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/1OS9WLVhAHEmDyzb6JzevXgiTTJ60sGtM?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
```

```
Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to ./data/cifar-10-python.tar.gz
100%|██████████| 170498071/170498071 [00:05<00:00, 33585071.19it/s]
Extracting ./data/cifar-10-python.tar.gz to ./data
Files already downloaded and verified
```

Part (a) -- 1 pt

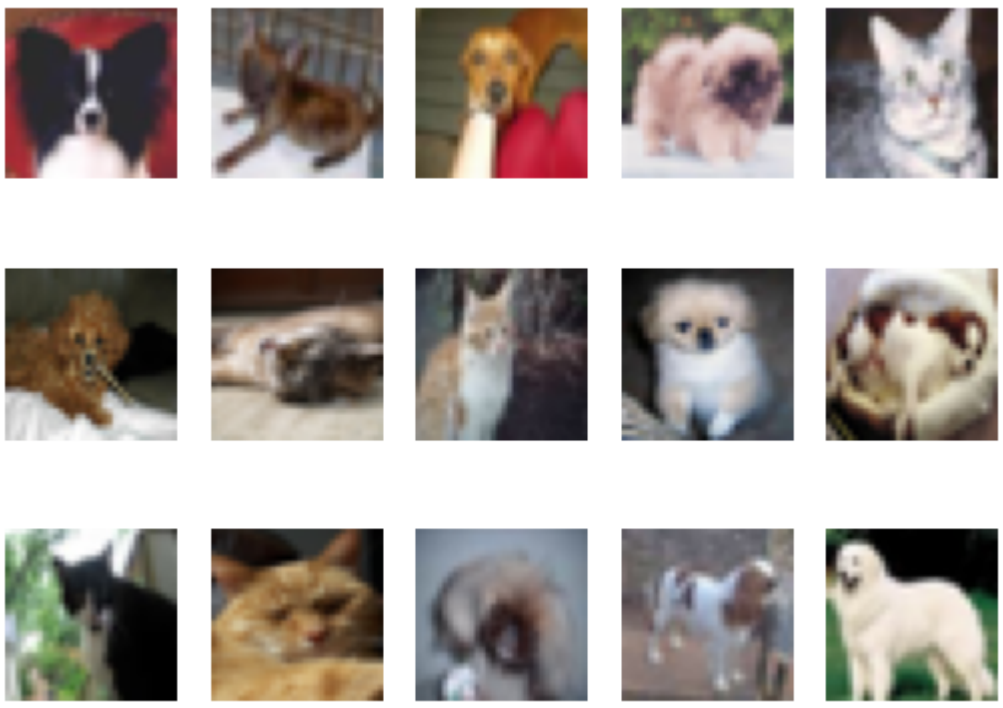
Visualize some of the data by running the code below. Include the visualization in your writeup.

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

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]:
```
distinct = set()
for images,labels in train_loader:
  distinct.add(tuple(labels.tolist()))
print(distinct)
# We only have cat and dog classes

print(f"Training Examples: {len(train_loader)}")
print(f"Validation Examples: {len(val_loader)}")
print(f"Test Examples: {len(test_loader)}")
```

```
{(3,), (5,)}
Training Examples: 8000
Validation Examples: 2000
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?

In [ ]:
```
#We need validation set because using training set to judge the performance of our models cannot indicate
#if the models generalize well to unseen data or not. If we use training set to evaluate
#the models' performance, the most complex model will always be selected,
#as it fits to all the data, including noises, giving low training error.
#However, if we use validation error, the best model to generalize to unseen data will be selected, which is what we care about.
```

In [ ]:
```

```

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]:
```
for param in small_net.parameters():
    print(param.shape)
#386 for SmallNet, including bias terms
#9705 for LargeNet, including bias terms
```

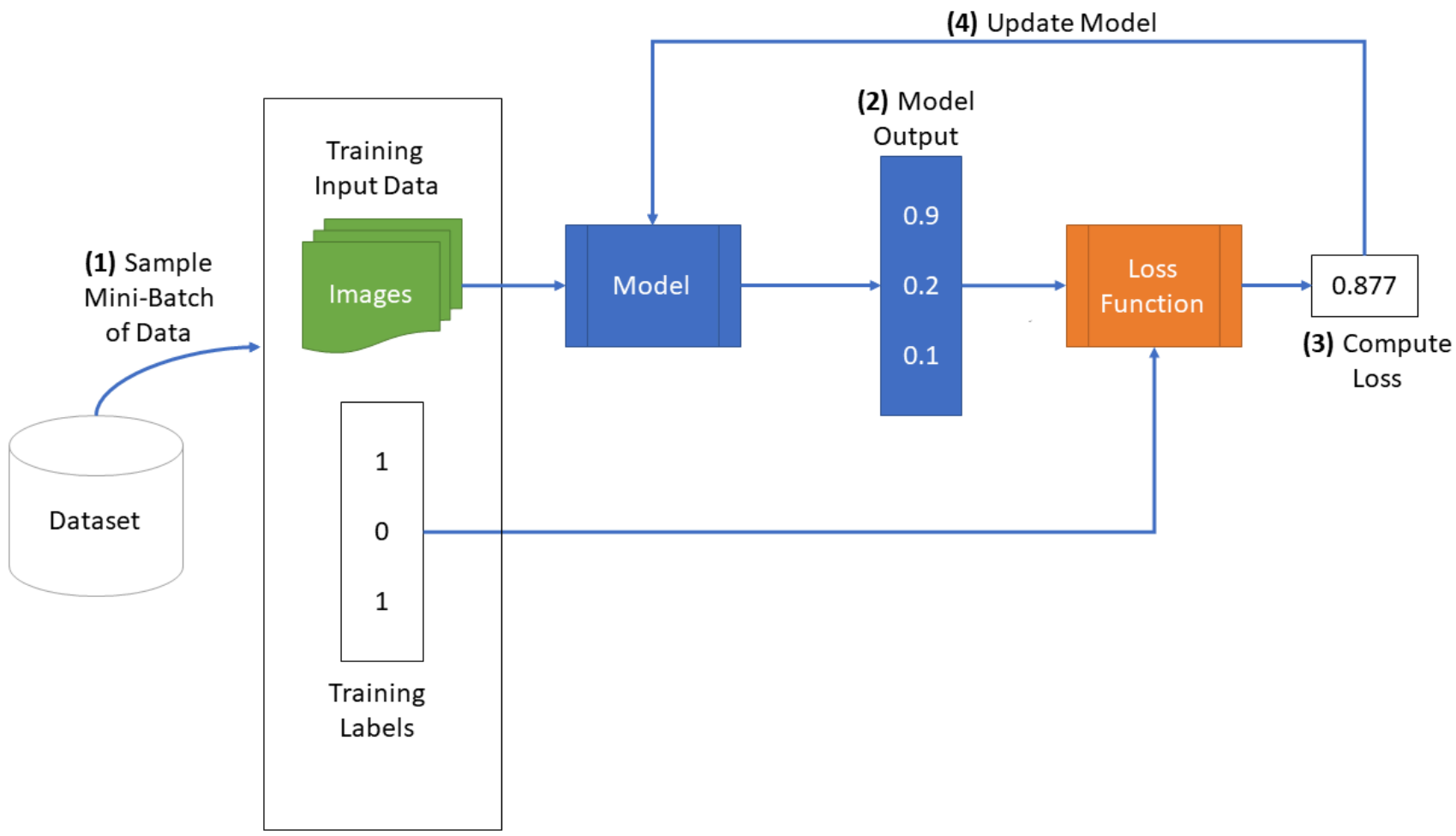
```
torch.Size([5, 3, 3, 3])
torch.Size([5])
torch.Size([1, 245])
torch.Size([1])
```

In [34]:
```
for param in large_net.parameters():
  print(param.shape)
```

```
torch.Size([5, 3, 5, 5])
torch.Size([5])
torch.Size([10, 5, 5, 5])
torch.Size([10])
torch.Size([32, 250])
torch.Size([32])
torch.Size([1, 32])
torch.Size([1])
```

The function train_net

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



In [9]:
```
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 [ ]:
```
#64, 0.01, 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 [11]:
```
#train_net(small_net, num_epochs=5)
#model_small_bs64_lr0.01_epoch0 => model's parameters for the first epoch
#model_small_bs64_lr0.01_epoch1 => model's parameters for the second epoch
#model_small_bs64_lr0.01_epoch2 => model's parameters for the third epoch
```

```
#model_small_bs64_lr0.01_epoch3 => model's parameters for the forth epoch
#model_small_bs64_lr0.01_epoch4 => model's parameters for the fifth epoch
#model_small_bs64_lr0.01_epoch4_train_err => mean training error for every epoch, total incorrect prediction/ total prediction,
#calculated while training in each epoch
#model_small_bs64_lr0.01_epoch4_train_loss => mean binary cross-entropy (BCE) error for every epoch based on the training data,
#calculated while training in each epoch
#model_small_bs64_lr0.01_epoch4_val_err => mean validation error for every epoch, total incorrect prediction / total prediction,
#calculated after training of each epoch
#model_small_bs64_lr0.01_epoch4_val_loss => mean binary cross-entropy (BCE) error for every epoch based on the validation data,
#calculated after training of each epoch
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.414375, Train loss: 0.6717066459655762 |Validation err: 0.382, Validation loss: 0.6557652298361063
Epoch 2: Train err: 0.3635, Train loss: 0.6459677453041077 |Validation err: 0.3835, Validation loss: 0.6642969120293856
Epoch 3: Train err: 0.349125, Train loss: 0.6306136822700501 |Validation err: 0.34, Validation loss: 0.6221622209995985
Epoch 4: Train err: 0.331125, Train loss: 0.6123361911773681 |Validation err: 0.3445, Validation loss: 0.6182842385023832
Epoch 5: Train err: 0.32025, Train loss: 0.6032502620220185 |Validation err: 0.325, Validation loss: 0.61301582865417
Finished Training
Total time elapsed: 25.60 seconds
```

