err: 0.358875, Train loss: 0.6319135682923454 |Validation err: 0.3695, Validation loss: 0.641961343
5864449
Epoch 17: Train err: 0.35625, Train loss: 0.6300226449966431 |Validation err: 0.3855, Validation loss: 0.6470319963
991642
Epoch 18: Train err: 0.348375, Train loss: 0.625270790523953 |Validation err: 0.3705, Validation loss: 0.6400919146
835804
Epoch 19: Train err: 0.346625, Train loss: 0.6228595243559943 |Validation err: 0.364, Validation loss: 0.6333517804
741859
Epoch 20: Train err: 0.340375, Train loss: 0.6144135934965951 |Validation err: 0.356, Validation loss: 0.6322385892
271996
Epoch 21: Train err: 0.335, Train loss: 0.6116347029095605 |Validation err: 0.355, Validation loss: 0.6274648457765
579
Epoch 22: Train err: 0.327125, Train loss: 0.6046467820803324 |Validation err: 0.355, Validation loss: 0.6266518495
976925
Epoch 23: Train err: 0.323125, Train loss: 0.5985852300174652 |Validation err: 0.344, Validation loss: 0.6204004175
961018
Epoch 24: Train err: 0.326125, Train loss: 0.5954039200903878 |Validation err: 0.3335, Validation loss: 0.617544036
3585949
Epoch 25: Train err: 0.311625, Train loss: 0.5851705992032611 |Validation err: 0.3285, Validation loss: 0.611986499
2797375
Epoch 26: Train err: 0.3065, Train loss: 0.5780152792022342 |Validation err: 0.3265, Validation loss: 0.60789039358
49667
Epoch 27: Train err: 0.304875, Train loss: 0.5721108657973153 |Validation err: 0.3255, Validation loss: 0.608403999
3584156
Epoch 28: Train err: 0.29775, Train loss: 0.5698803469302163 |Validation err: 0.3355, Validation loss: 0.6119430214
166641
Epoch 29: Train err: 0.2955, Train loss: 0.5603470627277617 |Validation err: 0.3235, Validation loss: 0.62185088917
61303
Epoch 30: Train err: 0.28825, Train loss: 0.5553928585279555 |Validation err: 0.3205, Validation loss: 0.6035062558
948994
Epoch 31: Train err: 0.286625, Train loss: 0.5519701001190004 |Validation err: 0.342, Validation loss: 0.6221818365
15665
Epoch 32: Train err: 0.276875, Train loss: 0.5457092022138929 |Validation err: 0.3175, Validation loss: 0.588867370
0392246
Epoch 33: Train err: 0.283125, Train loss: 0.5443956478247567 |Validation err: 0.317, Validation loss: 0.5931047797
203064
Epoch 34: Train err: 0.272875, Train loss: 0.533677642780637 |Validation err: 0.306, Validation loss: 0.59393728524
44649
Epoch 35: Train err: 0.26825, Train loss: 0.5300850234334431 |Validation err: 0.3065, Validation loss: 0.5930652040
988207
Epoch 36: Train err: 0.264125, Train loss: 0.5256856509617397 |Validation err: 0.304, Validation loss: 0.5851593222
469091
Epoch 37: Train err: 0.261625, Train loss: 0.5227410575700184 |Validation err: 0.308, Validation loss: 0.5821809023
618698
Epoch 38: Train err: 0.262375, Train loss: 0.5190259246599107 |Validation err: 0.301, Validation loss: 0.5799242295
324802
Epoch 39: Train err: 0.2565, Train loss: 0.5132552896227155 |Validation err: 0.312, Validation loss: 0.602854173630
```

