Lab 2: Cats vs Dogs

Deadline: Feb 01, 5:00pm

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

Marking TA: Tinglin (Francis) Duan

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

In this lab, you will train a convolutional neural network to classify an image into one of two classes: "cat" or "dog". The code for the neural networks you train will be written for you, and you are not (yet!) expected to understand all provided code. However, by the end of the lab, you should be able to:

1. Understand at a high level the training loop for a machine learning model.
2. Understand the distinction between training, validation, and test data.
3. The concepts of overfitting and underfitting.
4. Investigate how different hyperparameters, such as learning rate and batch size, affect the success of training.
5. Compare an ANN (aka Multi-Layer Perceptron) with a CNN.

What to submit

Submit a PDF file containing all your code, outputs, and write-up from parts 1-5. You can produce a PDF of your Google Colab file by going to **File > Print** and then save as PDF. The Colab instructions has more information.

Do not submit any other files produced by your code.

Include a link to your colab file in your submission.

Please use Google Colab to complete this assignment. If you want to use Jupyter Notebook, please complete the assignment and upload your Jupyter Notebook file to Google Colab for submission.

With Colab, you can export a PDF file using the menu option `File -> Print` and save as PDF file. **Adjust the scaling to ensure that the text is not cutoff at the margins.**

Colab Link

Include a link to your colab file here

Colab Link: https://colab.research.google.com/drive/11fmZwB9E_jdLVS-UP23Wd4_9Y83qOlRz?usp=sharing (https://colab.research.google.com/drive/11fmZwB9E_jdLVS-UP23Wd4_9Y83qOlRz?usp=sharing)

In [2]:
```
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 [3]:
```
##########################################################################
#######
# 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 c
lasses
                        Should be a subset of the 'classes'
    Returns:
        indices: list of indices that have labels corresponding to one o
f 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, va
lidation
    and testing datasets. Returns data loaders for the three preprocesse
d datasets.

    Args:
        target_classes: A list of strings denoting the name of the desir
ed
                        classes. Should be a subset of the argument 'cla
sses'
        batch_size: A int representing the number of samples per batch

    Returns:
        train_loader: iterable training dataset organized according to b
atch size
        val_loader: iterable validation dataset organized according to b
atch size
        test_loader: iterable testing dataset organized according to bat
ch 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=tra
nsform)
    # Get the list of indices to sample from
    relevant_indices = get_relevant_indices(trainset, classes, target_cl
asses)

    # Split into train and validation
    np.random.seed(1000) # Fixed numpy random seed for reproducible shuf
fling
    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[:spl
it], relevant_indices[split:]
    train_sampler = SubsetRandomSampler(relevant_train_indices)
    train_loader = torch.utils.data.DataLoader(trainset, batch_size=batc
h_size,
                                               num_workers=1, sampler=tr
ain_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=tran
sform)
    # Get the list of indices to sample from
    relevant_test_indices = get_relevant_indices(testset, classes, targe
t_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 hyperparamet
er values

    Args:
        config: Configuration object containing the hyperparameters
    Returns:
```

```
        path: A string with the hyperparameter name and value concatenat
ed
    """
    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 validat
ion set
         loss: A scalar for the average loss function over the validatio
n set
     """
    total_loss = 0.0
    total_err = 0.0
    total_epoch = 0
    for i, data in enumerate(loader, 0):
        inputs, labels = data
        labels = normalize_label(labels)  # Convert labels to 0/1
        outputs = net(inputs)
        loss = criterion(outputs, labels.float())
        corr = (outputs > 0.0).squeeze().long() != labels
        total_err += int(corr.sum())
        total_loss += loss.item()
        total_epoch += len(labels)
    err = float(total_err) / total_epoch
    loss = float(total_loss) / (i + 1)
    return err, loss

##########################################################################
#######
# Training Curve
def plot_training_curve(path):
    """ Plots the training curve for a model run, given the csv files
```

```
    containing the train/validation error/loss.

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

Part 1. Visualizing the Data [7 pt]

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

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

In [4]:
```
# 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


Extracting ./data/cifar-10-python.tar.gz to ./data
Files already downloaded and verified
```

Part (a) -- 1 pt

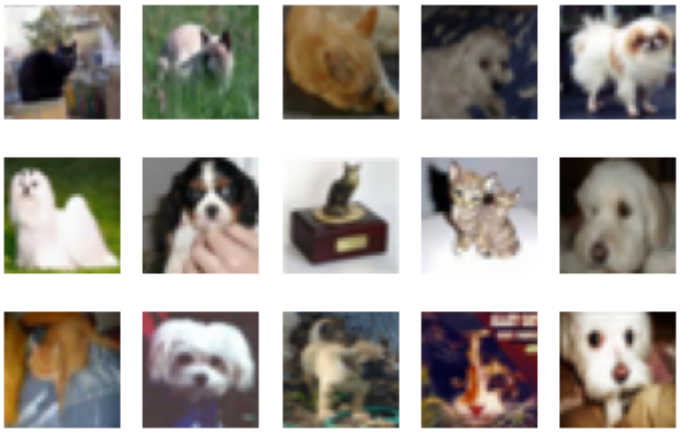
Visualize some of the data by running the code below. Include the visualization in your writeup.

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

In [5]:
```
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 [6]:
```
print("Training Examples =", len(train_loader))
print("Validation Examples =", len(val_loader))
print("Test Examples =", len(test_loader))
```

```
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?

We need validation set when training our model to avoid overfitting. We can use the validation set to tune the higher level hyperparameters. The model directly learns from a training set, whereas the model is indirectly learning from the validation set by the user changing the hyperparameters. If we judge the performance of our models using the training set loss/error instead of validation set loss/error, we could be overfitting our data.

In [6]:

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 [7]:
```
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 [8]:
```
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 [9]:
```
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 [10]:
```
for param in small_net.parameters():
    print(param.shape)
```

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

In [11]:
```
for param in small_net.parameters():
    print(param.shape.numel())
```