Part (d) -- 2pt

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

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

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

In [10]:
```
# 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 [11]:
```
train_net(small_net)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.427375, Train loss: 0.6774106206893921 |Validation err: 0.378, Validation loss: 0.6593555528670549
Epoch 2: Train err: 0.365125, Train loss: 0.6447394533157349 |Validation err: 0.367, Validation loss: 0.6504110544919968
Epoch 3: Train err: 0.349625, Train loss: 0.6308710169792175 |Validation err: 0.3425, Validation loss: 0.6241882015019655
Epoch 4: Train err: 0.338, Train loss: 0.6157628531455994 |Validation err: 0.3485, Validation loss: 0.6269324030727148
Epoch 5: Train err: 0.3285, Train loss: 0.6056356673240662 |Validation err: 0.3355, Validation loss: 0.6112872939556837
Epoch 6: Train err: 0.32075, Train loss: 0.5942276575565338 |Validation err: 0.3275, Validation loss: 0.6088105775415897
Epoch 7: Train err: 0.313375, Train loss: 0.5881726610660553 |Validation err: 0.3225, Validation loss: 0.6008554734289646
Epoch 8: Train err: 0.30175, Train loss: 0.5815456295013428 |Validation err: 0.3175, Validation loss: 0.5953264180570841
Epoch 9: Train err: 0.298625, Train loss: 0.5801559321880341 |Validation err: 0.313, Validation loss: 0.5949022714048624
Epoch 10: Train err: 0.29825, Train loss: 0.5729811382293701 |Validation err: 0.3125, Validation loss: 0.589093117043376
Epoch 11: Train err: 0.298875, Train loss: 0.5717778115272522 |Validation err: 0.2995, Validation loss: 0.5863559003919363
Epoch 12: Train err: 0.296875, Train loss: 0.5674805669784546 |Validation err: 0.32, Validation loss: 0.600241026841104
Epoch 13: Train err: 0.296625, Train loss: 0.5697256689071655 |Validation err: 0.31, Validation loss: 0.5929896887391806
Epoch 14: Train err: 0.29325, Train loss: 0.5647447674274445 |Validation err: 0.309, Validation loss: 0.5954217817634344
Epoch 15: Train err: 0.289375, Train loss: 0.5632237441539765 |Validation err: 0.3055, Validation loss: 0.5887885391712189
Epoch 16: Train err: 0.295125, Train loss: 0.5686455323696137 |Validation err: 0.3105, Validation loss: 0.599271391518414
Epoch 17: Train err: 0.288125, Train loss: 0.5643566098213196 |Validation err: 0.305, Validation loss: 0.5869869738817215
Epoch 18: Train err: 0.289125, Train loss: 0.5609148676395417 |Validation err: 0.3065, Validation loss: 0.5823023468255997
Epoch 19: Train err: 0.287, Train loss: 0.5569004604816437 |Validation err: 0.313, Validation loss: 0.6001858096569777
Epoch 20: Train err: 0.280125, Train loss: 0.5569331881999969 |Validation err: 0.306, Validation loss: 0.5925471279770136
Epoch 21: Train err: 0.287875, Train loss: 0.5594933335781097 |Validation err: 0.302, Validation loss: 0.579504206776619
Epoch 22: Train err: 0.288125, Train loss: 0.5575349633693695 |Validation err: 0.318, Validation loss: 0.593852823600173
Epoch 23: Train err: 0.28575, Train loss: 0.5563359546661377 |Validation err: 0.3005, Validation loss: 0.5779231283813715
Epoch 24: Train err: 0.279, Train loss: 0.5544918267726898 |Validation err: 0.3035, Validation loss: 0.5843276819214225
Epoch 25: Train err: 0.280625, Train loss: 0.5514677784442902 |Validation err: 0.299, Validation loss: 0.5839014071971178
Epoch 26: Train err: 0.27875, Train loss: 0.5509241244792938 |Validation err: 0.309, Validation loss: 0.5870394576340914
Epoch 27: Train err: 0.2815, Train loss: 0.5501162664890289 |Validation err: 0.3075, Validation loss: 0.5933564007282257
Epoch 28: Train err: 0.278875, Train loss: 0.5514844727516174 |Validation err: 0.3065, Validation loss: 0.5799137223511934
Epoch 29: Train err: 0.283625, Train loss: 0.5503289632797241 |Validation err: 0.31, Validation loss: 0.5904530016705394
Epoch 30: Train err: 0.281875, Train loss: 0.5498797154426575 |Validation err: 0.302, Validation loss: 0.5826800689101219
Finished Training
Total time elapsed: 158.69 seconds
```

In [12]:
```
train_net(large_net)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.474, Train loss: 0.6902655873298645 |Validation err: 0.454, Validation loss: 0.6862887628376484
Epoch 2: Train err: 0.4415, Train loss: 0.6827721524238587 |Validation err: 0.4345, Validation loss: 0.6911793779581785
Epoch 3: Train err: 0.416625, Train loss: 0.6732163019180298 |Validation err: 0.3815, Validation loss: 0.6616418343037367
Epoch 4: Train err: 0.380125, Train loss: 0.6556994457244874 |Validation err: 0.3645, Validation loss: 0.6509934235364199
Epoch 5: Train err: 0.362, Train loss: 0.6422031593322753 |Validation err: 0.356, Validation loss: 0.6344756800681353
Epoch 6: Train err: 0.34525, Train loss: 0.6208089771270752 |Validation err: 0.366, Validation loss: 0.6365893315523863
Epoch 7: Train err: 0.33025, Train loss: 0.6068008756637573 |Validation err: 0.3345, Validation loss: 0.6080429144203663
Epoch 8: Train err: 0.315375, Train loss: 0.5866262481212616 |Validation err: 0.318, Validation loss: 0.59233065135777
Epoch 9: Train err: 0.30575, Train loss: 0.5776429138183594 |Validation err: 0.3145, Validation loss: 0.5853198682889342
Epoch 10: Train err: 0.29175, Train loss: 0.5593229761123657 |Validation err: 0.3005, Validation loss: 0.582740468904376
Epoch 11: Train err: 0.282625, Train loss: 0.5500850903987885 |Validation err: 0.3135, Validation loss: 0.5915080513805151
Epoch 12: Train err: 0.27425, Train loss: 0.5324860475063324 |Validation err: 0.2945, Validation loss: 0.5726518584415317
Epoch 13: Train err: 0.265125, Train loss: 0.5227808911800385 |Validation err: 0.295, Validation loss: 0.5772928539663553
Epoch 14: Train err: 0.26125, Train loss: 0.5114823932647705 |Validation err: 0.287, Validation loss: 0.5789988692849874
Epoch 15: Train err: 0.2535, Train loss: 0.5068715319633484 |Validation err: 0.296, Validation loss: 0.5721825398504734
Epoch 16: Train err: 0.2505, Train loss: 0.5031460864543915 |Validation err: 0.296, Validation loss: 0.5720396535471082
Epoch 17: Train err: 0.24025, Train loss: 0.49094378733634947 |Validation err: 0.293, Validation loss: 0.5741251911967993
Epoch 18: Train err: 0.231375, Train loss: 0.4702023732662201 |Validation err: 0.298, Validation loss: 0.5871267896145582
Epoch 19: Train err: 0.227875, Train loss: 0.4647707657814026 |Validation err: 0.3005, Validation loss: 0.5962184630334377
Epoch 20: Train err: 0.22325, Train loss: 0.4538490693569183 |Validation err: 0.314, Validation loss: 0.6699163364246488
Epoch 21: Train err: 0.218875, Train loss: 0.45116176986694334 |Validation err: 0.293, Validation loss: 0.6219055037945509
Epoch 22: Train err: 0.20875, Train loss: 0.4396454017162323 |Validation err: 0.288, Validation loss: 0.5893153119832277
Epoch 23: Train err: 0.197, Train loss: 0.4220799179077148 |Validation err: 0.2895, Validation loss: 0.6079546883702278
Epoch 24: Train err: 0.19525, Train loss: 0.41562070679664614 |Validation err: 0.301, Validation loss: 0.6117397686466575
Epoch 25: Train err: 0.184, Train loss: 0.4009286606311798 |Validation err: 0.2985, Validation loss: 0.6233449960127473
Epoch 26: Train err: 0.177, Train loss: 0.3835159775018692 |Validation err: 0.2925, Validation loss: 0.6556866690516472
Epoch 27: Train err: 0.16975, Train loss: 0.37428823590278626 |Validation err: 0.3055, Validation loss: 0.6685116458684206
Epoch 28: Train err: 0.158375, Train loss: 0.361355631351471 |Validation err: 0.2965, Validation loss: 0.6604951163753867
Epoch 29: Train err: 0.154125, Train loss: 0.33970168483257296 |Validation err: 0.304, Validation loss: 0.7254619747400284
Epoch 30: Train err: 0.153125, Train loss: 0.3333042441606522 |Validation err: 0.308, Validation loss: 0.6744236890226603
Finished Training
Total time elapsed: 181.89 seconds
```

Large_net took longer to train because the architecture of neural network is Larger, meaning more weights and biases. This results in more computational effort in doing forward and backpropagation

Part (e) - 2pt

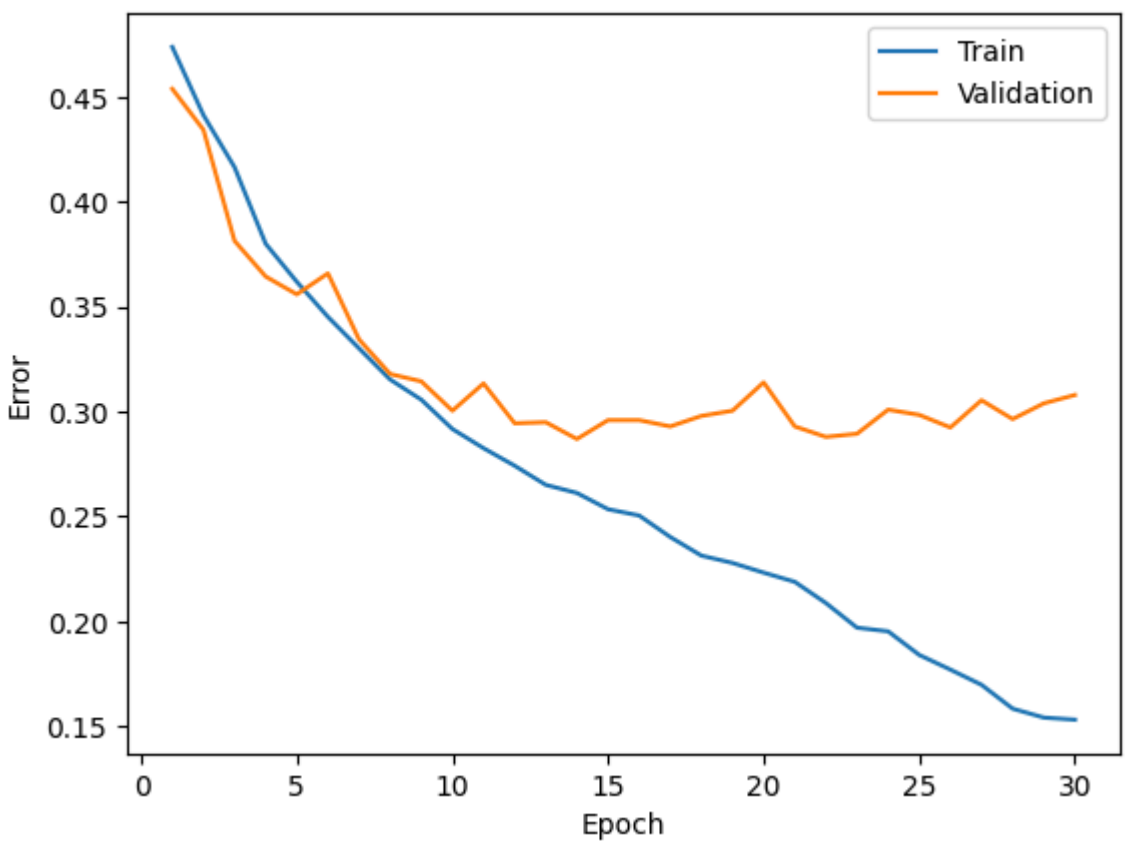
Use the function `plot_training_curve` to display the trajectory of the training/validation error and the training/validation loss. You will need to use the function `get_model_name` to generate the argument to the `plot_training_curve` function.

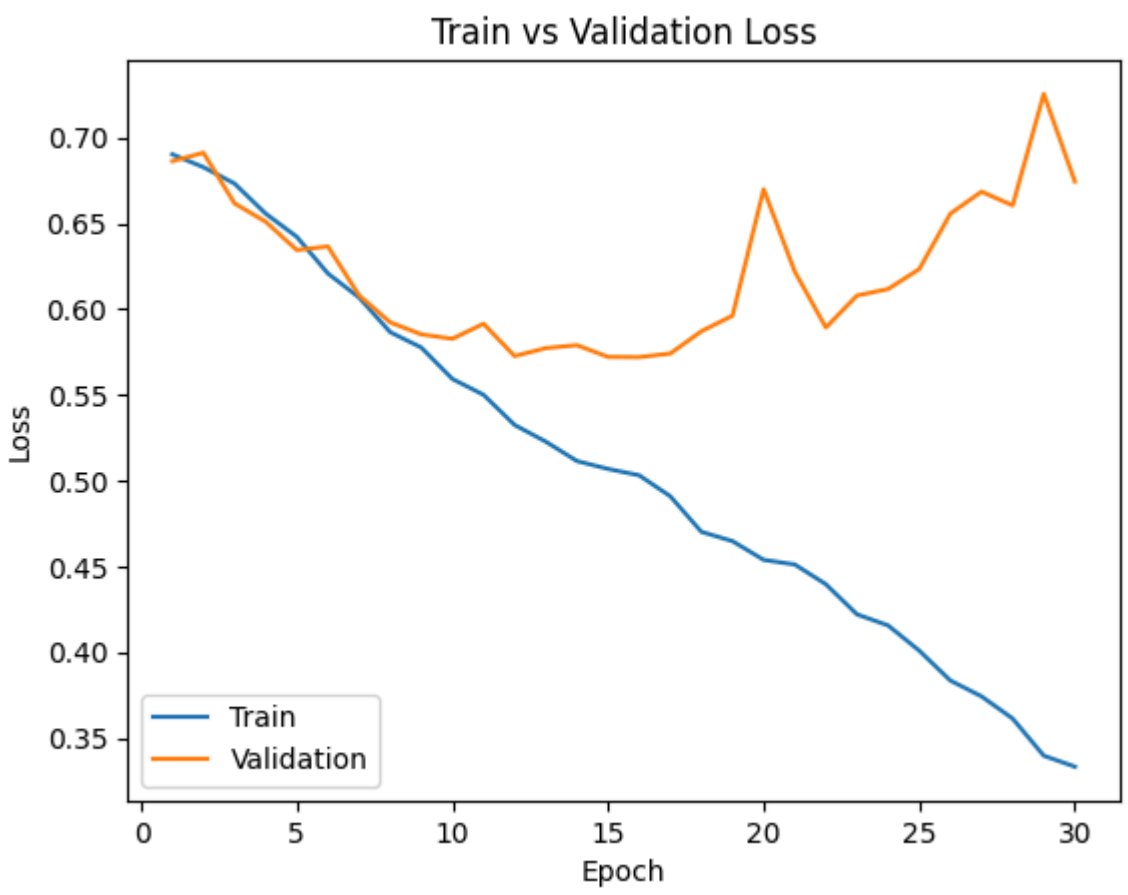
Do this for both the small network and the large network. Include both plots in your writeup.

In [14]:
```
#model_path = get_model_name("small", batch_size=??, 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")
plot_training_curve(small_path)
print("====================================")
print("Large")
plot_training_curve(large_path)
print("====================================")
```