```
476
Epoch 40: Train err: 0.254125, Train loss: 0.5053303662746672 |Validation err: 0.3085, Validation loss: 0.596704617
1426773
Epoch 41: Train err: 0.255, Train loss: 0.5049237902202304 |Validation err: 0.2945, Validation loss: 0.592937234789
1331
Epoch 42: Train err: 0.251, Train loss: 0.5055604023592812 |Validation err: 0.295, Validation loss: 0.5962063446640
968
Epoch 43: Train err: 0.249625, Train loss: 0.4992753877526238 |Validation err: 0.3015, Validation loss: 0.589449502
5277138
Epoch 44: Train err: 0.250125, Train loss: 0.5041983591185676 |Validation err: 0.33, Validation loss: 0.62867684662
34207
Epoch 45: Train err: 0.24425, Train loss: 0.4957300717868502 |Validation err: 0.309, Validation loss: 0.58854589983
82092
Epoch 46: Train err: 0.237625, Train loss: 0.4858499401145511 |Validation err: 0.3105, Validation loss: 0.584788557
1420193
Epoch 47: Train err: 0.243625, Train loss: 0.4965810960247403 |Validation err: 0.2955, Validation loss: 0.589199263
6024952
Epoch 48: Train err: 0.233, Train loss: 0.47616930518831524 |Validation err: 0.299, Validation loss: 0.579263653606
1764
Epoch 49: Train err: 0.2345, Train loss: 0.4818397746199653 |Validation err: 0.296, Validation loss: 0.582493184134
3641
Epoch 50: Train err: 0.229, Train loss: 0.47108897897932267 |Validation err: 0.303, Validation loss: 0.591888733208
1795
Finished Training
Total time elapsed: 240.19 seconds
```





Part 5. 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 = LargeNet()
model_path = get_model_name(net.name, batch_size=128, learning_rate=0.005, epoch=49)
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).

Answer:

- Validation error from part(a): 0.303
- Validation loss from part(a): 0.591

Both the validation loss and error here are better than previous results.

I'm selecting the large network because its higher capacity yields lower validation error than the small network. Setting the batch size to 128 and the learning rate to 0.005 significantly reduces overfitting. With this lower learning rate, extending training to 50 epochs further refines the model, minimizing validation loss and error.

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 [32]:
```
# If you use the `evaluate` function provided in part 0, you will need to
# set batch_size > 1
train_loader, val_loader, test_loader, classes = get_data_loader(
    target_classes=["cat", "dog"],
    batch_size=128)
```

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