```
135
5
245
1
```

Total number of parameters in small_net = 386

In [12]:
```
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])
```

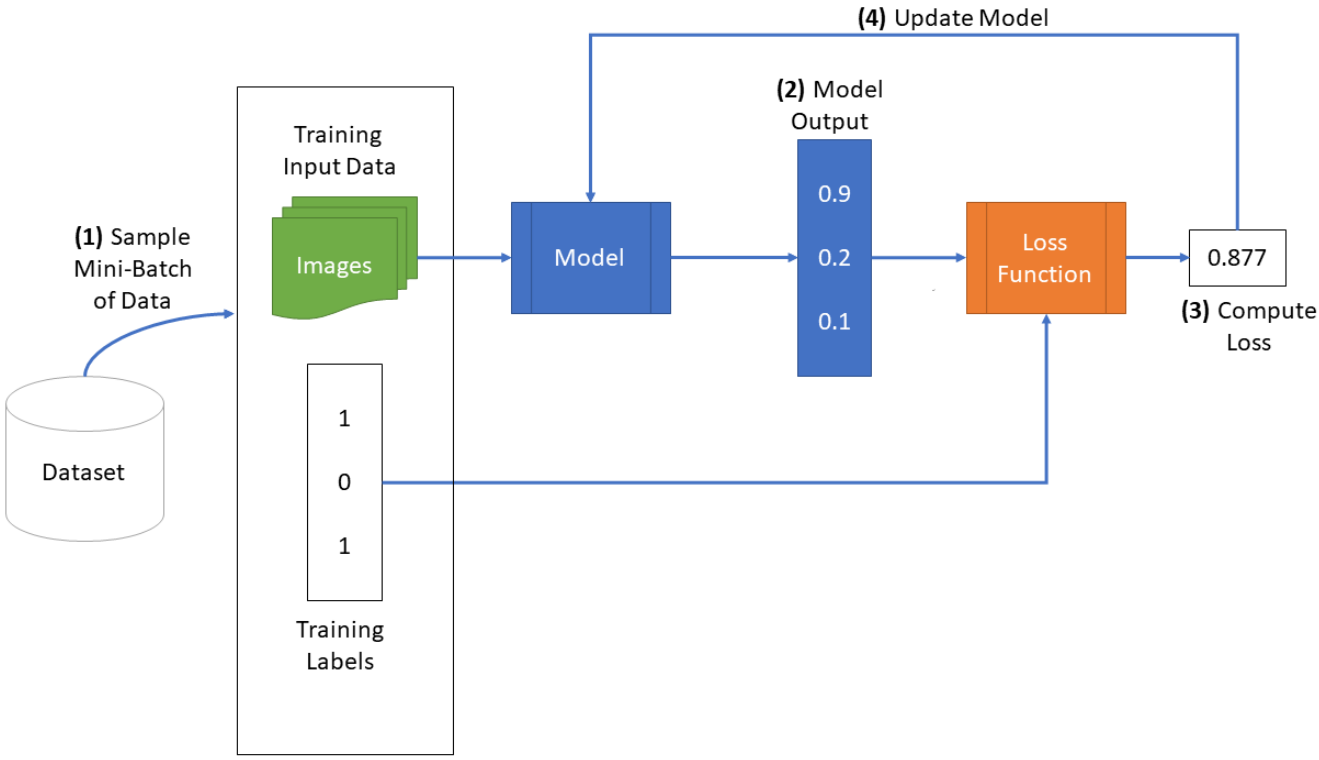
In [13]:
```
for param in large_net.parameters():
    print(param.shape.numel())
```

```
375
5
1250
10
8000
32
32
1
```

Total number of parameters in large_net = 9705

The function train_net

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



In [14]:
```
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 data sets
    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, crit
erion)
        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`?

Default value batch_size: 64

Default value learning_rate: 0.01

Default value num_epochs: 30

Part (c) -- 3 pt

What files are written to disk when we call `train_net` with `small_net`, and train for 5 epochs? Provide a list of all the files written to disk, and what information the files contain.

model_small_bs64_lr0.01_epoch0

model_small_bs64_lr0.01_epoch1

model_small_bs64_lr0.01_epoch2

model_small_bs64_lr0.01_epoch3

model_small_bs64_lr0.01_epoch4

model_small_bs64_lr0.01_epoch4_train_err.csv (training set error for epoch 4(5))

model_small_bs64_lr0.01_epoch4_train_loss.csv (training set loss for epoch 4(5))

model_small_bs64_lr0.01_epoch4_val_err.csv (validation set error for epoch 4(5))

model_small_bs64_lr0.01_epoch4_val_loss.csv (validation set loss for epoch 4(5))

In [15]:
```
train_net(small_net)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.427625, Train loss: 0.6741959056854248 |Validatio
n err: 0.3775, Validation loss: 0.6597265005111694
Epoch 2: Train err: 0.365125, Train loss: 0.6430219621658325 |Validatio
n err: 0.3705, Validation loss: 0.6553609501570463
Epoch 3: Train err: 0.347, Train loss: 0.6270276293754578 |Validation e
rr: 0.348, Validation loss: 0.6260313391685486
Epoch 4: Train err: 0.333625, Train loss: 0.6116100740432739 |Validatio
n err: 0.3565, Validation loss: 0.6285025924444199
Epoch 5: Train err: 0.32075, Train loss: 0.6046628150939941 |Validation
err: 0.3395, Validation loss: 0.6177340876311064
Epoch 6: Train err: 0.3145, Train loss: 0.5928595442771911 |Validation
err: 0.3355, Validation loss: 0.6153966169804335
Epoch 7: Train err: 0.310625, Train loss: 0.5870834405422211 |Validatio
n err: 0.3285, Validation loss: 0.6046376973390579
Epoch 8: Train err: 0.303625, Train loss: 0.5786808829307556 |Validatio
n err: 0.323, Validation loss: 0.6003656964749098
Epoch 9: Train err: 0.297125, Train loss: 0.5744030501842499 |Validatio
n err: 0.312, Validation loss: 0.5955771449953318
Epoch 10: Train err: 0.293125, Train loss: 0.5672098655700684 |Validati
on err: 0.3095, Validation loss: 0.5867132395505905
Epoch 11: Train err: 0.284875, Train loss: 0.565056690454483 |Validatio
n err: 0.3125, Validation loss: 0.5887837186455727
Epoch 12: Train err: 0.2875, Train loss: 0.5605644233226776 |Validation
err: 0.3135, Validation loss: 0.5852970713749528
Epoch 13: Train err: 0.28425, Train loss: 0.560438283920288 |Validation
err: 0.302, Validation loss: 0.584381926804781
Epoch 14: Train err: 0.283, Train loss: 0.5547531487941741 |Validation
err: 0.3175, Validation loss: 0.5920804627239704
Epoch 15: Train err: 0.28075, Train loss: 0.5518253128528595 |Validatio
n err: 0.3115, Validation loss: 0.596444352529943
Epoch 16: Train err: 0.280625, Train loss: 0.555060781955719 |Validatio
n err: 0.3105, Validation loss: 0.5944277262315154
Epoch 17: Train err: 0.276625, Train loss: 0.5498158397674561 |Validati
on err: 0.2955, Validation loss: 0.5777709009125829
Epoch 18: Train err: 0.272375, Train loss: 0.5473316473960876 |Validati
on err: 0.302, Validation loss: 0.5780399432405829
Epoch 19: Train err: 0.26975, Train loss: 0.5431778583526611 |Validatio
n err: 0.3045, Validation loss: 0.5853669550269842
Epoch 20: Train err: 0.2755, Train loss: 0.5432542805671692 |Validation
err: 0.2945, Validation loss: 0.582625100389123
Epoch 21: Train err: 0.275875, Train loss: 0.5445209119319916 |Validati
on err: 0.2905, Validation loss: 0.5738702621310949
Epoch 22: Train err: 0.27225, Train loss: 0.5436846029758453 |Validatio
n err: 0.299, Validation loss: 0.5847234949469566
Epoch 23: Train err: 0.266875, Train loss: 0.5407049005031586 |Validati
on err: 0.302, Validation loss: 0.5728532336652279
Epoch 24: Train err: 0.2745, Train loss: 0.5411817135810852 |Validation
err: 0.29, Validation loss: 0.579209977760911
Epoch 25: Train err: 0.272125, Train loss: 0.5385495419502259 |Validati
on err: 0.3035, Validation loss: 0.5823763059452176
Epoch 26: Train err: 0.26825, Train loss: 0.5372853455543518 |Validatio
n err: 0.294, Validation loss: 0.5718310056254268
Epoch 27: Train err: 0.265375, Train loss: 0.5354727454185486 |Validati
on err: 0.3105, Validation loss: 0.594637275673449
Epoch 28: Train err: 0.268875, Train loss: 0.5384577159881592 |Validati
```

```
on err: 0.3, Validation loss: 0.5777103006839752
Epoch 29: Train err: 0.26775, Train loss: 0.5374719173908233 |Validatio
n err: 0.2935, Validation loss: 0.5779346721246839
Epoch 30: Train err: 0.266625, Train loss: 0.5363995409011841 |Validati
on err: 0.292, Validation loss: 0.5878843413665891
Finished Training
Total time elapsed: 111.43 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 [16]:
```
# 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 [17]:
```
train_net(small_net)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.27, Train loss: 0.53282723402977 |Validation err:
0.295, Validation loss: 0.5743504017591476
Epoch 2: Train err: 0.269625, Train loss: 0.5335272345542907 |Validatio
n err: 0.297, Validation loss: 0.5843025427311659
Epoch 3: Train err: 0.267, Train loss: 0.5322322626113891 |Validation e
rr: 0.293, Validation loss: 0.5706519680097699
Epoch 4: Train err: 0.267125, Train loss: 0.5336958305835724 |Validatio
n err: 0.2915, Validation loss: 0.5748661998659372
Epoch 5: Train err: 0.265375, Train loss: 0.5314749937057495 |Validatio
n err: 0.2855, Validation loss: 0.5670599397271872
Epoch 6: Train err: 0.2675, Train loss: 0.5314255261421204 |Validation
err: 0.2975, Validation loss: 0.5833839969709516
Epoch 7: Train err: 0.26975, Train loss: 0.5325355689525604 |Validation
err: 0.292, Validation loss: 0.5673605212941766
Epoch 8: Train err: 0.263, Train loss: 0.5310534467697143 |Validation e
rr: 0.2805, Validation loss: 0.5656690709292889
Epoch 9: Train err: 0.268875, Train loss: 0.5327912881374359 |Validatio
n err: 0.289, Validation loss: 0.5801336131989956
Epoch 10: Train err: 0.262375, Train loss: 0.5289756796360016 |Validati
on err: 0.293, Validation loss: 0.571611300110817
Epoch 11: Train err: 0.2595, Train loss: 0.528243061542511 |Validation
err: 0.295, Validation loss: 0.5687992004677653
Epoch 12: Train err: 0.262125, Train loss: 0.5257628262042999 |Validati
on err: 0.2775, Validation loss: 0.5660881763324142
Epoch 13: Train err: 0.26775, Train loss: 0.5319077432155609 |Validatio
n err: 0.2945, Validation loss: 0.5734228445217013
Epoch 14: Train err: 0.26425, Train loss: 0.5272781596183777 |Validatio
n err: 0.2865, Validation loss: 0.5835625343024731
Epoch 15: Train err: 0.26425, Train loss: 0.525968053817749 |Validation
err: 0.2925, Validation loss: 0.5708476528525352
Epoch 16: Train err: 0.262875, Train loss: 0.5313924973011017 |Validati
on err: 0.2905, Validation loss: 0.5806977637112141
Epoch 17: Train err: 0.26275, Train loss: 0.5270006461143494 |Validatio
n err: 0.278, Validation loss: 0.565776789560914
Epoch 18: Train err: 0.26375, Train loss: 0.5260548992156983 |Validatio
n err: 0.287, Validation loss: 0.569947924464941
Epoch 19: Train err: 0.260125, Train loss: 0.5235783925056458 |Validati
on err: 0.2865, Validation loss: 0.5734773082658648
Epoch 20: Train err: 0.26075, Train loss: 0.5246248772144317 |Validatio
n err: 0.278, Validation loss: 0.5720528559759259
Epoch 21: Train err: 0.26125, Train loss: 0.5259754571914673 |Validatio
n err: 0.2795, Validation loss: 0.5666486155241728
Epoch 22: Train err: 0.26225, Train loss: 0.5267154183387757 |Validatio
n err: 0.279, Validation loss: 0.5834407983347774
Epoch 23: Train err: 0.25825, Train loss: 0.525192144870758 |Validation
err: 0.286, Validation loss: 0.5667675230652094
Epoch 24: Train err: 0.259125, Train loss: 0.5241440949440003 |Validati
on err: 0.2875, Validation loss: 0.562714571133256
Epoch 25: Train err: 0.26675, Train loss: 0.5245032577514649 |Validatio
n err: 0.2885, Validation loss: 0.5712021542713046
Epoch 26: Train err: 0.26275, Train loss: 0.5233962409496308 |Validatio
n err: 0.285, Validation loss: 0.5646006958559155
Epoch 27: Train err: 0.256375, Train loss: 0.5199962890148163 |Validati
on err: 0.3005, Validation loss: 0.5806670319288969
Epoch 28: Train err: 0.26175, Train loss: 0.5266856839656829 |Validatio
```

```
n err: 0.2805, Validation loss: 0.5639690523967147
Epoch 29: Train err: 0.26475, Train loss: 0.5261635258197784 |Validatio
n err: 0.2965, Validation loss: 0.572383938357234
Epoch 30: Train err: 0.26475, Train loss: 0.5271118569374085 |Validatio
n err: 0.2975, Validation loss: 0.575899419374764
Finished Training
Total time elapsed: 112.26 seconds
```