```
Small
```

==========
Large





==========

Part (f) - 5pt

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

In [ ]:
```
#The training curve tends to decrease smoothly, while the validation curve decreases with lots of spikes.
#The train curve for small Net decrease exponentially,
#while large Net decrease almost linearly. The best performance (Lowest Validation error/ loss)
#is about the same for both models; however, Large Net
#got to the lowest validation point much faster; after that, it starts increasing indicating overfit seen
#from Large Net graphs at around epoch 15.
#Underfitting occured at the start of the graph, as the models still have insufficient training.
```

Part 3. Optimization Parameters [12 pt]

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

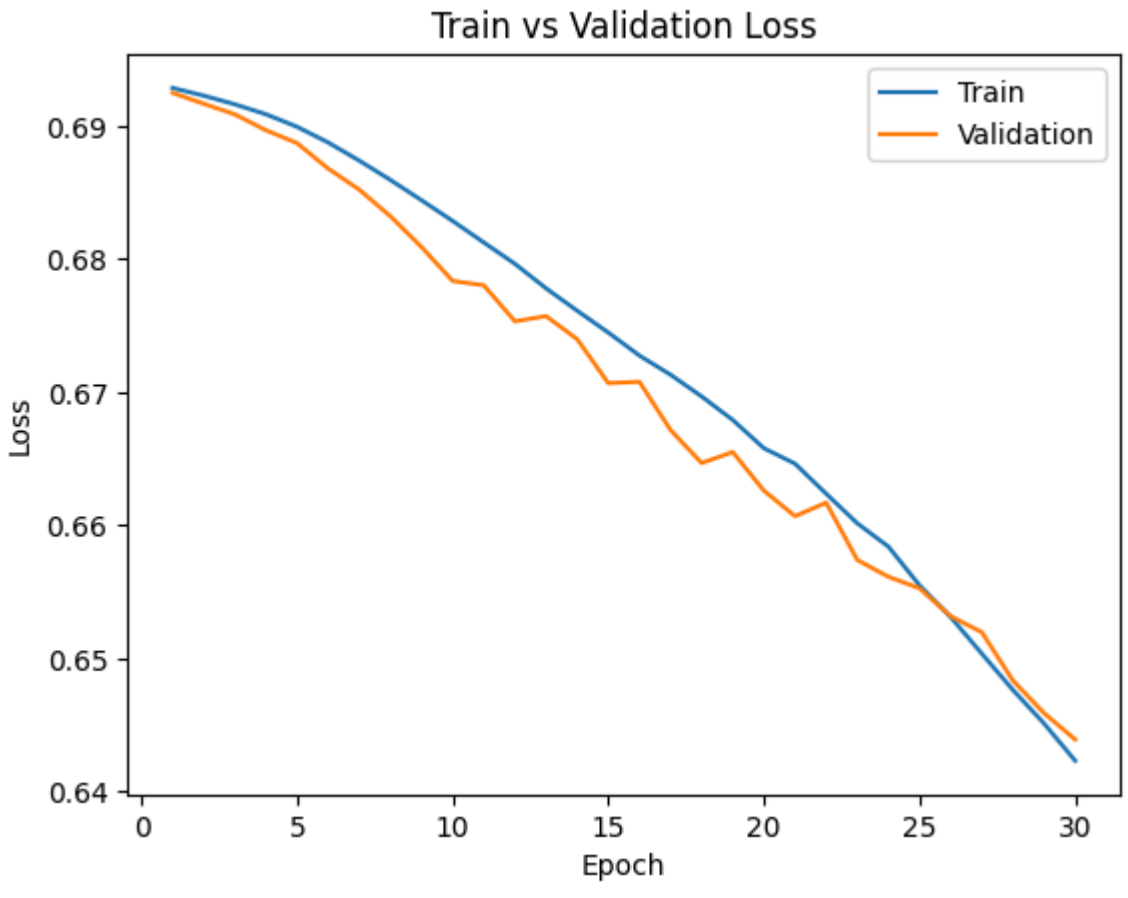
Part (a) - 3pt

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 [15]:
```
# 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)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.47625, Train loss: 0.6928360004425049 |Validation err: 0.467, Validation loss: 0.6924686580896378
Epoch 2: Train err: 0.448625, Train loss: 0.6922589740753173 |Validation err: 0.4305, Validation loss: 0.6916493494063616
Epoch 3: Train err: 0.43575, Train loss: 0.6916067256927491 |Validation err: 0.4285, Validation loss: 0.6908544301986694
Epoch 4: Train err: 0.43, Train loss: 0.6908613419532776 |Validation err: 0.424, Validation loss: 0.6896595824509859
Epoch 5: Train err: 0.434125, Train loss: 0.6899194955825806 |Validation err: 0.4195, Validation loss: 0.6886935662478209
Epoch 6: Train err: 0.43575, Train loss: 0.688741192817688 |Validation err: 0.4195, Validation loss: 0.6867824867367744
Epoch 7: Train err: 0.437125, Train loss: 0.6873774199485779 |Validation err: 0.4185, Validation loss: 0.6851983051747084
Epoch 8: Train err: 0.4375, Train loss: 0.6859278454780579 |Validation err: 0.412, Validation loss: 0.6831997763365507
Epoch 9: Train err: 0.424375, Train loss: 0.6844058051109314 |Validation err: 0.411, Validation loss: 0.6808880735188723
Epoch 10: Train err: 0.424, Train loss: 0.6828502945899964 |Validation err: 0.408, Validation loss: 0.6783502567559481
Epoch 11: Train err: 0.425375, Train loss: 0.6812348775863647 |Validation err: 0.4125, Validation loss: 0.6780214440077543
Epoch 12: Train err: 0.42, Train loss: 0.6796319665908813 |Validation err: 0.4125, Validation loss: 0.6753159128129482
Epoch 13: Train err: 0.414875, Train loss: 0.6777918725013733 |Validation err: 0.415, Validation loss: 0.6757059413939714
Epoch 14: Train err: 0.412375, Train loss: 0.6761112008094787 |Validation err: 0.412, Validation loss: 0.673973485827446
Epoch 15: Train err: 0.40925, Train loss: 0.6744726777076722 |Validation err: 0.415, Validation loss: 0.6706762481480837
Epoch 16: Train err: 0.406375, Train loss: 0.6727448830604553 |Validation err: 0.4105, Validation loss: 0.6707733031362295
Epoch 17: Train err: 0.4015, Train loss: 0.6713076605796814 |Validation err: 0.4045, Validation loss: 0.6671545337885618
Epoch 18: Train err: 0.3995, Train loss: 0.6696742882728577 |Validation err: 0.4055, Validation loss: 0.6646782532334328
Epoch 19: Train err: 0.40075, Train loss: 0.6679086318016052 |Validation err: 0.396, Validation loss: 0.6655019484460354
Epoch 20: Train err: 0.392375, Train loss: 0.6657879824638366 |Validation err: 0.405, Validation loss: 0.6626011151820421
Epoch 21: Train err: 0.38975, Train loss: 0.6646300611495972 |Validation err: 0.394, Validation loss: 0.6606878526508808
Epoch 22: Train err: 0.388875, Train loss: 0.6623730535507202 |Validation err: 0.393, Validation loss: 0.6616998631507158
Epoch 23: Train err: 0.38425, Train loss: 0.6601516304016113 |Validation err: 0.3975, Validation loss: 0.6573981866240501
Epoch 24: Train err: 0.382375, Train loss: 0.6584009370803833 |Validation err: 0.386, Validation loss: 0.6561364699155092
Epoch 25: Train err: 0.37875, Train loss: 0.6554971733093262 |Validation err: 0.388, Validation loss: 0.6552744191139936
Epoch 26: Train err: 0.376625, Train loss: 0.6531173238754272 |Validation err: 0.3875, Validation loss: 0.6531743723899126
Epoch 27: Train err: 0.375, Train loss: 0.6503696317672729 |Validation err: 0.387, Validation loss: 0.6519789230078459
Epoch 28: Train err: 0.371375, Train loss: 0.6476435804367066 |Validation err: 0.3875, Validation loss: 0.6483502611517906
Epoch 29: Train err: 0.368375, Train loss: 0.645125765323639 |Validation err: 0.3825, Validation loss: 0.6459067296236753
Epoch 30: Train err: 0.362625, Train loss: 0.6423329501152039 |Validation err: 0.3785, Validation loss: 0.6439236979931593
Finished Training
Total time elapsed: 213.70 seconds
```

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



The model takes longer to train. The Training and Validation Error/ Loss decreases as a much slower rate. From the large Net with default parameters graphs, we can see that the large Net with small learning rate still hasn't moved enough to pass the optimal point (Lowest Validation Error).

Part (b) - 3pt

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

In [17]:
```
large_net = LargeNet()
train_net(large_net, learning_rate=0.1)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.4295, Train loss: 0.6743778004646301 |Validation err: 0.3595, Validation loss: 0.6350856963545084
Epoch 2: Train err: 0.36075, Train loss: 0.6411805462837219 |Validation err: 0.3535, Validation loss: 0.6361209936439991
Epoch 3: Train err: 0.365125, Train loss: 0.6321813464164734 |Validation err: 0.3385, Validation loss: 0.6056603863835335
Epoch 4: Train err: 0.352625, Train loss: 0.623345623254776 |Validation err: 0.3575, Validation loss: 0.6362800160422921
Epoch 5: Train err: 0.34075, Train loss: 0.610801386833191 |Validation err: 0.3305, Validation loss: 0.6064918749034405
Epoch 6: Train err: 0.323375, Train loss: 0.5921835992336273 |Validation err: 0.317, Validation loss: 0.5967769687995315
Epoch 7: Train err: 0.3145, Train loss: 0.5817317562103271 |Validation err: 0.3365, Validation loss: 0.6204487904906273
Epoch 8: Train err: 0.29825, Train loss: 0.5660300071239471 |Validation err: 0.3285, Validation loss: 0.5983372181653976
Epoch 9: Train err: 0.290875, Train loss: 0.5528094999790192 |Validation err: 0.3315, Validation loss: 0.6084455195814371
Epoch 10: Train err: 0.278625, Train loss: 0.5390326056480408 |Validation err: 0.306, Validation loss: 0.5918631944805384
Epoch 11: Train err: 0.272375, Train loss: 0.5236025860309601 |Validation err: 0.33, Validation loss: 0.6430060267448425
Epoch 12: Train err: 0.267375, Train loss: 0.5220149426460267 |Validation err: 0.2925, Validation loss: 0.6413561562076211
Epoch 13: Train err: 0.266, Train loss: 0.5160510141849518 |Validation err: 0.3125, Validation loss: 0.6349832899868488
Epoch 14: Train err: 0.24875, Train loss: 0.49515900206565855 |Validation err: 0.3145, Validation loss: 0.7193072661757469
Epoch 15: Train err: 0.264625, Train loss: 0.5192319476604461 |Validation err: 0.314, Validation loss: 0.6381420735269785
Epoch 16: Train err: 0.252625, Train loss: 0.5020012385845184 |Validation err: 0.3225, Validation loss: 0.6551959468051791
Epoch 17: Train err: 0.23875, Train loss: 0.48171478748321533 |Validation err: 0.357, Validation loss: 0.6440742611885071
Epoch 18: Train err: 0.23375, Train loss: 0.4764550621509552 |Validation err: 0.3375, Validation loss: 0.6777342865243554
Epoch 19: Train err: 0.218125, Train loss: 0.45134368777275086 |Validation err: 0.3445, Validation loss: 0.7232250459492207
Epoch 20: Train err: 0.217875, Train loss: 0.45516351199150085 |Validation err: 0.3245, Validation loss: 0.6354951094835997
Epoch 21: Train err: 0.23275, Train loss: 0.47897080254554747 |Validation err: 0.3255, Validation loss: 0.8348111072555184
Epoch 22: Train err: 0.234875, Train loss: 0.4808810555934906 |Validation err: 0.334, Validation loss: 0.7191346473991871
Epoch 23: Train err: 0.21575, Train loss: 0.45636477398872377 |Validation err: 0.316, Validation loss: 0.7083508120849729
Epoch 24: Train err: 0.2355, Train loss: 0.477182511806488 |Validation err: 0.327, Validation loss: 0.7333047613501549
Epoch 25: Train err: 0.22025, Train loss: 0.45834142971038816 |Validation err: 0.3315, Validation loss: 0.7806987632066011
Epoch 26: Train err: 0.209625, Train loss: 0.4519626944065094 |Validation err: 0.3435, Validation loss: 0.7715998683124781
Epoch 27: Train err: 0.22175, Train loss: 0.4636160418987274 |Validation err: 0.3215, Validation loss: 0.7656293641775846
Epoch 28: Train err: 0.219375, Train loss: 0.4631477723121643 |Validation err: 0.348, Validation loss: 0.8202023096382618
Epoch 29: Train err: 0.235875, Train loss: 0.49053542375564574 |Validation err: 0.326, Validation loss: 0.8150459919124842
Epoch 30: Train err: 0.22, Train loss: 0.4623157210350037 |Validation err: 0.3165, Validation loss: 0.7585078477859497
Finished Training
Total time elapsed: 175.01 seconds
```

In [18]:
```
large_path = get_model_name("large", batch_size=64, learning_rate=0.1, epoch=29)
plot_training_curve(large_path)
```

It takes shorter to train this model. The validation curves have lots more spikes and it becomes overfitted quicker only at around 10th epoch.

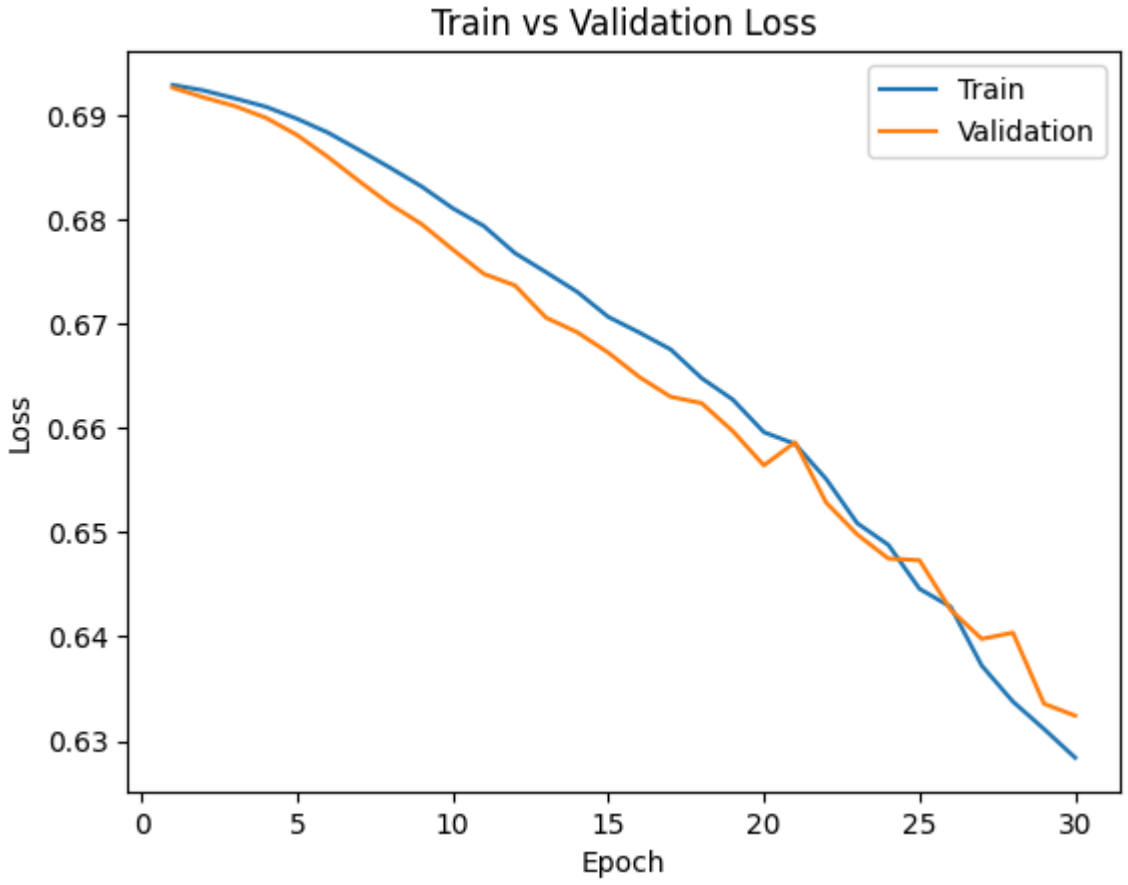
Part (c) - 3pt

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 [19]:
```
large_net = LargeNet()
train_net(large_net,batch_size=512)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.48175, Train loss: 0.6929379552602768 |Validation err: 0.478, Validation loss: 0.6926824003458023
Epoch 2: Train err: 0.457625, Train loss: 0.6924104057252407 |Validation err: 0.434, Validation loss: 0.6917425245046616
Epoch 3: Train err: 0.437, Train loss: 0.6916500627994537 |Validation err: 0.4265, Validation loss: 0.6909129917621613
Epoch 4: Train err: 0.433625, Train loss: 0.6908449903130531 |Validation err: 0.424, Validation loss: 0.6897870302200317
Epoch 5: Train err: 0.434, Train loss: 0.6896935515105724 |Validation err: 0.424, Validation loss: 0.6881355047225952
Epoch 6: Train err: 0.438, Train loss: 0.6883532106876373 |Validation err: 0.4285, Validation loss: 0.686011865735054
Epoch 7: Train err: 0.439375, Train loss: 0.6866871751844883 |Validation err: 0.426, Validation loss: 0.6836968660354614
Epoch 8: Train err: 0.43525, Train loss: 0.6849770732223988 |Validation err: 0.4115, Validation loss: 0.68146713078022
Epoch 9: Train err: 0.42375, Train loss: 0.6832009293138981 |Validation err: 0.414, Validation loss: 0.679591491818428
Epoch 10: Train err: 0.421, Train loss: 0.6811089366674423 |Validation err: 0.416, Validation loss: 0.6771548539400101
Epoch 11: Train err: 0.420875, Train loss: 0.6794026605784893 |Validation err: 0.4095, Validation loss: 0.6748111099004745
Epoch 12: Train err: 0.41475, Train loss: 0.6768048144876957 |Validation err: 0.412, Validation loss: 0.6737060546875
Epoch 13: Train err: 0.4105, Train loss: 0.6749702766537666 |Validation err: 0.412, Validation loss: 0.6706101596355438
Epoch 14: Train err: 0.407125, Train loss: 0.6730880849063396 |Validation err: 0.4125, Validation loss: 0.6692148000001907
Epoch 15: Train err: 0.4005, Train loss: 0.6706806868314743 |Validation err: 0.4105, Validation loss: 0.6672526895999908
Epoch 16: Train err: 0.397625, Train loss: 0.6691771373152733 |Validation err: 0.405, Validation loss: 0.6649097055196762
Epoch 17: Train err: 0.393875, Train loss: 0.6675694584846497 |Validation err: 0.401, Validation loss: 0.6630225032567978
Epoch 18: Train err: 0.393, Train loss: 0.6648042872548103 |Validation err: 0.3945, Validation loss: 0.6624014377593994
Epoch 19: Train err: 0.38625, Train loss: 0.6627466157078743 |Validation err: 0.388, Validation loss: 0.6597220301628113
Epoch 20: Train err: 0.38175, Train loss: 0.6596181951463223 |Validation err: 0.4005, Validation loss: 0.6564337313175201
Epoch 21: Train err: 0.38575, Train loss: 0.6584899760782719 |Validation err: 0.3885, Validation loss: 0.6586423963308334
Epoch 22: Train err: 0.378125, Train loss: 0.6551233902573586 |Validation err: 0.3855, Validation loss: 0.6528600305318832
Epoch 23: Train err: 0.372125, Train loss: 0.6508794091641903 |Validation err: 0.3835, Validation loss: 0.6497963666915894
Epoch 24: Train err: 0.37675, Train loss: 0.6488028429448605 |Validation err: 0.385, Validation loss: 0.6474899798631668
Epoch 25: Train err: 0.368625, Train loss: 0.6445869281888008 |Validation err: 0.382, Validation loss: 0.6473268419504166
Epoch 26: Train err: 0.372625, Train loss: 0.6428566128015518 |Validation err: 0.3745, Validation loss: 0.6425703316926956
Epoch 27: Train err: 0.359375, Train loss: 0.6372117511928082 |Validation err: 0.379, Validation loss: 0.6397799849510193
Epoch 28: Train err: 0.35425, Train loss: 0.6337667480111122 |Validation err: 0.3695, Validation loss: 0.6403782963752747
Epoch 29: Train err: 0.3535, Train loss: 0.6311352998018265 |Validation err: 0.366, Validation loss: 0.6335585117340088
Epoch 30: Train err: 0.353, Train loss: 0.6283832415938377 |Validation err: 0.3675, Validation loss: 0.6324127167463303
Finished Training
Total time elapsed: 151.79 seconds
```

In [20]:
```
large_path = get_model_name("large", batch_size=512, learning_rate=0.01, epoch=29)
plot_training_curve(large_path)
```



It takes shorter to train compared to the default model. The graph is more smooth; however, it converges to the optimal validation point slower.

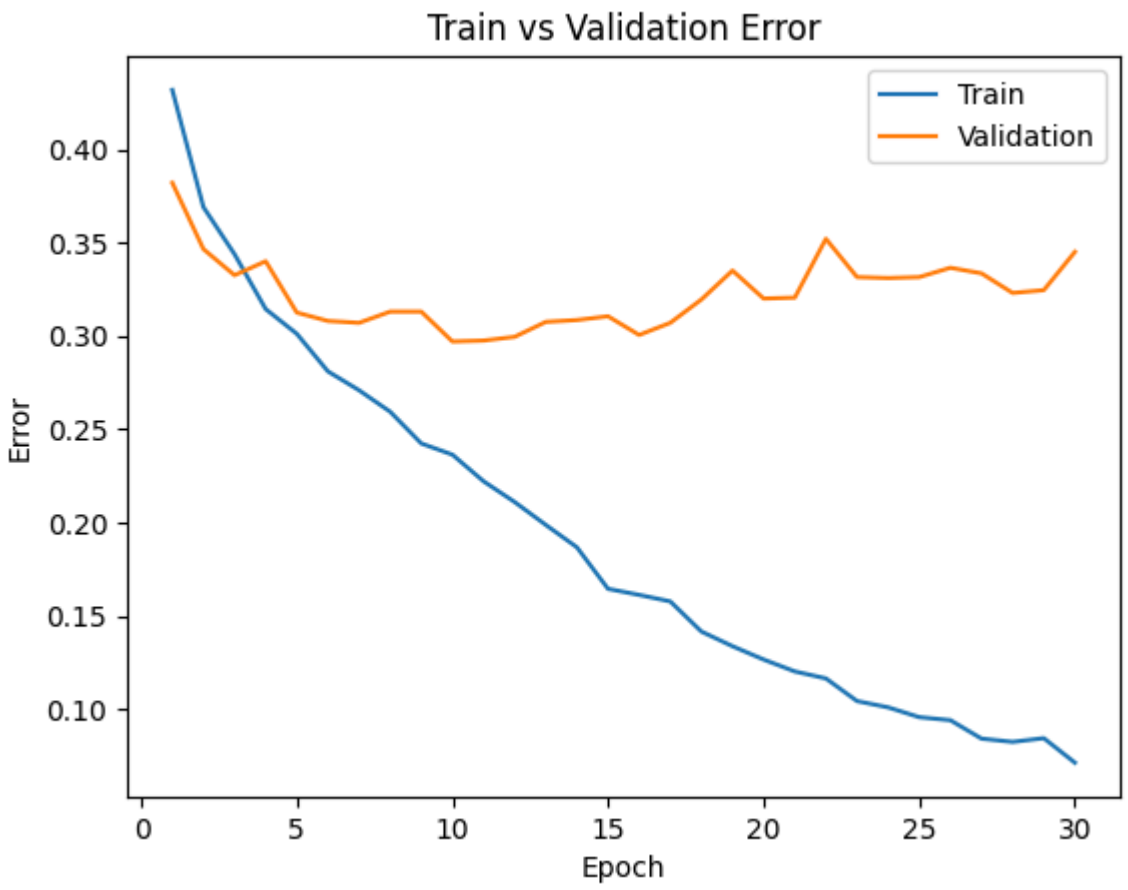
Part (d) - 3pt

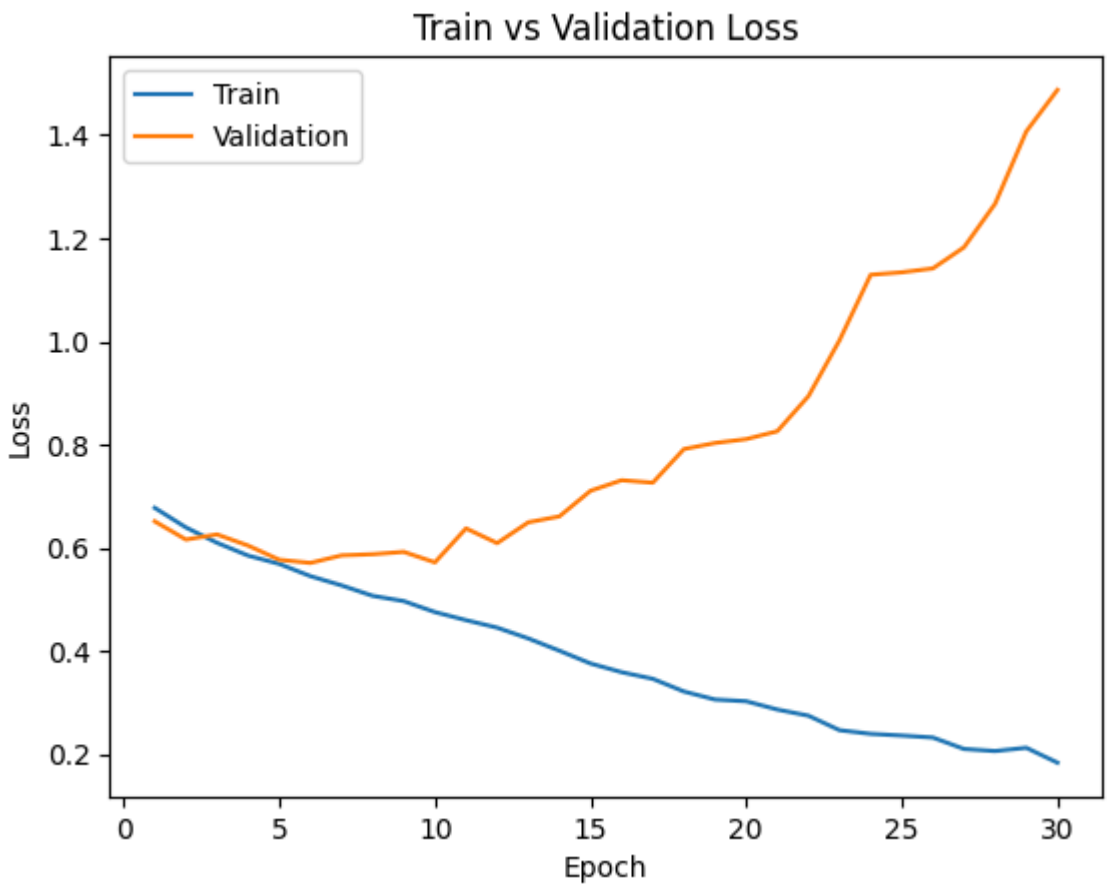
Train `large_net` with all default parameters, including with `learning_rate=0.01`. Now, set `batch_size=16`. Does the model take longer/shorter to train? Plot the training curve. Describe the effect of *decreasing* the batch size.

In [21]:
```
large_net = LargeNet()
train_net(large_net,batch_size=16)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.43175, Train loss: 0.6774994033575058 |Validation err: 0.382, Validation loss: 0.6513170146942139
Epoch 2: Train err: 0.369, Train loss: 0.6396398993134499 |Validation err: 0.3465, Validation loss: 0.6161113579273224
Epoch 3: Train err: 0.34375, Train loss: 0.6098222960829734 |Validation err: 0.3325, Validation loss: 0.6260210766792297
Epoch 4: Train err: 0.314375, Train loss: 0.584969149172306 |Validation err: 0.34, Validation loss: 0.6044013905525207
Epoch 5: Train err: 0.301125, Train loss: 0.5689119317531586 |Validation err: 0.3125, Validation loss: 0.5769183149337769
Epoch 6: Train err: 0.281, Train loss: 0.5452213580608368 |Validation err: 0.308, Validation loss: 0.570844743013382
Epoch 7: Train err: 0.270875, Train loss: 0.5272981309890747 |Validation err: 0.307, Validation loss: 0.5854293291568756
Epoch 8: Train err: 0.259375, Train loss: 0.507090549826622 |Validation err: 0.313, Validation loss: 0.5877130846977234
Epoch 9: Train err: 0.242375, Train loss: 0.49683444169163704 |Validation err: 0.313, Validation loss: 0.5922425067424775
Epoch 10: Train err: 0.236375, Train loss: 0.47561015680432317 |Validation err: 0.297, Validation loss: 0.5718690168857574
Epoch 11: Train err: 0.222125, Train loss: 0.45997694665193556 |Validation err: 0.2975, Validation loss: 0.6376970813274384
Epoch 12: Train err: 0.211, Train loss: 0.4454492364227772 |Validation err: 0.2995, Validation loss: 0.609202568769455
Epoch 13: Train err: 0.19875, Train loss: 0.42454217198491095 |Validation err: 0.3075, Validation loss: 0.6494987757205963
Epoch 14: Train err: 0.18675, Train loss: 0.4007472902536392 |Validation err: 0.3085, Validation loss: 0.6610016564130783
Epoch 15: Train err: 0.1645, Train loss: 0.3759974044710398 |Validation err: 0.3105, Validation loss: 0.7106090523004532
Epoch 16: Train err: 0.16125, Train loss: 0.35914554065465926 |Validation err: 0.3005, Validation loss: 0.7310364973545075
Epoch 17: Train err: 0.15775, Train loss: 0.3463234778419137 |Validation err: 0.307, Validation loss: 0.7263009355068207
Epoch 18: Train err: 0.141625, Train loss: 0.32175366409868 |Validation err: 0.3195, Validation loss: 0.7913952922821045
Epoch 19: Train err: 0.13375, Train loss: 0.3061810576841235 |Validation err: 0.335, Validation loss: 0.8032052783966065
Epoch 20: Train err: 0.126625, Train loss: 0.30290717820078134 |Validation err: 0.32, Validation loss: 0.8106685200929642
Epoch 21: Train err: 0.12025, Train loss: 0.28682796521484855 |Validation err: 0.3205, Validation loss: 0.8259474363327026
Epoch 22: Train err: 0.1165, Train loss: 0.2748908795714378 |Validation err: 0.352, Validation loss: 0.8937610728740693
Epoch 23: Train err: 0.104375, Train loss: 0.2467898515611887 |Validation err: 0.3315, Validation loss: 1.0021928179264068
Epoch 24: Train err: 0.101, Train loss: 0.23970085600204766 |Validation err: 0.331, Validation loss: 1.1290796512365342
Epoch 25: Train err: 0.09575, Train loss: 0.23643119525164366 |Validation err: 0.3315, Validation loss: 1.1338514356613159
Epoch 26: Train err: 0.094125, Train loss: 0.23259535063058137 |Validation err: 0.3365, Validation loss: 1.141426316022873
Epoch 27: Train err: 0.08425, Train loss: 0.21040759443677962 |Validation err: 0.3335, Validation loss: 1.182367821574211
Epoch 28: Train err: 0.0825, Train loss: 0.20643112601805477 |Validation err: 0.323, Validation loss: 1.2668361866474152
Epoch 29: Train err: 0.0845, Train loss: 0.21273409315384925 |Validation err: 0.3245, Validation loss: 1.406717713713646
Epoch 30: Train err: 0.071375, Train loss: 0.18387044004537165 |Validation err: 0.345, Validation loss: 1.4871552119255065
Finished Training
Total time elapsed: 229.95 seconds
```

In [22]:
```
large_path = get_model_name("large", batch_size=16, learning_rate=0.01, epoch=29)
plot_training_curve(large_path)
```





The model takes longer to train compared to the default model. The model converges to the optimal validation point much faster. Therefore, it overfits much faster and the training error decrease as a faster rate.

Part 4. Hyperparameter Search [6 pt]

Part (a) - 2pt

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

In [ ]:
```
#(LargeNet(), 64 ,0.005) Because learning rate of 0.01 overfits very fast,
#and learning rate of 0.001 hasn't converge to the optimal validation point.