```
The test classification error and loss are : 0.2855 and 0.5665808189660311
```

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 error (0.2855) is actually slightly lower than the validation error (0.303). In practice, one usually expects the test error to be a bit higher because hyperparameters are tuned to minimize validation loss—so the model has "seen" the validation set indirectly via that tuning. Here, though, the difference is small and likely just sampling noise: the particular split used for testing happened to be easier than the validation split.

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:

We reserve the test set for the very end because it has never been used for training or validation, making it a true simulation of real-world unseen data and an unbiased measure of final model performance. If we evaluate or tune on the test set during development, it effectively becomes part of the training process—leading to overfitting, where the model "remembers" the test examples rather than generalizing to new data.

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.

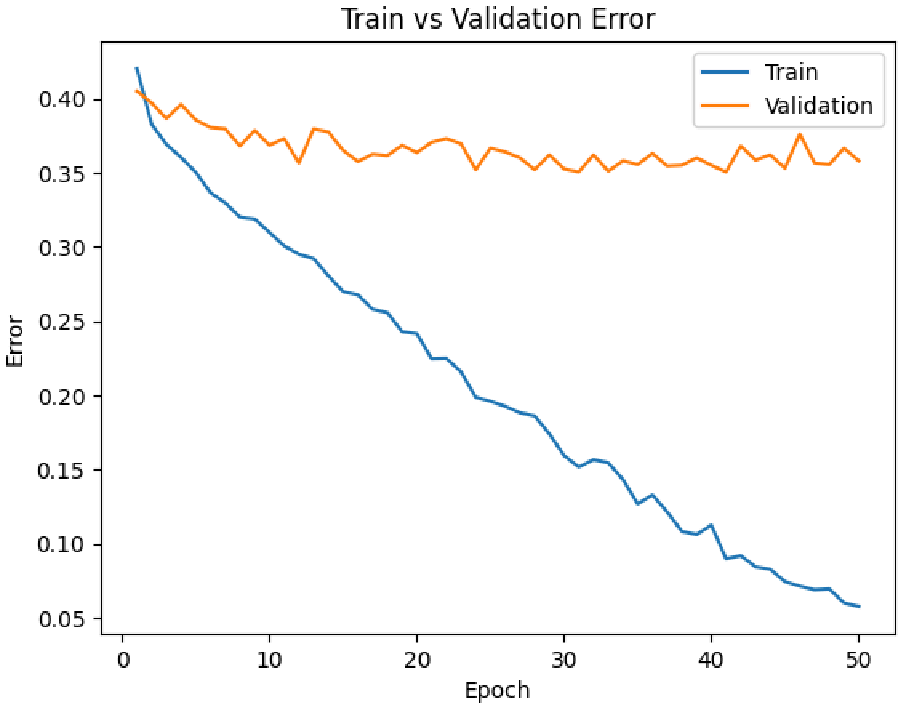
Hint: The ANN in lab 1 was applied on greyscale images. The cat and dog images are colour (RGB) and so you will need to flatted and concatinate all three colour layers before feeding them into an ANN.

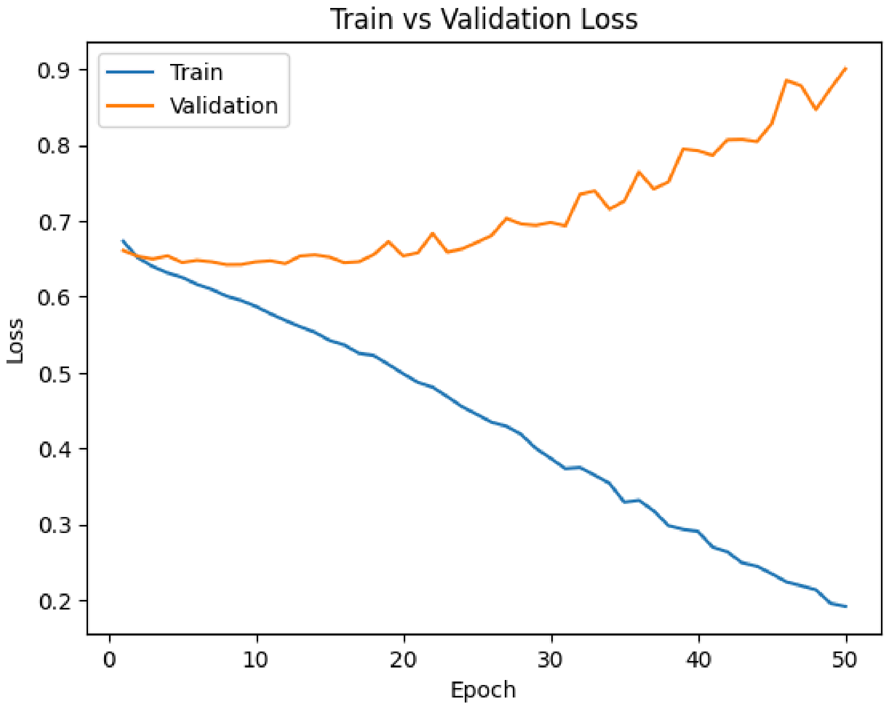
In [33]:

```
torch.manual_seed(1) #set random seed
class CatAndDog(nn.Module): # 2-layer artificial neural network
    def __init__(self):
        self.name = "CatAndDog"
        super(CatAndDog, self).__init__()
        self.layer1 = nn.Linear(3 * 32 * 32, 60)
        self.layer2 = nn.Linear(60, 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

train_ANN = CatAndDog()
train_net(train_ANN, batch_size=128, learning_rate=0.005, num_epochs=50)

cdpath = get_model_name("CatAndDog", batch_size=128, learning_rate=0.005, epoch=49)
plot_training_curve(cdpath)
```