In [18]:
```
large_net = LargeNet()
train_net(large_net, 64, 0.01, 30)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.44475, Train loss: 0.6900203123092651 |Validation
err: 0.4285, Validation loss: 0.6807542946189642
Epoch 2: Train err: 0.4195, Train loss: 0.67819615650177 |Validation er
r: 0.413, Validation loss: 0.6741204150021076
Epoch 3: Train err: 0.39875, Train loss: 0.6658886175155639 |Validation
err: 0.3925, Validation loss: 0.6518177818506956
Epoch 4: Train err: 0.373875, Train loss: 0.6490470728874207 |Validatio
n err: 0.406, Validation loss: 0.6635930277407169
Epoch 5: Train err: 0.35375, Train loss: 0.6330237889289856 |Validation
err: 0.353, Validation loss: 0.6284337677061558
Epoch 6: Train err: 0.339125, Train loss: 0.6160062110424042 |Validatio
n err: 0.34, Validation loss: 0.6150468923151493
Epoch 7: Train err: 0.32625, Train loss: 0.6001663441658021 |Validation
err: 0.335, Validation loss: 0.6086486000567675
Epoch 8: Train err: 0.313875, Train loss: 0.5828366029262543 |Validatio
n err: 0.3325, Validation loss: 0.5983812175691128
Epoch 9: Train err: 0.3085, Train loss: 0.5771152818202973 |Validation
err: 0.315, Validation loss: 0.5944901742041111
Epoch 10: Train err: 0.2935, Train loss: 0.5622252209186553 |Validation
err: 0.3155, Validation loss: 0.5953319733962417
Epoch 11: Train err: 0.284375, Train loss: 0.5485934205055237 |Validati
on err: 0.3155, Validation loss: 0.611855155788362
Epoch 12: Train err: 0.2785, Train loss: 0.5407600071430206 |Validation
err: 0.31, Validation loss: 0.5998768676072359
Epoch 13: Train err: 0.278125, Train loss: 0.5309763443470001 |Validati
on err: 0.297, Validation loss: 0.583570459857583
Epoch 14: Train err: 0.26275, Train loss: 0.5173734092712402 |Validatio
n err: 0.2985, Validation loss: 0.5952338296920061
Epoch 15: Train err: 0.2555, Train loss: 0.5114965040683747 |Validation
err: 0.2975, Validation loss: 0.5921047143638134
Epoch 16: Train err: 0.249625, Train loss: 0.5027769944667816 |Validati
on err: 0.3025, Validation loss: 0.6024623941630125
Epoch 17: Train err: 0.247125, Train loss: 0.49459292030334473 |Validat
ion err: 0.298, Validation loss: 0.5841752551496029
Epoch 18: Train err: 0.235, Train loss: 0.4815113615989685 |Validation
err: 0.3055, Validation loss: 0.5952335279434919
Epoch 19: Train err: 0.234375, Train loss: 0.4761629197597504 |Validati
on err: 0.3205, Validation loss: 0.6141199134290218
Epoch 20: Train err: 0.225875, Train loss: 0.4626086118221283 |Validati
on err: 0.3085, Validation loss: 0.6088314475491643
Epoch 21: Train err: 0.219125, Train loss: 0.4528284652233124 |Validati
on err: 0.287, Validation loss: 0.5983596304431558
Epoch 22: Train err: 0.214125, Train loss: 0.4414942805767059 |Validati
on err: 0.3045, Validation loss: 0.6131745660677552
Epoch 23: Train err: 0.209125, Train loss: 0.4349266333580017 |Validati
on err: 0.301, Validation loss: 0.6113477284088731
Epoch 24: Train err: 0.202625, Train loss: 0.4223342254161835 |Validati
on err: 0.3195, Validation loss: 0.6655794447287917
Epoch 25: Train err: 0.194, Train loss: 0.4071682426929474 |Validation
err: 0.3015, Validation loss: 0.6275045238435268
Epoch 26: Train err: 0.181, Train loss: 0.38940074408054354 |Validation
err: 0.309, Validation loss: 0.662933480925858
Epoch 27: Train err: 0.1765, Train loss: 0.3803082858324051 |Validation
err: 0.303, Validation loss: 0.6525407964363694
Epoch 28: Train err: 0.16925, Train loss: 0.37183564841747285 |Validati
```

```
on err: 0.2975, Validation loss: 0.6434079697355628
Epoch 29: Train err: 0.1605, Train loss: 0.35386401546001434 |Validatio
n err: 0.319, Validation loss: 0.8143855566158891
Epoch 30: Train err: 0.1595, Train loss: 0.3514791972637176 |Validation
err: 0.3025, Validation loss: 0.702620318159461
Finished Training
Total time elapsed: 125.25 seconds
```