# So, I have chosen lr = 0.005 which is between 0.01 and 0.001 with the hope to
# find optimal validation point with more precision than lr = 0.01
```

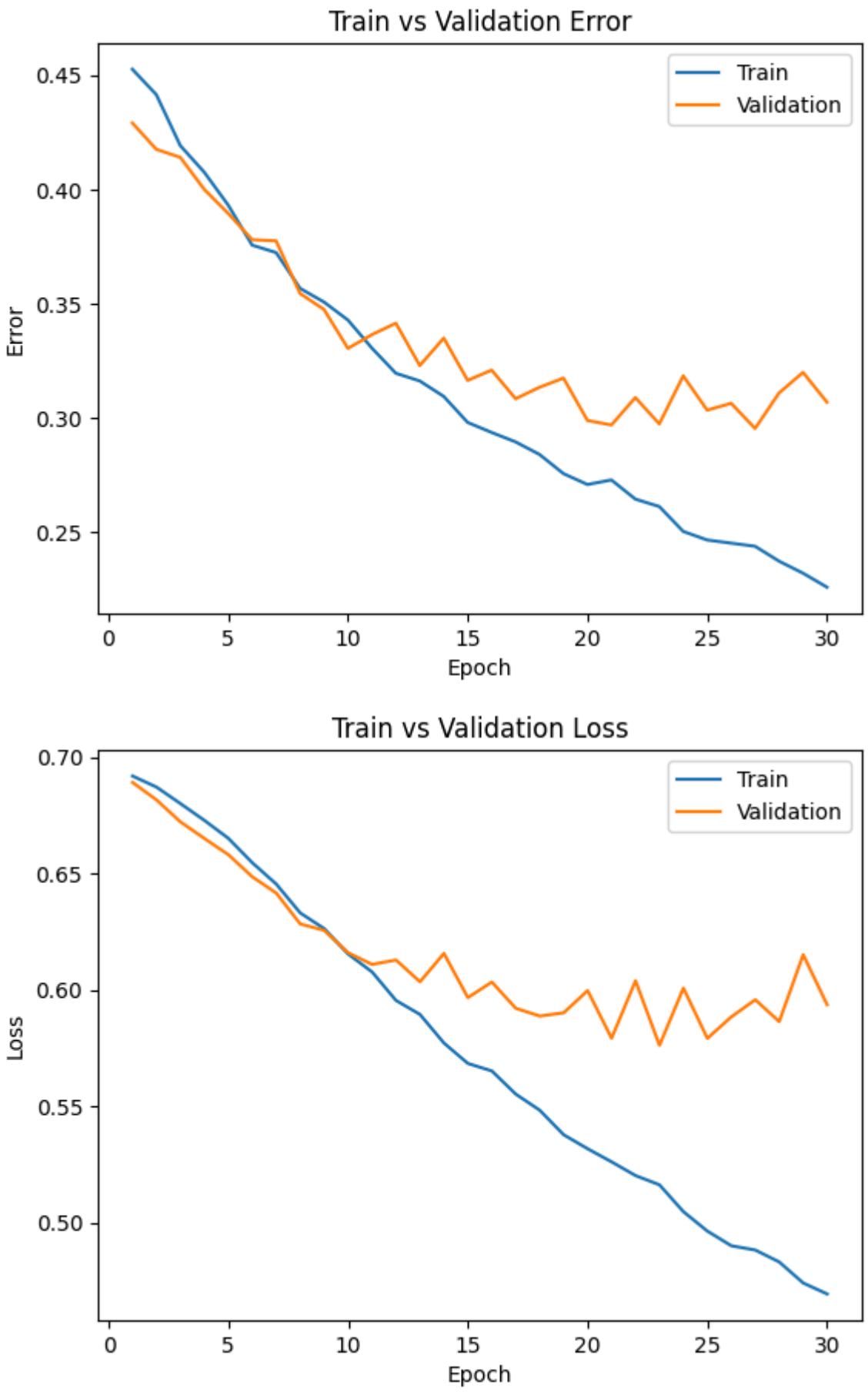
Part (b) - 1pt

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

In [23]:
```
large_net = LargeNet()
train_net(large_net,learning_rate=0.005)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.4525, Train loss: 0.6918498039245605 |Validation err: 0.429, Validation loss: 0.6890992540866137
Epoch 2: Train err: 0.441375, Train loss: 0.6870906562805176 |Validation err: 0.4175, Validation loss: 0.6815983355045319
Epoch 3: Train err: 0.419125, Train loss: 0.6800040378570557 |Validation err: 0.414, Validation loss: 0.6721472628414631
Epoch 4: Train err: 0.4075, Train loss: 0.6727880077362061 |Validation err: 0.4, Validation loss: 0.6650289222598076
Epoch 5: Train err: 0.393125, Train loss: 0.6650955910682679 |Validation err: 0.3895, Validation loss: 0.6580633856356144
Epoch 6: Train err: 0.375625, Train loss: 0.6546039323806763 |Validation err: 0.378, Validation loss: 0.6485581323504448
Epoch 7: Train err: 0.372375, Train loss: 0.6454356803894042 |Validation err: 0.3775, Validation loss: 0.6415849011391401
Epoch 8: Train err: 0.35675, Train loss: 0.6330658745765686 |Validation err: 0.3545, Validation loss: 0.6283326148986816
Epoch 9: Train err: 0.35075, Train loss: 0.6263206467628479 |Validation err: 0.3475, Validation loss: 0.6255707871168852
Epoch 10: Train err: 0.342875, Train loss: 0.6154069499969482 |Validation err: 0.3305, Validation loss: 0.6159476228058338
Epoch 11: Train err: 0.330625, Train loss: 0.607706931591034 |Validation err: 0.3365, Validation loss: 0.6109566707164049
Epoch 12: Train err: 0.319625, Train loss: 0.5954657964706421 |Validation err: 0.3415, Validation loss: 0.6128380848094821
Epoch 13: Train err: 0.31625, Train loss: 0.5894628224372864 |Validation err: 0.323, Validation loss: 0.6034849472343922
Epoch 14: Train err: 0.3095, Train loss: 0.5771885201931 |Validation err: 0.335, Validation loss: 0.6156702265143394
Epoch 15: Train err: 0.298125, Train loss: 0.5684188063144684 |Validation err: 0.3165, Validation loss: 0.5966824218630791
Epoch 16: Train err: 0.29375, Train loss: 0.565195871591568 |Validation err: 0.321, Validation loss: 0.603372392244637
Epoch 17: Train err: 0.289625, Train loss: 0.5552479808330536 |Validation err: 0.3085, Validation loss: 0.5921099754050374
Epoch 18: Train err: 0.284125, Train loss: 0.5483348236083985 |Validation err: 0.3135, Validation loss: 0.5887665553018451
Epoch 19: Train err: 0.27575, Train loss: 0.5377972214221954 |Validation err: 0.3175, Validation loss: 0.5901659782975912
Epoch 20: Train err: 0.271, Train loss: 0.5317816915512085 |Validation err: 0.299, Validation loss: 0.5997155867516994
Epoch 21: Train err: 0.273, Train loss: 0.5261548852920532 |Validation err: 0.297, Validation loss: 0.5792500302195549
Epoch 22: Train err: 0.264625, Train loss: 0.5202640609741211 |Validation err: 0.309, Validation loss: 0.6039150953292847
Epoch 23: Train err: 0.261375, Train loss: 0.5162914700508118 |Validation err: 0.2975, Validation loss: 0.5762067250907421
Epoch 24: Train err: 0.2505, Train loss: 0.5047600004673004 |Validation err: 0.3185, Validation loss: 0.6007640128955245
Epoch 25: Train err: 0.24675, Train loss: 0.4964101016521454 |Validation err: 0.3035, Validation loss: 0.5791856553405523
Epoch 26: Train err: 0.245375, Train loss: 0.490117956161499 |Validation err: 0.3065, Validation loss: 0.5884320810437202
Epoch 27: Train err: 0.244, Train loss: 0.48828510522842405 |Validation err: 0.2955, Validation loss: 0.5957792242988944
Epoch 28: Train err: 0.2375, Train loss: 0.4831665117740631 |Validation err: 0.311, Validation loss: 0.5864075161516666
Epoch 29: Train err: 0.23225, Train loss: 0.4741382575035095 |Validation err: 0.32, Validation loss: 0.6151109952479601
Epoch 30: Train err: 0.226125, Train loss: 0.4694191563129425 |Validation err: 0.307, Validation loss: 0.5936459172517061
Finished Training
Total time elapsed: 175.80 seconds
```

In [24]:
```
large_path = get_model_name("large", batch_size=64, learning_rate=0.005, epoch=29)
plot_training_curve(large_path)
```



Part (c) - 2pt

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

In [ ]:
```
#(LargeNet(), 32 ,0.005) The model still hasn't shown any sign of overfit,
#meaning it still might not have passed the optimal validation point.
#Decreasing batch size will make it converges faster.
```

Part (d) - 1pt

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

In [25]:
```
large_net = LargeNet()
train_net(large_net,learning_rate=0.005, batch_size=32)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.451125, Train loss: 0.6895513634681701 |Validation err: 0.4235, Validation loss: 0.6799672132446652
Epoch 2: Train err: 0.41475, Train loss: 0.6770808389186859 |Validation err: 0.426, Validation loss: 0.6783659694686769
Epoch 3: Train err: 0.399, Train loss: 0.6652117218971253 |Validation err: 0.399, Validation loss: 0.6516326363124545
Epoch 4: Train err: 0.3785, Train loss: 0.6488407545089722 |Validation err: 0.393, Validation loss: 0.6554539099572196
Epoch 5: Train err: 0.357, Train loss: 0.630372572183609 |Validation err: 0.342, Validation loss: 0.6204922927750481
Epoch 6: Train err: 0.339, Train loss: 0.6117297103404998 |Validation err: 0.348, Validation loss: 0.6352075660039508
Epoch 7: Train err: 0.323375, Train loss: 0.5982237352132798 |Validation err: 0.3235, Validation loss: 0.6010712626434508
Epoch 8: Train err: 0.316375, Train loss: 0.5813896110057831 |Validation err: 0.3275, Validation loss: 0.59802912199308
Epoch 9: Train err: 0.30025, Train loss: 0.5679361228942871 |Validation err: 0.3295, Validation loss: 0.5980716328772288
Epoch 10: Train err: 0.29375, Train loss: 0.5566963509321213 |Validation err: 0.313, Validation loss: 0.5937082630301279
Epoch 11: Train err: 0.2835, Train loss: 0.5451354386806488 |Validation err: 0.3055, Validation loss: 0.5893454267865136
Epoch 12: Train err: 0.275125, Train loss: 0.5368142924308776 |Validation err: 0.309, Validation loss: 0.5910099276474544
Epoch 13: Train err: 0.270625, Train loss: 0.5264033428430557 |Validation err: 0.2975, Validation loss: 0.5848427050643497
Epoch 14: Train err: 0.264625, Train loss: 0.5190897635221481 |Validation err: 0.3055, Validation loss: 0.586686182589758
Epoch 15: Train err: 0.257625, Train loss: 0.5118338980674744 |Validation err: 0.2985, Validation loss: 0.5910106112086584
Epoch 16: Train err: 0.2485, Train loss: 0.4986576300859451 |Validation err: 0.2985, Validation loss: 0.5861731890648131
Epoch 17: Train err: 0.2485, Train loss: 0.49048795491456987 |Validation err: 0.3, Validation loss: 0.5888396208248441
Epoch 18: Train err: 0.231875, Train loss: 0.47393429481983185 |Validation err: 0.2975, Validation loss: 0.5872668406319996
Epoch 19: Train err: 0.225625, Train loss: 0.4685859904885292 |Validation err: 0.309, Validation loss: 0.6007849195646862
Epoch 20: Train err: 0.221, Train loss: 0.4554621467590332 |Validation err: 0.319, Validation loss: 0.6511462061178117
Epoch 21: Train err: 0.212, Train loss: 0.44733059430122374 |Validation err: 0.3045, Validation loss: 0.6009630636563377
Epoch 22: Train err: 0.205625, Train loss: 0.43213254684209823 |Validation err: 0.297, Validation loss: 0.610547912972314
Epoch 23: Train err: 0.197125, Train loss: 0.4226604918837547 |Validation err: 0.2925, Validation loss: 0.6120050365016574
Epoch 24: Train err: 0.192875, Train loss: 0.4180468382239342 |Validation err: 0.305, Validation loss: 0.6439091104363638
Epoch 25: Train err: 0.18525, Train loss: 0.4023225308060646 |Validation err: 0.2935, Validation loss: 0.6296065355104113
Epoch 26: Train err: 0.176125, Train loss: 0.38931622326374055 |Validation err: 0.315, Validation loss: 0.6435245270766909
Epoch 27: Train err: 0.168875, Train loss: 0.3731733102202415 |Validation err: 0.298, Validation loss: 0.6803916484590561
Epoch 28: Train err: 0.16925, Train loss: 0.36944942063093184 |Validation err: 0.303, Validation loss: 0.6574611200226678
Epoch 29: Train err: 0.156125, Train loss: 0.3434563274979591 |Validation err: 0.3035, Validation loss: 0.7566462717359028
Epoch 30: Train err: 0.14425, Train loss: 0.32549949443340304 |Validation err: 0.3015, Validation loss: 0.7118366327550676
Finished Training
Total time elapsed: 198.16 seconds
```

In [26]:
```
large_path = get_model_name("large", batch_size=32, learning_rate=0.005, epoch=29)
plot_training_curve(large_path)
```

Part 4. Evaluating the Best Model [15 pt]

Part (a) - 1pt

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

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

In [27]:
```
net = LargeNet()
model_path = get_model_name(net.name, batch_size=64, learning_rate=0.01, epoch=11)
state = torch.load(model_path)
net.load_state_dict(state)
```