```
Epoch 1: Train err: 0.420125, Train loss: 0.673239356941647 |Validation err: 0.405, Validation loss: 0.661061722785
2345
Epoch 2: Train err: 0.382625, Train loss: 0.651417494766296 |Validation err: 0.397, Validation loss: 0.653374370187
521
Epoch 3: Train err: 0.369125, Train loss: 0.6399649864151364 |Validation err: 0.3865, Validation loss: 0.6498257815
83786
Epoch 4: Train err: 0.36025, Train loss: 0.6315847427125961 |Validation err: 0.396, Validation loss: 0.653963312506
6757
Epoch 5: Train err: 0.35025, Train loss: 0.6255956140775529 |Validation err: 0.3855, Validation loss: 0.64519455283
88023
Epoch 6: Train err: 0.3365, Train loss: 0.6164521328986637 |Validation err: 0.3805, Validation loss: 0.648137576878
0708
Epoch 7: Train err: 0.329625, Train loss: 0.6098158993418255 |Validation err: 0.3795, Validation loss: 0.6461284793
913364
Epoch 8: Train err: 0.32, Train loss: 0.6008976754688081 |Validation err: 0.368, Validation loss: 0.642281174659729
Epoch 9: Train err: 0.31875, Train loss: 0.5951670493398394 |Validation err: 0.3785, Validation loss: 0.64245909824
96738
Epoch 10: Train err: 0.30975, Train loss: 0.587331097277384 |Validation err: 0.3685, Validation loss: 0.64607052877
54536
Epoch 11: Train err: 0.300625, Train loss: 0.5775917257581439 |Validation err: 0.373, Validation loss: 0.6475792638
9575
Epoch 12: Train err: 0.295, Train loss: 0.5687684615453085 |Validation err: 0.3565, Validation loss: 0.643708620220
4227
Epoch 13: Train err: 0.292125, Train loss: 0.5605954963063436 |Validation err: 0.3795, Validation loss: 0.653732180
595398
Epoch 14: Train err: 0.2805, Train loss: 0.5532135897212558 |Validation err: 0.3775, Validation loss: 0.65532146021
72375
Epoch 15: Train err: 0.26975, Train loss: 0.5424539191382272 |Validation err: 0.365, Validation loss: 0.65238736569
88144
Epoch 16: Train err: 0.267625, Train loss: 0.5364074565115429 |Validation err: 0.3575, Validation loss: 0.644939459
8603249
Epoch 17: Train err: 0.257875, Train loss: 0.5254545476701524 |Validation err: 0.3625, Validation loss: 0.646231323
4806061
Epoch 18: Train err: 0.255625, Train loss: 0.5225144790278541 |Validation err: 0.3615, Validation loss: 0.655661832
5412273
Epoch 19: Train err: 0.24275, Train loss: 0.5108178385666439 |Validation err: 0.3685, Validation loss: 0.6728108972
31102
Epoch 20: Train err: 0.2415, Train loss: 0.49850498636563617 |Validation err: 0.3635, Validation loss: 0.6538236998
021603
Epoch 21: Train err: 0.224625, Train loss: 0.48706356353229946 |Validation err: 0.3705, Validation loss: 0.65819843
11342239
Epoch 22: Train err: 0.224875, Train loss: 0.4807430718626295 |Validation err: 0.373, Validation loss: 0.6835006698
966026
Epoch 23: Train err: 0.215875, Train loss: 0.46818161625710747 |Validation err: 0.3695, Validation loss: 0.65887328
60982418
Epoch 24: Train err: 0.198625, Train loss: 0.45491003138678415 |Validation err: 0.352, Validation loss: 0.663257122
0397949
Epoch 25: Train err: 0.196, Train loss: 0.4448101203592997 |Validation err: 0.3665, Validation loss: 0.671755049377
6798
Epoch 26: Train err: 0.192625, Train loss: 0.4344807347607991 |Validation err: 0.364, Validation loss: 0.6809879615
902901
Epoch 27: Train err: 0.18825, Train loss: 0.42924191270555767 |Validation err: 0.36, Validation loss: 0.70343190431
59485
Epoch 28: Train err: 0.186125, Train loss: 0.41850896487160333 |Validation err: 0.352, Validation loss: 0.696032881
7367554
Epoch 29: Train err: 0.174, Train loss: 0.4000347199894133 |Validation err: 0.362, Validation loss: 0.6942745745182
037
Epoch 30: Train err: 0.159375, Train loss: 0.38698555363549125 |Validation err: 0.3525, Validation loss: 0.69802417
23358631
Epoch 31: Train err: 0.151625, Train loss: 0.3733342725133139 |Validation err: 0.3505, Validation loss: 0.693617589
7717476
Epoch 32: Train err: 0.156625, Train loss: 0.374767557969169 |Validation err: 0.362, Validation loss: 0.73524991422
89162
Epoch 33: Train err: 0.1545, Train loss: 0.3645764111526429 |Validation err: 0.351, Validation loss: 0.739649277180
4333
Epoch 34: Train err: 0.14325, Train loss: 0.3538515600893233 |Validation err: 0.358, Validation loss: 0.71553310379
38595
Epoch 35: Train err: 0.12675, Train loss: 0.329066820087887 |Validation err: 0.3555, Validation loss: 0.72628996521
23451
Epoch 36: Train err: 0.132875, Train loss: 0.3315145470320232 |Validation err: 0.363, Validation loss: 0.7645671069
62204
Epoch 37: Train err: 0.121375, Train loss: 0.3176128679797763 |Validation err: 0.3545, Validation loss: 0.742336049
6759415
Epoch 38: Train err: 0.10825, Train loss: 0.2981732565732229 |Validation err: 0.355, Validation loss: 0.75149273499
84646
Epoch 39: Train err: 0.106125, Train loss: 0.29321087636644877 |Validation err: 0.36, Validation loss: 0.7950831502
67601
```