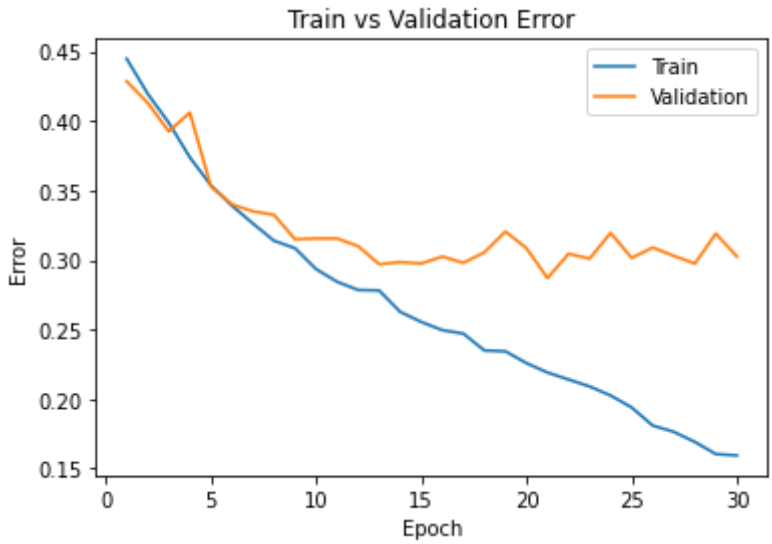
Part (e) - 2pt

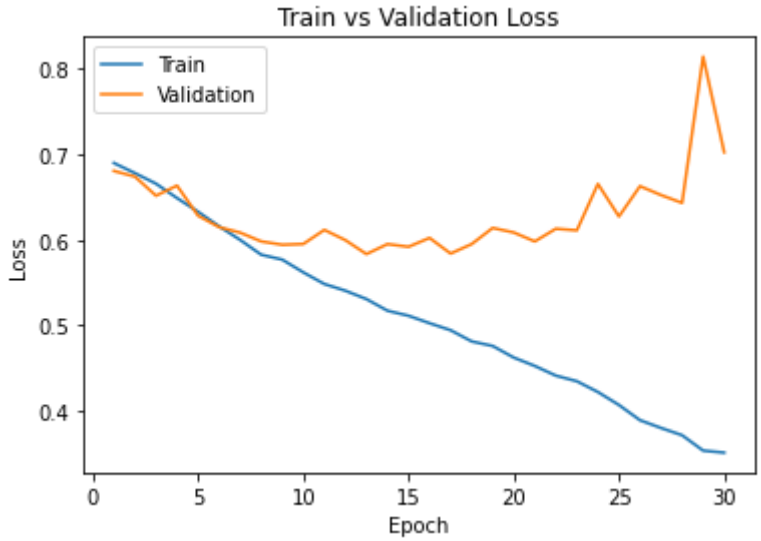
Use the function `plot_training_curve` to display the trajectory of the training/validation error and the training/validation loss. You will need to use the function `get_model_name` to generate the argument to the `plot_training_curve` function.

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

In [19]:
```
plot_training_curve(get_model_name("small", batch_size=64, learning_rate
=0.01, epoch=29))
```

In [20]:
```
plot_training_curve(get_model_name("large", batch_size=64, learning_rate
=0.01, epoch=29))
```





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.

The training curve for small_net is somewhat stable (oscillates up and down) as the number of epoch increases, whereas the training curve for large_net is continuously decreasing as the number of epoch increses.

The small_net seems to be overfitting because the error on the training set is low, but the error on the validation set is large.

The large_net seems to be underfitting at lower number of epochs because both the error on the training set and the validation set is high. As the number of epoch increases, large_net seems to be overfitting because the error on the training set is low, whereas the error on the validation set is large.

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 [21]:
```
# 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, 64, 0.001, 30)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.47625, Train loss: 0.6928360013961792 |Validation
err: 0.467, Validation loss: 0.6924686580896378
Epoch 2: Train err: 0.448625, Train loss: 0.6922589712142945 |Validatio
n err: 0.4305, Validation loss: 0.691649341955781
Epoch 3: Train err: 0.43575, Train loss: 0.6916067280769348 |Validation
err: 0.4285, Validation loss: 0.690854424610734
Epoch 4: Train err: 0.43, Train loss: 0.690861343383789 |Validation er
r: 0.424, Validation loss: 0.6896595880389214
Epoch 5: Train err: 0.434125, Train loss: 0.6899195008277893 |Validatio
n err: 0.4195, Validation loss: 0.6886935643851757
Epoch 6: Train err: 0.43575, Train loss: 0.6887411961555481 |Validation
err: 0.4195, Validation loss: 0.6867824867367744
Epoch 7: Train err: 0.437125, Train loss: 0.6873774147033691 |Validatio
n err: 0.4185, Validation loss: 0.6851982977241278
Epoch 8: Train err: 0.4375, Train loss: 0.6859278454780579 |Validation
err: 0.412, Validation loss: 0.683199780061841
Epoch 9: Train err: 0.424375, Train loss: 0.6844058036804199 |Validatio
n err: 0.411, Validation loss: 0.6808880660682917
Epoch 10: Train err: 0.424, Train loss: 0.6828502931594849 |Validation
err: 0.408, Validation loss: 0.6783502567559481
Epoch 11: Train err: 0.425375, Train loss: 0.6812348766326904 |Validati
on err: 0.4125, Validation loss: 0.6780214440077543
Epoch 12: Train err: 0.42, Train loss: 0.6796319708824158 |Validation e
rr: 0.4125, Validation loss: 0.6753159202635288
Epoch 13: Train err: 0.414875, Train loss: 0.6777918744087219 |Validati
on err: 0.415, Validation loss: 0.6757059413939714
Epoch 14: Train err: 0.412375, Train loss: 0.6761112003326416 |Validati
on err: 0.412, Validation loss: 0.6739734839648008
Epoch 15: Train err: 0.40925, Train loss: 0.674472680568695 |Validation
err: 0.415, Validation loss: 0.6706762500107288
Epoch 16: Train err: 0.406375, Train loss: 0.6727448840141297 |Validati
on err: 0.4105, Validation loss: 0.6707733049988747
Epoch 17: Train err: 0.4015, Train loss: 0.6713076601028443 |Validation
err: 0.4045, Validation loss: 0.6671545393764973
Epoch 18: Train err: 0.3995, Train loss: 0.6696742882728577 |Validation
err: 0.4055, Validation loss: 0.6646782550960779
Epoch 19: Train err: 0.40075, Train loss: 0.6679086356163025 |Validatio
n err: 0.396, Validation loss: 0.6655019577592611
Epoch 20: Train err: 0.392375, Train loss: 0.665787980556488 |Validatio
n err: 0.405, Validation loss: 0.6626011095941067
Epoch 21: Train err: 0.38975, Train loss: 0.6646300601959229 |Validatio
n err: 0.394, Validation loss: 0.660687854513526
Epoch 22: Train err: 0.388875, Train loss: 0.662373058795929 |Validatio
n err: 0.393, Validation loss: 0.6616998575627804
Epoch 23: Train err: 0.38425, Train loss: 0.6601516346931458 |Validatio
n err: 0.3975, Validation loss: 0.6573981791734695
Epoch 24: Train err: 0.382375, Train loss: 0.6584009389877319 |Validati
on err: 0.386, Validation loss: 0.6561364810913801
Epoch 25: Train err: 0.37875, Train loss: 0.6554971766471863 |Validatio
n err: 0.388, Validation loss: 0.6552744228392839
Epoch 26: Train err: 0.376625, Train loss: 0.6531173253059387 |Validati
on err: 0.3875, Validation loss: 0.6531743723899126
Epoch 27: Train err: 0.375, Train loss: 0.6503696331977844 |Validation
err: 0.387, Validation loss: 0.6519789285957813
Epoch 28: Train err: 0.371375, Train loss: 0.6476435809135437 |Validati
```