```
<ipython-input-27-97529e3b3902>:3: FutureWarning: You are using `torch.load` with `weights_only=False` (the current default valu
e), which uses the default pickle module implicitly. It is possible to construct malicious pickle data which will execute arbitr
ary code during unpickling (See https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for more details). In
a future release, the default value for `weights_only` will be flipped to `True`. This limits the functions that could be execut
ed during unpickling. Arbitrary objects will no longer be allowed to be loaded via this mode unless they are explicitly allowlis
ted by the user via `torch.serialization.add_safe_globals`. We recommend you start setting `weights_only=True` for any use case
where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experimental
feature.
  state = torch.load(model_path)
```

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

In [ ]:
```
#I will choose Epoch 12 from Original LargeNet()
```

Part (b) - 2pt

Justify your choice of model from part (a).

In [ ]:
```
# The lowest validation error is around 0.287
# The lowest validation loss is around 0.571
#Epoch 12: Train err: 0.27425, Train loss: 0.5324860475063324 |Validation err: 0.2945, Validation loss: 0.5726518584415317
#I have chosen this model because the validation error and loss is very close to the lowest achieved across all models.
```

Part (c) - 2pt

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

In [28]:
```
# If you use the `evaluate` function provided in part 0, you will need to
# set batch_size > 1
train_loader, val_loader, test_loader, classes = get_data_loader(
    target_classes=["cat", "dog"],
    batch_size=64)
criterion = nn.BCEWithLogitsLoss()
err, loss = evaluate(net, test_loader, criterion)
print(f"Prediction Error: {err}, BCE Error: {loss}")
```