```
Epoch 40: Train err: 0.1125, Train loss: 0.29049530294206405 |Validation err: 0.355, Validation loss: 0.79269282519
81735
Epoch 41: Train err: 0.08975, Train loss: 0.2697687998177513 |Validation err: 0.3505, Validation loss: 0.7866638563
57336
Epoch 42: Train err: 0.091875, Train loss: 0.2634454866250356 |Validation err: 0.368, Validation loss: 0.8070601522
922516
Epoch 43: Train err: 0.084375, Train loss: 0.24909782291404783 |Validation err: 0.3585, Validation loss: 0.80753416
56804085
Epoch 44: Train err: 0.08275, Train loss: 0.2445363224971862 |Validation err: 0.362, Validation loss: 0.80478303879
49944
Epoch 45: Train err: 0.074375, Train loss: 0.2346846525158201 |Validation err: 0.353, Validation loss: 0.8283248022
198677
Epoch 46: Train err: 0.071375, Train loss: 0.22386179202132755 |Validation err: 0.376, Validation loss: 0.885537423
1934547
Epoch 47: Train err: 0.068875, Train loss: 0.21881483873677632 |Validation err: 0.3565, Validation loss: 0.87848749
01175499
Epoch 48: Train err: 0.0695, Train loss: 0.2134316246660929 |Validation err: 0.3555, Validation loss: 0.84684133157
13406
Epoch 49: Train err: 0.06, Train loss: 0.19554019305441114 |Validation err: 0.3665, Validation loss: 0.874863587319
8509
Epoch 50: Train err: 0.057625, Train loss: 0.19158580000438388 |Validation err: 0.358, Validation loss: 0.900794167
0715809
Finished Training
Total time elapsed: 215.55 seconds
```





In [34]:

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

error, loss = evaluate(train_ANN, test_loader, nn.BCEWithLogitsLoss())
print("Test error is: ", error, ", and test loss is: ", loss)

error1, loss1 = evaluate(train_ANN, val_loader, nn.BCEWithLogitsLoss())
print("Validation error: ", error1, ", and validation loss: ", loss1)
```

```
Test error is:  0.3565 , and test loss is:  0.9038791991770267
Validation error:  0.358 , and validation loss:  0.9007261469960213
```

Answer:

The ANN model shows higher test error and loss, so the CNN model performs better.