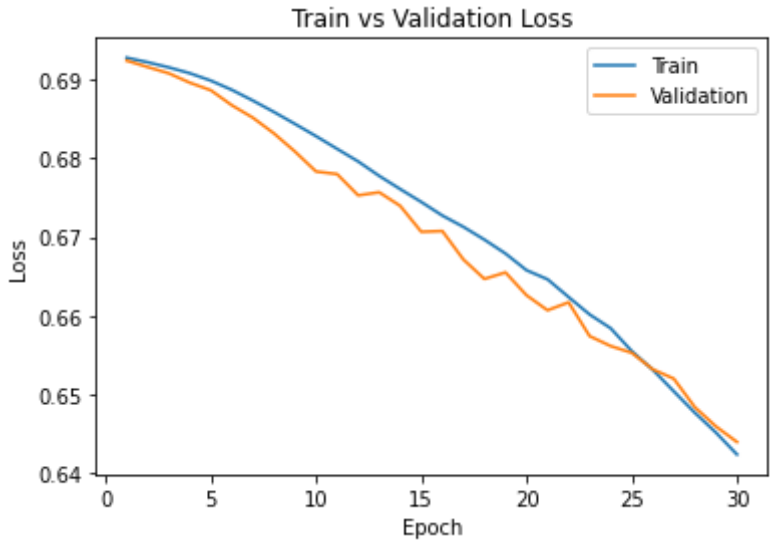
```
on err: 0.3875, Validation loss: 0.6483502741903067
Epoch 29: Train err: 0.368375, Train loss: 0.6451257643699646 |Validati
on err: 0.3825, Validation loss: 0.6459067314863205
Epoch 30: Train err: 0.362625, Train loss: 0.6423329524993896 |Validati
on err: 0.3785, Validation loss: 0.6439237017184496
Finished Training
Total time elapsed: 130.09 seconds
```

Originally, with the learning_rate = 0.01, the large_net model took 120.32 seconds to train.

Now, with the learning rate = 0.001, the large_net model takes 121.63 seconds to train.

Thus, the model takes longer time to train with a smaller learning rate (lr = 0.001).

In [22]:
```
plot_training_curve(get_model_name("large", batch_size=64, learning_rate
=0.001, epoch=29))
```



As the learning rate was lowered, there is less overfitting because the error on the training set is not as low compared to the error on the validation set.

Before, the model showed overfitting when the number of epoch increased because the error on the training set was low while the error on the validation set was large.

However, as the learning rate was lowered, it seems that the error has increased for both the training set and the validation set. Also, as the learning rate was lowered, it seems that the loss has increased for training set, but has decreased for the validation set.

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 [23]:
```
large_net = LargeNet()
train_net(large_net, 64, 0.1, 30)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.4295, Train loss: 0.67437779712677 |Validation er
r: 0.3595, Validation loss: 0.6350857093930244
Epoch 2: Train err: 0.36075, Train loss: 0.6411805458068848 |Validation
err: 0.3535, Validation loss: 0.6361209936439991
Epoch 3: Train err: 0.365125, Train loss: 0.6321813461780548 |Validatio
n err: 0.3385, Validation loss: 0.6056603882461786
Epoch 4: Train err: 0.352625, Train loss: 0.6233456182479858 |Validatio
n err: 0.3575, Validation loss: 0.6362800188362598
Epoch 5: Train err: 0.34075, Train loss: 0.6108013873100281 |Validation
err: 0.3305, Validation loss: 0.6064918786287308
Epoch 6: Train err: 0.323375, Train loss: 0.5921835997104645 |Validatio
n err: 0.317, Validation loss: 0.5967769594863057
Epoch 7: Train err: 0.3145, Train loss: 0.5817317583560944 |Validation
err: 0.3365, Validation loss: 0.6204487886279821
Epoch 8: Train err: 0.29825, Train loss: 0.5660300073623658 |Validation
err: 0.3285, Validation loss: 0.5983372200280428
Epoch 9: Train err: 0.290875, Train loss: 0.552809501171112 |Validation
err: 0.3315, Validation loss: 0.6084455158561468
Epoch 10: Train err: 0.278625, Train loss: 0.539032607793808 |Validatio
n err: 0.306, Validation loss: 0.5918631898239255
Epoch 11: Train err: 0.272375, Train loss: 0.5236025826931 |Validation
err: 0.33, Validation loss: 0.6430060230195522
Epoch 12: Train err: 0.267375, Train loss: 0.5220149435997009 |Validati
on err: 0.2925, Validation loss: 0.6413561534136534
Epoch 13: Train err: 0.266, Train loss: 0.5160510110855102 |Validation
err: 0.3125, Validation loss: 0.6349832843989134
Epoch 14: Train err: 0.24875, Train loss: 0.4951590054035187 |Validatio
n err: 0.3145, Validation loss: 0.7193072671070695
Epoch 15: Train err: 0.264625, Train loss: 0.519231944322586 |Validatio
n err: 0.314, Validation loss: 0.6381420725956559
Epoch 16: Train err: 0.252625, Train loss: 0.5020012385845184 |Validati
on err: 0.3225, Validation loss: 0.6551959458738565
Epoch 17: Train err: 0.23875, Train loss: 0.481714787364006 |Validation
err: 0.357, Validation loss: 0.6440742611885071
Epoch 18: Train err: 0.23375, Train loss: 0.47645506453514097 |Validati
on err: 0.3375, Validation loss: 0.6777342790737748
Epoch 19: Train err: 0.218125, Train loss: 0.45134368968009947 |Validat
ion err: 0.3445, Validation loss: 0.7232250478118658
Epoch 20: Train err: 0.217875, Train loss: 0.45516350817680357 |Validat
ion err: 0.3245, Validation loss: 0.6354950983077288
Epoch 21: Train err: 0.23275, Train loss: 0.47897080445289614 |Validati
on err: 0.3255, Validation loss: 0.8348110988736153
Epoch 22: Train err: 0.234875, Train loss: 0.4808810565471649 |Validati
on err: 0.334, Validation loss: 0.7191346418112516
Epoch 23: Train err: 0.21575, Train loss: 0.4563647754192352 |Validatio
n err: 0.316, Validation loss: 0.7083508176729083
Epoch 24: Train err: 0.2355, Train loss: 0.47718250966072084 |Validatio
n err: 0.327, Validation loss: 0.7333047650754452
Epoch 25: Train err: 0.22025, Train loss: 0.4583414270877838 |Validatio
n err: 0.3315, Validation loss: 0.7806987538933754
Epoch 26: Train err: 0.209625, Train loss: 0.4519626965522766 |Validati
on err: 0.3435, Validation loss: 0.7715998776257038
Epoch 27: Train err: 0.22175, Train loss: 0.4636160457134247 |Validatio
n err: 0.3215, Validation loss: 0.7656293725594878
Epoch 28: Train err: 0.219375, Train loss: 0.46314777398109436 |Validat
```

```
ion err: 0.348, Validation loss: 0.8202023077756166
Epoch 29: Train err: 0.235875, Train loss: 0.49053542733192446 |Validat
ion err: 0.326, Validation loss: 0.8150460105389357
Epoch 30: Train err: 0.22, Train loss: 0.4623157248497009 |Validation e
rr: 0.3165, Validation loss: 0.7585078496485949
Finished Training
Total time elapsed: 126.47 seconds
```