```
Files already downloaded and verified
Files already downloaded and verified
Prediction Error: 0.277, BCE Error: 0.5437365910038352
```

Part (d) - 3pt

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

In [ ]:
```
#Note: Optimistic means the evaluation metrics indicates model performance to be
#better than what the model would actually achieve when facing unseen data.
```

In [ ]:
```
#Test classification error is higher than the validation error because
#We have used validation data in our training process; therefore, we know that the model
#we selected is the best model for the validation data, which makes validation error
#becomes more optimistic than test error which is unseen.
```

Part (e) - 2pt

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

In [ ]:
```
#The test data is used at the very end to get an unbiased performance estimate.
#If test data have been used in the model building process, the evaluation will
#be optimistic, which will not reflect the model's real performance.
```

Part (f) - 5pt

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

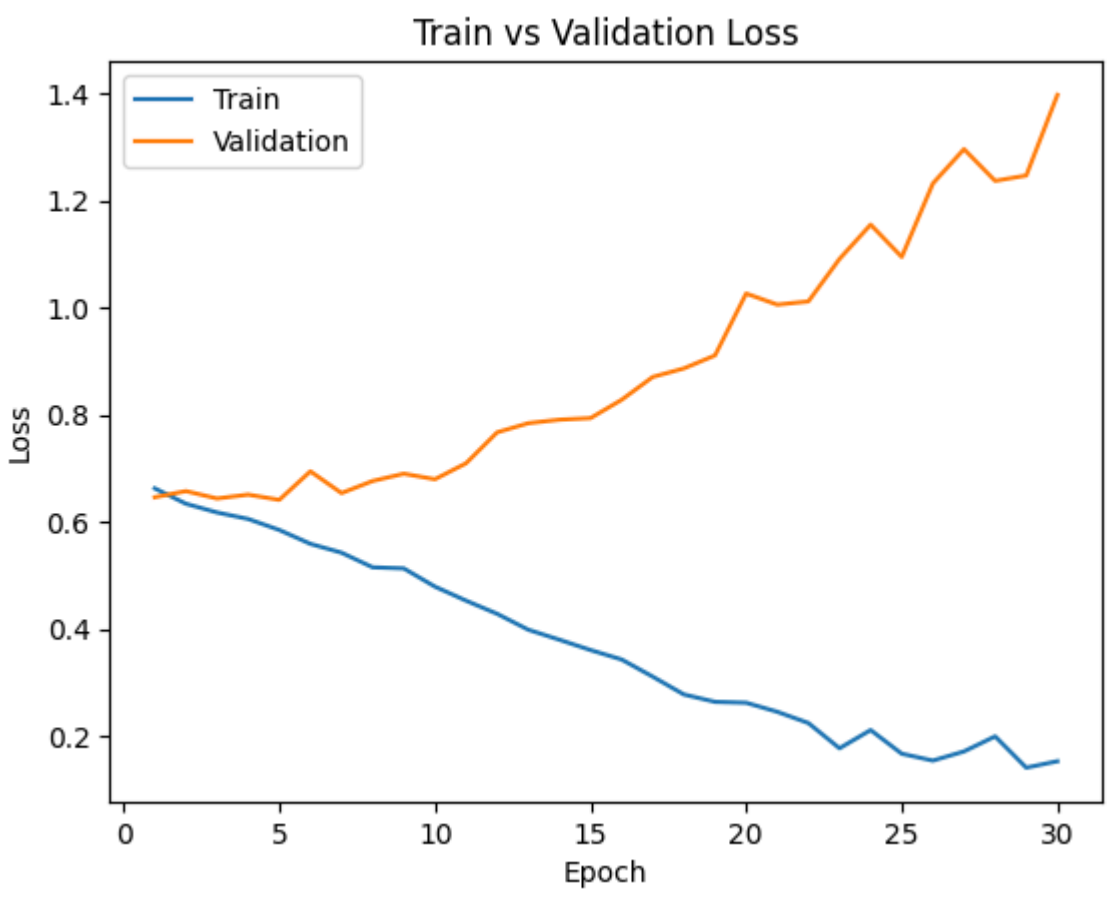
Hint: The ANN in lab 1 was applied on greyscale images. The cat and dog images are colour (RGB) and so you will need to flatted and concatinate all three colour layers before feeding them into an ANN.

In [30]:
```
class ANN(nn.Module):
    def __init__(self):
        super(ANN, self).__init__()
        self.name = "ANN"
        self.layer1 = nn.Linear(3*32*32,256)
        self.layer2 = nn.Linear(256,1)

    def forward(self, x):
        x = x.reshape(-1,3*32*32)
        x = F.relu(self.layer1(x))
        x = self.layer2(x)
        x = x.squeeze(1)
        return x
ann = ANN()
train_net(ann)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.402, Train loss: 0.6620317826271057 |Validation err: 0.3775, Validation loss: 0.6455250792205334
Epoch 2: Train err: 0.362875, Train loss: 0.6333760628700257 |Validation err: 0.3925, Validation loss: 0.6569452602416277
Epoch 3: Train err: 0.346125, Train loss: 0.6170727481842041 |Validation err: 0.379, Validation loss: 0.6431782953441143
Epoch 4: Train err: 0.331375, Train loss: 0.604934368133545 |Validation err: 0.3805, Validation loss: 0.6504221875220537
Epoch 5: Train err: 0.308375, Train loss: 0.5844113292694092 |Validation err: 0.3685, Validation loss: 0.640621105208993
Epoch 6: Train err: 0.286, Train loss: 0.5584578578472137 |Validation err: 0.3915, Validation loss: 0.6938819326460361
Epoch 7: Train err: 0.277625, Train loss: 0.5419544172286987 |Validation err: 0.366, Validation loss: 0.6532614957541227
Epoch 8: Train err: 0.252875, Train loss: 0.514641832113266 |Validation err: 0.3555, Validation loss: 0.6758758630603552
Epoch 9: Train err: 0.248375, Train loss: 0.5126257991790771 |Validation err: 0.376, Validation loss: 0.6894327457994223
Epoch 10: Train err: 0.228, Train loss: 0.4786594619750977 |Validation err: 0.361, Validation loss: 0.6791243441402912
Epoch 11: Train err: 0.210125, Train loss: 0.45245014715194704 |Validation err: 0.355, Validation loss: 0.7089117504656315
Epoch 12: Train err: 0.19225, Train loss: 0.42779327774047854 |Validation err: 0.372, Validation loss: 0.7666997015476227
Epoch 13: Train err: 0.17425, Train loss: 0.39814749312400816 |Validation err: 0.3775, Validation loss: 0.7837714087218046
Epoch 14: Train err: 0.15775, Train loss: 0.37941932964324954 |Validation err: 0.3635, Validation loss: 0.7903527040034533
Epoch 15: Train err: 0.15325, Train loss: 0.36018795156478883 |Validation err: 0.343, Validation loss: 0.7932575158774853
Epoch 16: Train err: 0.1415, Train loss: 0.34252315664291383 |Validation err: 0.3785, Validation loss: 0.8274636808782816
Epoch 17: Train err: 0.12675, Train loss: 0.310465083360672 |Validation err: 0.368, Validation loss: 0.8701796680688858
Epoch 18: Train err: 0.110125, Train loss: 0.2772138223648071 |Validation err: 0.3565, Validation loss: 0.88609704002738
Epoch 19: Train err: 0.1015, Train loss: 0.26344901049137115 |Validation err: 0.347, Validation loss: 0.9103728383779526
Epoch 20: Train err: 0.103, Train loss: 0.26178903663158415 |Validation err: 0.3765, Validation loss: 1.0258640479296446
Epoch 21: Train err: 0.08975, Train loss: 0.24487922942638396 |Validation err: 0.3565, Validation loss: 1.0051684603095055
Epoch 22: Train err: 0.08725, Train loss: 0.22385863721370697 |Validation err: 0.36, Validation loss: 1.0111762415617704
Epoch 23: Train err: 0.061, Train loss: 0.17676357048749924 |Validation err: 0.3825, Validation loss: 1.0905357375741005
Epoch 24: Train err: 0.07525, Train loss: 0.21078718411922454 |Validation err: 0.377, Validation loss: 1.1545320823788643
Epoch 25: Train err: 0.057375, Train loss: 0.16657211792469023 |Validation err: 0.359, Validation loss: 1.094309138134122
Epoch 26: Train err: 0.051375, Train loss: 0.1537803353369236 |Validation err: 0.367, Validation loss: 1.231441343203187
Epoch 27: Train err: 0.056875, Train loss: 0.17086763229966165 |Validation err: 0.3565, Validation loss: 1.2955334819853306
Epoch 28: Train err: 0.071875, Train loss: 0.19902453374862672 |Validation err: 0.3635, Validation loss: 1.2362949214875698
Epoch 29: Train err: 0.045625, Train loss: 0.14055159774422646 |Validation err: 0.365, Validation loss: 1.2462520208209753
Epoch 30: Train err: 0.04975, Train loss: 0.15260857701301575 |Validation err: 0.3765, Validation loss: 1.3967042248696089
Finished Training
Total time elapsed: 166.37 seconds
```

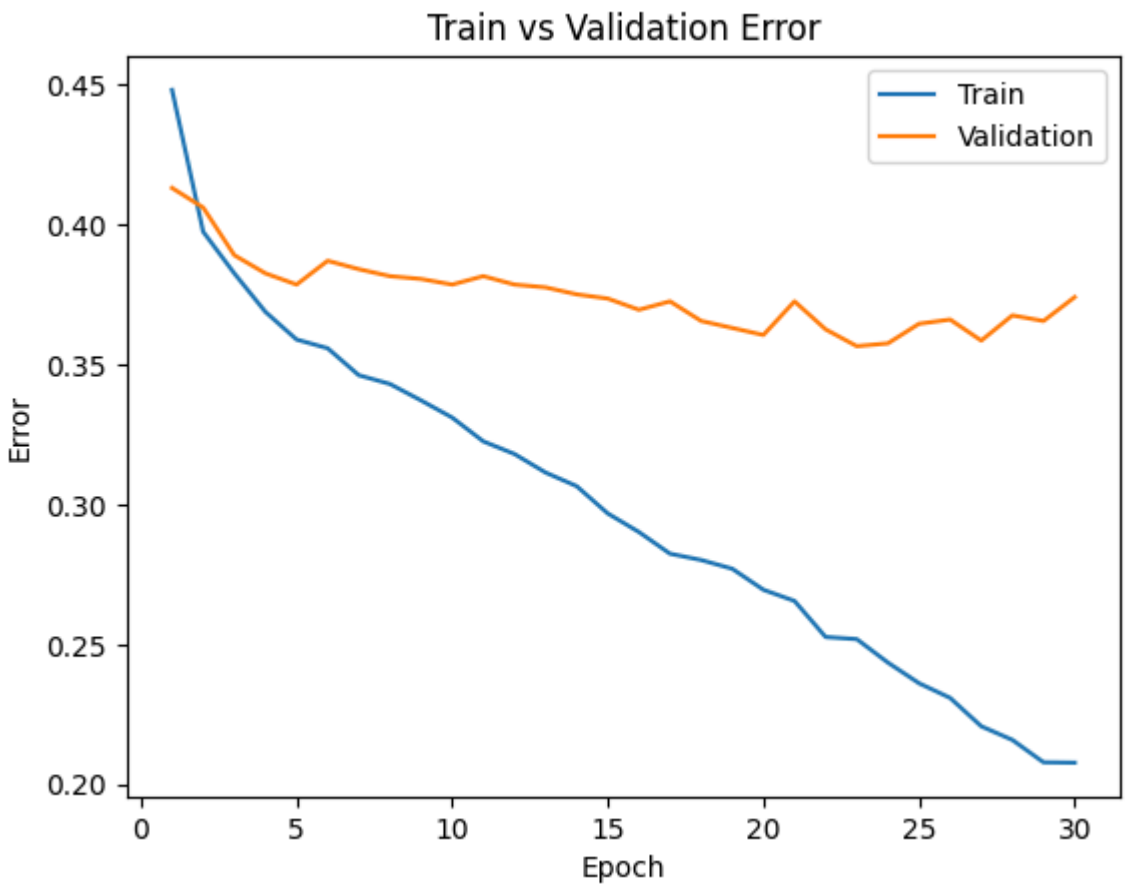
In [32]:
```
ann_path = get_model_name("ANN", batch_size=64, learning_rate=0.01, epoch=29)
plot_training_curve(ann_path)
```

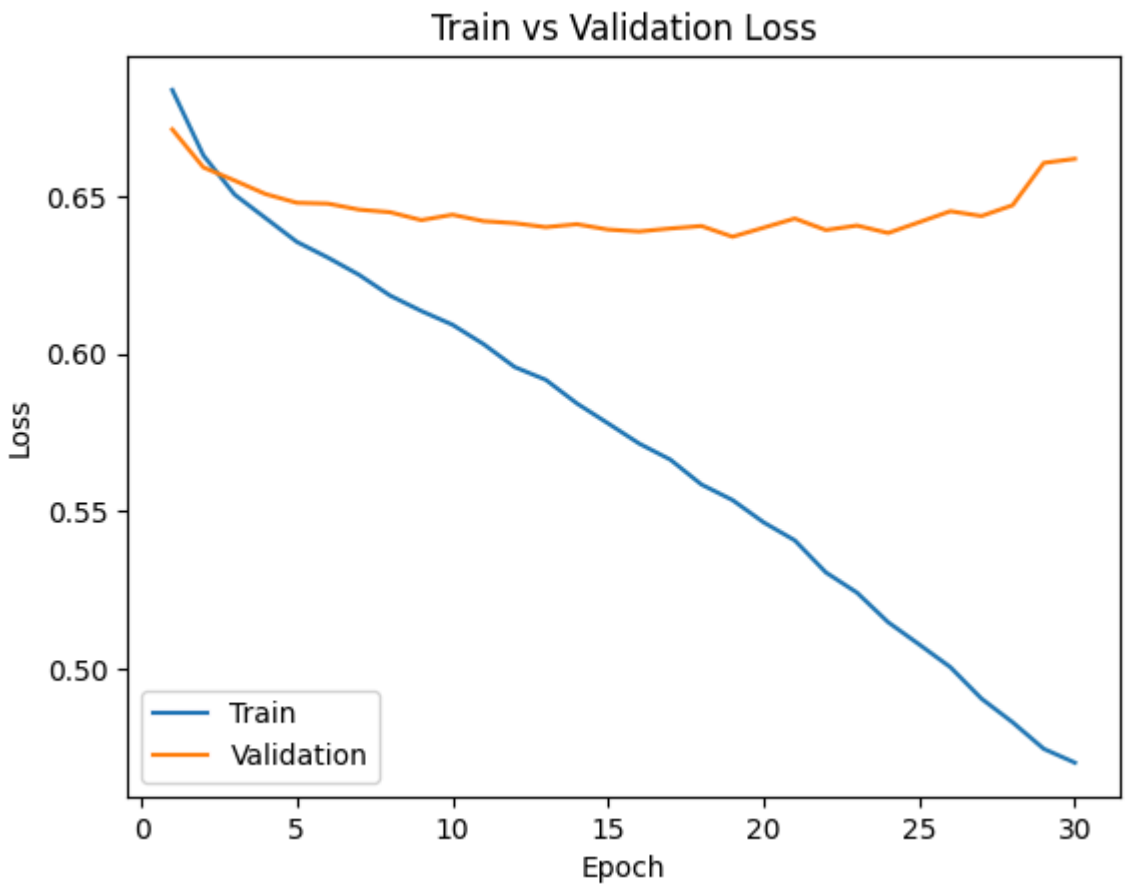


In [33]:
```
ann = ANN()
train_net(ann, batch_size=512)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.448, Train loss: 0.6837703734636307 |Validation err: 0.413, Validation loss: 0.6712464541196823
Epoch 2: Train err: 0.397375, Train loss: 0.6629869788885117 |Validation err: 0.406, Validation loss: 0.6591872870922089
Epoch 3: Train err: 0.3825, Train loss: 0.6505701690912247 |Validation err: 0.389, Validation loss: 0.6550218313932419
Epoch 4: Train err: 0.36875, Train loss: 0.6430697776377201 |Validation err: 0.3825, Validation loss: 0.650707557797432
Epoch 5: Train err: 0.358875, Train loss: 0.6355562768876553 |Validation err: 0.3785, Validation loss: 0.6479547917842865
Epoch 6: Train err: 0.35575, Train loss: 0.6305481307208538 |Validation err: 0.387, Validation loss: 0.6476765125989914
Epoch 7: Train err: 0.346125, Train loss: 0.6250808127224445 |Validation err: 0.384, Validation loss: 0.645769014954567
Epoch 8: Train err: 0.343, Train loss: 0.6184783428907394 |Validation err: 0.3815, Validation loss: 0.644948199391365
Epoch 9: Train err: 0.337125, Train loss: 0.6135965846478939 |Validation err: 0.3805, Validation loss: 0.6424409300088882
Epoch 10: Train err: 0.331, Train loss: 0.6092442087829113 |Validation err: 0.3785, Validation loss: 0.6441623717546463
Epoch 11: Train err: 0.3225, Train loss: 0.6030816920101643 |Validation err: 0.3815, Validation loss: 0.6421205997467041
Epoch 12: Train err: 0.318, Train loss: 0.5958059467375278 |Validation err: 0.3785, Validation loss: 0.6414729654788971
Epoch 13: Train err: 0.311375, Train loss: 0.5917382538318634 |Validation err: 0.3775, Validation loss: 0.6402156800031662
Epoch 14: Train err: 0.3065, Train loss: 0.5842450633645058 |Validation err: 0.375, Validation loss: 0.6411473900079727
Epoch 15: Train err: 0.29675, Train loss: 0.5779213011264801 |Validation err: 0.3735, Validation loss: 0.6394416391849518
Epoch 16: Train err: 0.290125, Train loss: 0.5715059041976929 |Validation err: 0.3695, Validation loss: 0.638837143778801
Epoch 17: Train err: 0.282375, Train loss: 0.5664167031645775 |Validation err: 0.3725, Validation loss: 0.6397869735956192
Epoch 18: Train err: 0.280125, Train loss: 0.5585118941962719 |Validation err: 0.3655, Validation loss: 0.6405715793371201
Epoch 19: Train err: 0.277, Train loss: 0.5535659864544868 |Validation err: 0.363, Validation loss: 0.6371669471263885
Epoch 20: Train err: 0.2695, Train loss: 0.5464844219386578 |Validation err: 0.3605, Validation loss: 0.6400474160909653
Epoch 21: Train err: 0.2655, Train loss: 0.5407705344259739 |Validation err: 0.3725, Validation loss: 0.6429529637098312
Epoch 22: Train err: 0.252625, Train loss: 0.5305873453617096 |Validation err: 0.3625, Validation loss: 0.6392880082130432
Epoch 23: Train err: 0.251875, Train loss: 0.5241591930389404 |Validation err: 0.3565, Validation loss: 0.6407211571931839
Epoch 24: Train err: 0.2435, Train loss: 0.5147828757762909 |Validation err: 0.3575, Validation loss: 0.6383553445339203
Epoch 25: Train err: 0.236125, Train loss: 0.5077062081545591 |Validation err: 0.3645, Validation loss: 0.6417924612760544
Epoch 26: Train err: 0.230875, Train loss: 0.5005052555352449 |Validation err: 0.366, Validation loss: 0.6452701240777969
Epoch 27: Train err: 0.22075, Train loss: 0.4905069787055254 |Validation err: 0.3585, Validation loss: 0.6437399983406067
Epoch 28: Train err: 0.215875, Train loss: 0.48300440423190594 |Validation err: 0.3675, Validation loss: 0.6471763104200363
Epoch 29: Train err: 0.207875, Train loss: 0.4745766185224056 |Validation err: 0.3655, Validation loss: 0.6606065332889557
Epoch 30: Train err: 0.20775, Train loss: 0.4701828137040138 |Validation err: 0.374, Validation loss: 0.6618973612785339
Finished Training
Total time elapsed: 126.97 seconds
```

In [35]:
```
ann_path = get_model_name("ANN", batch_size=512, learning_rate=0.01, epoch=29)
plot_training_curve(ann_path)
```

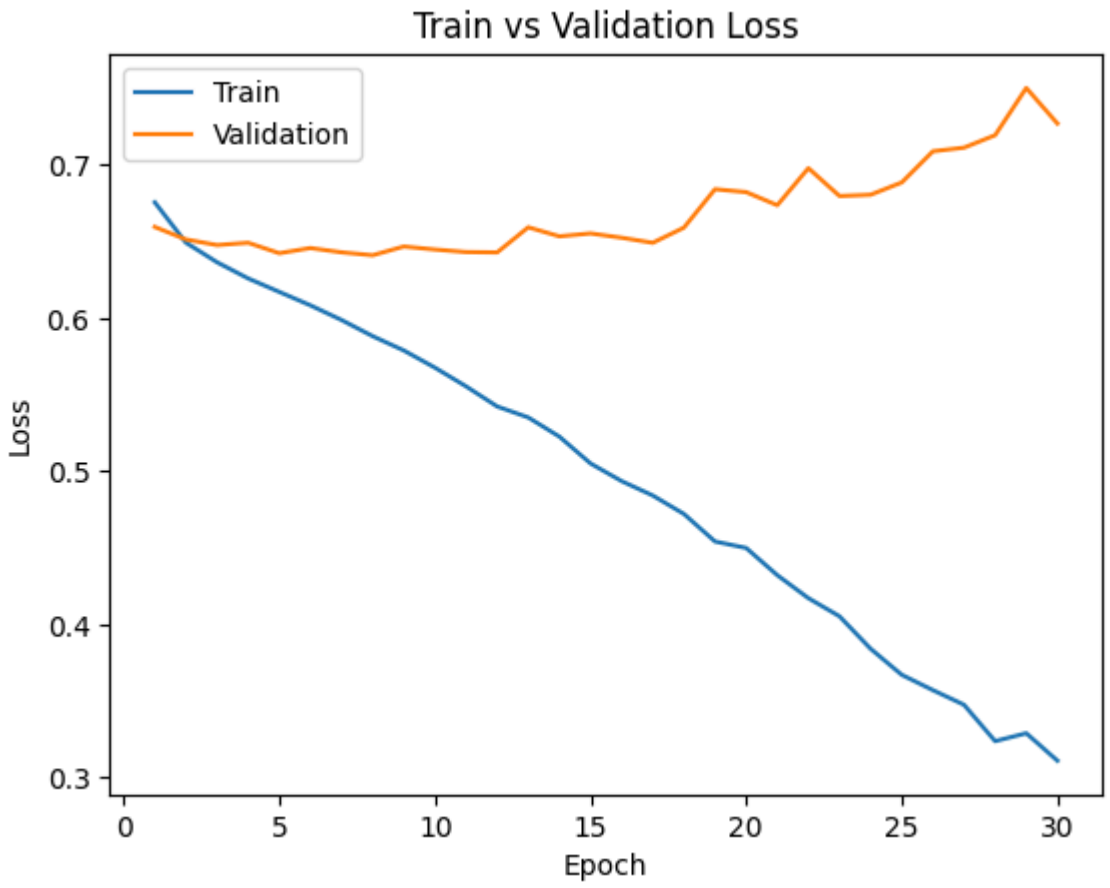




In [36]:
```
ann = ANN()
train_net(ann, batch_size=256)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.425, Train loss: 0.6756256204098463 |Validation err: 0.3965, Validation loss: 0.6595589891076088
Epoch 2: Train err: 0.383625, Train loss: 0.6493016015738249 |Validation err: 0.3845, Validation loss: 0.6513916552066803
Epoch 3: Train err: 0.366, Train loss: 0.6365065649151802 |Validation err: 0.3825, Validation loss: 0.6477233543992043
Epoch 4: Train err: 0.3525, Train loss: 0.6259343307465315 |Validation err: 0.384, Validation loss: 0.6492223888635635
Epoch 5: Train err: 0.339625, Train loss: 0.6171437539160252 |Validation err: 0.377, Validation loss: 0.6423950791358948
Epoch 6: Train err: 0.331, Train loss: 0.6083634681999683 |Validation err: 0.3845, Validation loss: 0.6457751840353012
Epoch 7: Train err: 0.324, Train loss: 0.5987990312278271 |Validation err: 0.3785, Validation loss: 0.6429669857025146
Epoch 8: Train err: 0.31175, Train loss: 0.5882361251860857 |Validation err: 0.369, Validation loss: 0.6410704925656319
Epoch 9: Train err: 0.299375, Train loss: 0.5789091177284718 |Validation err: 0.3845, Validation loss: 0.6467743366956711
Epoch 10: Train err: 0.292875, Train loss: 0.5676777120679617 |Validation err: 0.37, Validation loss: 0.6448171734809875
Epoch 11: Train err: 0.2795, Train loss: 0.55547502823174 |Validation err: 0.3655, Validation loss: 0.6431547403335571
Epoch 12: Train err: 0.267625, Train loss: 0.5423325337469578 |Validation err: 0.361, Validation loss: 0.6429204419255257
Epoch 13: Train err: 0.265875, Train loss: 0.5351567063480616 |Validation err: 0.3725, Validation loss: 0.6593351289629936
Epoch 14: Train err: 0.254, Train loss: 0.5226200064644217 |Validation err: 0.375, Validation loss: 0.6534329503774643
Epoch 15: Train err: 0.232875, Train loss: 0.5051219947636127 |Validation err: 0.3605, Validation loss: 0.655324287712574
Epoch 16: Train err: 0.228, Train loss: 0.49357288889586926 |Validation err: 0.363, Validation loss: 0.652514860033989
Epoch 17: Train err: 0.220625, Train loss: 0.4841812364757061 |Validation err: 0.3585, Validation loss: 0.6491980776190758
Epoch 18: Train err: 0.21025, Train loss: 0.47214608173817396 |Validation err: 0.359, Validation loss: 0.6591078862547874
Epoch 19: Train err: 0.203875, Train loss: 0.4541560159996152 |Validation err: 0.3675, Validation loss: 0.6841889172792435
Epoch 20: Train err: 0.20475, Train loss: 0.4499262059107423 |Validation err: 0.367, Validation loss: 0.6822977066040039
Epoch 21: Train err: 0.18925, Train loss: 0.4321643877774477 |Validation err: 0.3665, Validation loss: 0.6737541183829308
Epoch 22: Train err: 0.175375, Train loss: 0.41714267525821924 |Validation err: 0.3665, Validation loss: 0.6979332566261292
Epoch 23: Train err: 0.169625, Train loss: 0.40528510324656963 |Validation err: 0.3625, Validation loss: 0.6797430515289307
Epoch 24: Train err: 0.1565, Train loss: 0.38408661913126707 |Validation err: 0.3575, Validation loss: 0.680581346154213
Epoch 25: Train err: 0.1405, Train loss: 0.3670927369967103 |Validation err: 0.372, Validation loss: 0.6886191889643669
Epoch 26: Train err: 0.1365, Train loss: 0.3570960061624646 |Validation err: 0.3555, Validation loss: 0.7089632079005241
Epoch 27: Train err: 0.13225, Train loss: 0.3475195895880461 |Validation err: 0.363, Validation loss: 0.711324617266655
Epoch 28: Train err: 0.1165, Train loss: 0.32377623673528433 |Validation err: 0.3555, Validation loss: 0.7194659113883972
Epoch 29: Train err: 0.12975, Train loss: 0.32894963677972555 |Validation err: 0.361, Validation loss: 0.7503189295530319
Epoch 30: Train err: 0.114, Train loss: 0.3110282402485609 |Validation err: 0.3605, Validation loss: 0.7271039038896561
Finished Training
Total time elapsed: 131.51 seconds
```

In [37]:
```
ann_path = get_model_name("ANN", batch_size=256, learning_rate=0.01, epoch=29)
plot_training_curve(ann_path)
```



In [ ]:
```
#The best model is from Batch size = 512
#Epoch 24: Train err: 0.2435, Train loss: 0.5147828757762909 |Validation err: 0.3575, Validation loss: 0.6383553445339203
```

In [38]:
```
ann_net = ANN()
model_path = get_model_name(ann_net.name, batch_size=512, learning_rate=0.01, epoch=23)
state = torch.load(model_path)
ann_net.load_state_dict(state)
```