Originally, with the learning_rate = 0.01, the large_net model took 120.32 seconds to train.

Now, with the learning rate = 0.1, the large_net model takes 119.16 seconds to train.

Thus, the model takes shorter time to train with a larger learning rate (lr = 0.1).

In [24]:
```
plot_training_curve(get_model_name("large", batch_size=64, learning_rate
=0.1, epoch=29))
```

As the learning rate was increased, there is more overfitting because the error on the training set is lower compared to the error on the validation set.

Before, the model showed overfitting when the number of epoch increased because the error on the training set was low while the error on the validation set was large. Now, there is more difference in the training set and validation set.

Also, there seems to be more noise when the learning rate was increased.

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 [25]:
```
large_net = LargeNet()
train_net(large_net, 512, 0.01, 30)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.48175, Train loss: 0.6929379552602768 |Validation
err: 0.478, Validation loss: 0.6926824003458023
Epoch 2: Train err: 0.457625, Train loss: 0.6924104019999504 |Validatio
n err: 0.434, Validation loss: 0.6917425245046616
Epoch 3: Train err: 0.437, Train loss: 0.6916500590741634 |Validation e
rr: 0.4265, Validation loss: 0.6909129917621613
Epoch 4: Train err: 0.433625, Train loss: 0.6908449940383434 |Validatio
n err: 0.424, Validation loss: 0.6897870451211929
Epoch 5: Train err: 0.434, Train loss: 0.6896935552358627 |Validation e
rr: 0.424, Validation loss: 0.6881355047225952
Epoch 6: Train err: 0.438, Train loss: 0.688353206962347 |Validation er
r: 0.4285, Validation loss: 0.686011865735054
Epoch 7: Train err: 0.439375, Train loss: 0.6866871677339077 |Validatio
n err: 0.426, Validation loss: 0.6836968809366226
Epoch 8: Train err: 0.43525, Train loss: 0.6849770769476891 |Validation
err: 0.4115, Validation loss: 0.6814671903848648
Epoch 9: Train err: 0.42375, Train loss: 0.6832009293138981 |Validation
err: 0.414, Validation loss: 0.679591491818428
Epoch 10: Train err: 0.421, Train loss: 0.6811089366674423 |Validation
err: 0.416, Validation loss: 0.6771548539400101
Epoch 11: Train err: 0.420875, Train loss: 0.6794026419520378 |Validati
on err: 0.4095, Validation loss: 0.6748111099004745
Epoch 12: Train err: 0.41475, Train loss: 0.6768048219382763 |Validatio
n err: 0.412, Validation loss: 0.6737060546875
Epoch 13: Train err: 0.4105, Train loss: 0.6749702803790569 |Validation
err: 0.412, Validation loss: 0.6706101596355438
Epoch 14: Train err: 0.407125, Train loss: 0.6730880849063396 |Validati
on err: 0.4125, Validation loss: 0.6692148000001907
Epoch 15: Train err: 0.4005, Train loss: 0.6706806942820549 |Validation
err: 0.4105, Validation loss: 0.667252704501152
Epoch 16: Train err: 0.397625, Train loss: 0.6691771410405636 |Validati
on err: 0.405, Validation loss: 0.6649097055196762
Epoch 17: Train err: 0.393875, Train loss: 0.6675694733858109 |Validati
on err: 0.401, Validation loss: 0.6630224883556366
Epoch 18: Train err: 0.393, Train loss: 0.6648042872548103 |Validation
err: 0.3945, Validation loss: 0.6624014377593994
Epoch 19: Train err: 0.38625, Train loss: 0.662746611982584 |Validation
err: 0.388, Validation loss: 0.6597220152616501
Epoch 20: Train err: 0.38175, Train loss: 0.6596181839704514 |Validatio
n err: 0.4005, Validation loss: 0.6564337313175201
Epoch 21: Train err: 0.38575, Train loss: 0.6584899798035622 |Validatio
n err: 0.3885, Validation loss: 0.6586423963308334
Epoch 22: Train err: 0.378125, Train loss: 0.655123382806778 |Validatio
n err: 0.3855, Validation loss: 0.6528600305318832
Epoch 23: Train err: 0.372125, Train loss: 0.6508794128894806 |Validati
on err: 0.3835, Validation loss: 0.6497963815927505
Epoch 24: Train err: 0.37675, Train loss: 0.6488028429448605 |Validatio
n err: 0.385, Validation loss: 0.6474899500608444
Epoch 25: Train err: 0.368625, Train loss: 0.6445869170129299 |Validati
on err: 0.382, Validation loss: 0.6473268568515778
Epoch 26: Train err: 0.372625, Train loss: 0.6428566053509712 |Validati
on err: 0.3745, Validation loss: 0.6425703465938568
Epoch 27: Train err: 0.359375, Train loss: 0.6372117549180984 |Validati
on err: 0.379, Validation loss: 0.6397799849510193
Epoch 28: Train err: 0.35425, Train loss: 0.6337667480111122 |Validatio
```

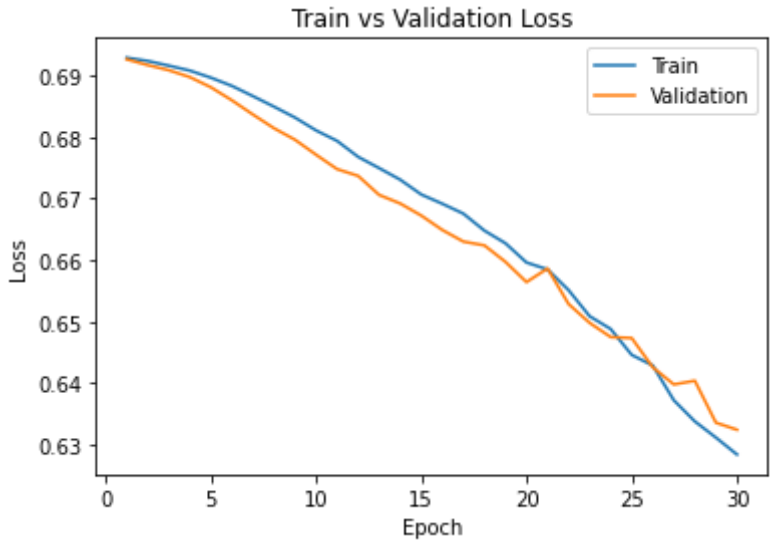
```
n err: 0.3695, Validation loss: 0.6403783112764359
Epoch 29: Train err: 0.3535, Train loss: 0.6311353109776974 |Validation
err: 0.366, Validation loss: 0.6335585117340088
Epoch 30: Train err: 0.353, Train loss: 0.6283832415938377 |Validation
err: 0.3675, Validation loss: 0.6324127316474915
Finished Training
Total time elapsed: 112.23 seconds
```

Originally, with the batch_size = 64, the large_net model took 120.32 seconds to train.

Now, with the batch_size = 512, the large_net model takes 103.43 seconds to train.

Thus, the model takes shorter time to train with a larger batch_size (batch_size = 512).

In [26]:
```
plot_training_curve(get_model_name("large", batch_size=512, learning_rat
e=0.01, epoch=29))
```



As the batch size was increased, there is less overfitting because the error on the training set is not as low compared to the error on the validation set.

Before, the model showed overfitting when the number of epoch increased because the error on the training set was low while the error on the validation set was large.

However, as the batch size was increased, it seems that the error has increased for both the training set and the validation set. Also, as the batch size was increased, it seems that the loss has increased for training set, but has decreased for the validation set.

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 [27]:
```
large_net = LargeNet()
train_net(large_net, 16, 0.01, 30)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.43175, Train loss: 0.6774994022846222 |Validation
err: 0.382, Validation loss: 0.6513170118331909
Epoch 2: Train err: 0.369, Train loss: 0.639639899969101 |Validation er
r: 0.3465, Validation loss: 0.6161113576889038
Epoch 3: Train err: 0.34375, Train loss: 0.6098222947120666 |Validation
err: 0.3325, Validation loss: 0.6260210764408112
Epoch 4: Train err: 0.314375, Train loss: 0.5849691489338875 |Validatio
n err: 0.34, Validation loss: 0.6044013917446136
Epoch 5: Train err: 0.301125, Train loss: 0.5689119303822517 |Validatio
n err: 0.3125, Validation loss: 0.576918310880661
Epoch 6: Train err: 0.281, Train loss: 0.5452213581204415 |Validation e
rr: 0.308, Validation loss: 0.5708447456359863
Epoch 7: Train err: 0.270875, Train loss: 0.5272981298565864 |Validatio
n err: 0.307, Validation loss: 0.5854293291568756
Epoch 8: Train err: 0.259375, Train loss: 0.5070905526578426 |Validatio
n err: 0.313, Validation loss: 0.5877130818367005
Epoch 9: Train err: 0.242375, Train loss: 0.4968344421982765 |Validatio
n err: 0.313, Validation loss: 0.5922425072193146
Epoch 10: Train err: 0.236375, Train loss: 0.4756101597249508 |Validati
on err: 0.297, Validation loss: 0.5718690166473389
Epoch 11: Train err: 0.222125, Train loss: 0.4599769461452961 |Validati
on err: 0.2975, Validation loss: 0.6376970833539963
Epoch 12: Train err: 0.211, Train loss: 0.4454492371380329 |Validation
err: 0.2995, Validation loss: 0.609202565908432
Epoch 13: Train err: 0.19875, Train loss: 0.4245421719551086 |Validatio
n err: 0.3075, Validation loss: 0.6494987765550614
Epoch 14: Train err: 0.18675, Train loss: 0.4007472907453775 |Validatio
n err: 0.3085, Validation loss: 0.6610016552209854
Epoch 15: Train err: 0.1645, Train loss: 0.3759974058121443 |Validation
err: 0.3105, Validation loss: 0.7106090537309646
Epoch 16: Train err: 0.16125, Train loss: 0.3591455406397581 |Validatio
n err: 0.3005, Validation loss: 0.7310364942550659
Epoch 17: Train err: 0.15775, Train loss: 0.3463234790861607 |Validatio
n err: 0.307, Validation loss: 0.7263009325265884
Epoch 18: Train err: 0.141625, Train loss: 0.32175366275012496 |Validat
ion err: 0.3195, Validation loss: 0.7913952842950821
Epoch 19: Train err: 0.13375, Train loss: 0.30618105667084455 |Validati
on err: 0.335, Validation loss: 0.8032052783966065
Epoch 20: Train err: 0.126625, Train loss: 0.3029071792438626 |Validati
on err: 0.32, Validation loss: 0.8106685240268707
Epoch 21: Train err: 0.12025, Train loss: 0.28682796490937473 |Validati
on err: 0.3205, Validation loss: 0.8259474284648896
Epoch 22: Train err: 0.1165, Train loss: 0.27489088076353074 |Validatio
n err: 0.352, Validation loss: 0.8937610774040222
Epoch 23: Train err: 0.104375, Train loss: 0.2467898527495563 |Validati
on err: 0.3315, Validation loss: 1.0021928198337555
Epoch 24: Train err: 0.101, Train loss: 0.23970085787773132 |Validation
err: 0.331, Validation loss: 1.1290796399116516
Epoch 25: Train err: 0.09575, Train loss: 0.23643119425699116 |Validati
on err: 0.3315, Validation loss: 1.1338514368534087
Epoch 26: Train err: 0.094125, Train loss: 0.2325953512713313 |Validati
on err: 0.3365, Validation loss: 1.1414263204336166
Epoch 27: Train err: 0.08425, Train loss: 0.21040759468451142 |Validati
on err: 0.3335, Validation loss: 1.1823678107261657
Epoch 28: Train err: 0.0825, Train loss: 0.20643112615589052 |Validatio
```

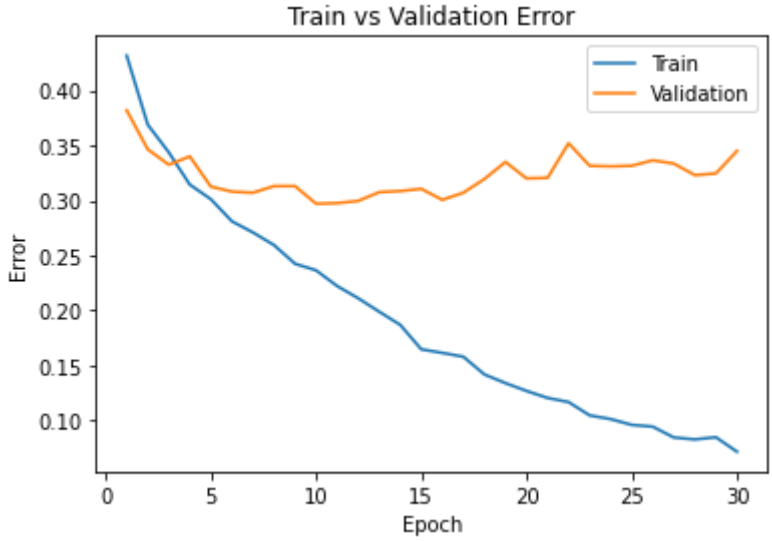
```
n err: 0.323, Validation loss: 1.266836181640625
Epoch 29: Train err: 0.0845, Train loss: 0.21273409337876364 |Validatio
n err: 0.3245, Validation loss: 1.406717705130577
Epoch 30: Train err: 0.071375, Train loss: 0.18387044295761734 |Validat
ion err: 0.345, Validation loss: 1.4871552000045776
Finished Training
Total time elapsed: 181.06 seconds
```

Originally, with the batch_size = 64, the large_net model took 120.32 seconds to train.