```
<ipython-input-38-1214e49913de>:3: FutureWarning: You are using `torch.load` with `weights_only=False` (the current default valu
e), which uses the default pickle module implicitly. It is possible to construct malicious pickle data which will execute arbitr
ary code during unpickling (See https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for more details). In
a future release, the default value for `weights_only` will be flipped to `True`. This limits the functions that could be execut
ed during unpickling. Arbitrary objects will no longer be allowed to be loaded via this mode unless they are explicitly allowlis
ted by the user via `torch.serialization.add_safe_globals`. We recommend you start setting `weights_only=True` for any use case
where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experimental
feature.
  state = torch.load(model_path)
```

Out[38]:
```
<All keys matched successfully>
```

In [39]:
```
train_loader, val_loader, test_loader, classes = get_data_loader(
    target_classes=["cat", "dog"],
    batch_size=64)
criterion = nn.BCEWithLogitsLoss()
err, loss = evaluate(ann_net, test_loader, criterion)
print(f"Prediction Error: {err}, BCE Error: {loss}")
```

```
Files already downloaded and verified
Files already downloaded and verified
Prediction Error: 0.346, BCE Error: 0.6320203319191933
```

The best CNN model is able to achieve

Prediction Error: 0.277, BCE Error: 0.5437365910038352

The best ANN model is able to achieve

Prediction Error: 0.346, BCE Error: 0.6320203319191933

Therefore, CNN performs better, as it has lower error.