Now, with the batch_size = 16, the large_net model takes 182.18 seconds to train.

Thus, the model takes longer time to train with a smaller batch_size (batch_size = 16).

In [28]:
```
plot_training_curve(get_model_name("large", batch_size=16, learning_rate
=0.01, epoch=29))
```



As the batch size was decreased, there is more overfitting because the error on the training set is lower compared to the error on the validation set.

Before, the model showed overfitting when the number of epoch increased because the error on the training set was low while the error on the validation set was large. Now, there is more difference in the training set and validation set.

Also, there seems to be less noise when the batch size was decreased.

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.

Large_net, batch_size = 16, learning_rate = 0.001.

Becuase changing the learning_rate to 0.001 reduced the overfitting of the sets but the overall error increased for the validation, whereas changing the batch_size to 16 increased the overfitting, but decreased the overall error for the validation. Thus together, there was less overfitting and less validation error, improving the validation accuracy.

Part (b) - 1pt

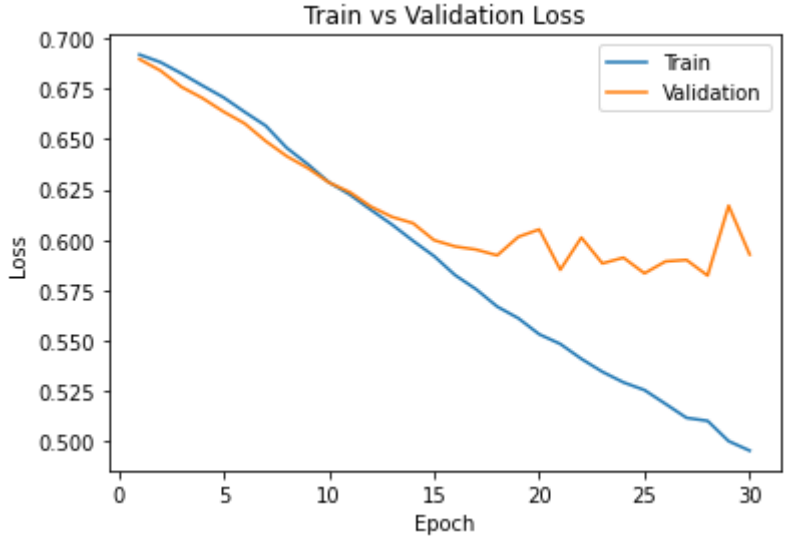
Train the model with the hyperparameters you chose in part(a), and include the training curve.

In [29]:
```
large_net = LargeNet()
train_net(large_net, 16, 0.001, 30)

plot_training_curve(get_model_name("large", batch_size=16, learning_rate
=0.001, epoch=29))
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.454875, Train loss: 0.6919687836170196 |Validatio
n err: 0.4335, Validation loss: 0.6896940703392029
Epoch 2: Train err: 0.441875, Train loss: 0.6881972500085831 |Validatio
n err: 0.4155, Validation loss: 0.6840243172645569
Epoch 3: Train err: 0.425, Train loss: 0.6826031126976013 |Validation e
rr: 0.4065, Validation loss: 0.6759178109169006
Epoch 4: Train err: 0.413375, Train loss: 0.676551971077919 |Validation
err: 0.411, Validation loss: 0.6703446621894836
Epoch 5: Train err: 0.403, Train loss: 0.6706955729722976 |Validation e
rr: 0.405, Validation loss: 0.6636172285079956
Epoch 6: Train err: 0.388375, Train loss: 0.6634951171875 |Validation e
rr: 0.388, Validation loss: 0.6577684187889099
Epoch 7: Train err: 0.38325, Train loss: 0.6566604214906693 |Validation
err: 0.387, Validation loss: 0.6490317993164062
Epoch 8: Train err: 0.372625, Train loss: 0.6456744936108589 |Validatio
n err: 0.3795, Validation loss: 0.6415270042419433
Epoch 9: Train err: 0.36425, Train loss: 0.637563487291336 |Validation
err: 0.362, Validation loss: 0.6356466286182404
Epoch 10: Train err: 0.353375, Train loss: 0.6286305035948754 |Validati
on err: 0.3565, Validation loss: 0.6284913148880005
Epoch 11: Train err: 0.348875, Train loss: 0.6224352505207061 |Validati
on err: 0.343, Validation loss: 0.6236779315471649
Epoch 12: Train err: 0.3395, Train loss: 0.6149048793315888 |Validation
err: 0.3335, Validation loss: 0.6165166666507721
Epoch 13: Train err: 0.337625, Train loss: 0.6076011454463005 |Validati
on err: 0.3345, Validation loss: 0.6113973023891449
Epoch 14: Train err: 0.326, Train loss: 0.5995380475521087 |Validation
err: 0.332, Validation loss: 0.6082773017883301
Epoch 15: Train err: 0.318625, Train loss: 0.5919455865621567 |Validati
on err: 0.3175, Validation loss: 0.5998473017215729
Epoch 16: Train err: 0.31475, Train loss: 0.5824866974949837 |Validatio
n err: 0.3165, Validation loss: 0.596711287021637
Epoch 17: Train err: 0.304625, Train loss: 0.5754690542817116 |Validati
on err: 0.305, Validation loss: 0.5951672253608704
Epoch 18: Train err: 0.301, Train loss: 0.5668730340003967 |Validation
err: 0.318, Validation loss: 0.5922847588062287
Epoch 19: Train err: 0.295125, Train loss: 0.5611101251840591 |Validati
on err: 0.332, Validation loss: 0.6014690661430359
Epoch 20: Train err: 0.2875, Train loss: 0.5531035642623902 |Validation
err: 0.312, Validation loss: 0.605187665939331
Epoch 21: Train err: 0.289375, Train loss: 0.5483110781908035 |Validati
on err: 0.304, Validation loss: 0.5851338193416595
Epoch 22: Train err: 0.280125, Train loss: 0.5409212954640389 |Validati
on err: 0.3235, Validation loss: 0.6011423192024231
Epoch 23: Train err: 0.276875, Train loss: 0.534481340944767 |Validatio
n err: 0.303, Validation loss: 0.5883027715682984
Epoch 24: Train err: 0.27025, Train loss: 0.5292306969761849 |Validatio
n err: 0.297, Validation loss: 0.5911655931472778
Epoch 25: Train err: 0.263875, Train loss: 0.5254164955615998 |Validati
on err: 0.294, Validation loss: 0.5833725001811981
Epoch 26: Train err: 0.2605, Train loss: 0.5185559968054294 |Validation
err: 0.3065, Validation loss: 0.5893082580566407
Epoch 27: Train err: 0.2565, Train loss: 0.5116140705645085 |Validation
err: 0.288, Validation loss: 0.5900122969150543
Epoch 28: Train err: 0.259125, Train loss: 0.5101234393119812 |Validati
```

```
on err: 0.3015, Validation loss: 0.582257523059845
Epoch 29: Train err: 0.247875, Train loss: 0.5000545628666878 |Validati
on err: 0.3205, Validation loss: 0.616970046043396
Epoch 30: Train err: 0.248875, Train loss: 0.49533707863092424 |Validat
ion err: 0.2995, Validation loss: 0.5927074880599975
Finished Training
Total time elapsed: 180.75 seconds
```



Part (c) - 2pt

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

Large_net, batch_size = 512, learning_rate = 0.1.

Becuase changing the batch_size to 512 reduced the overfitting of the sets but the overall error increased for the validation, whereas changing the learning_rate to 0.1 increased the overfitting, but decreased the overall error for the validation. Thus together, there was less overfitting and less validation error, improving the validation accuracy.

Part (d) - 1pt

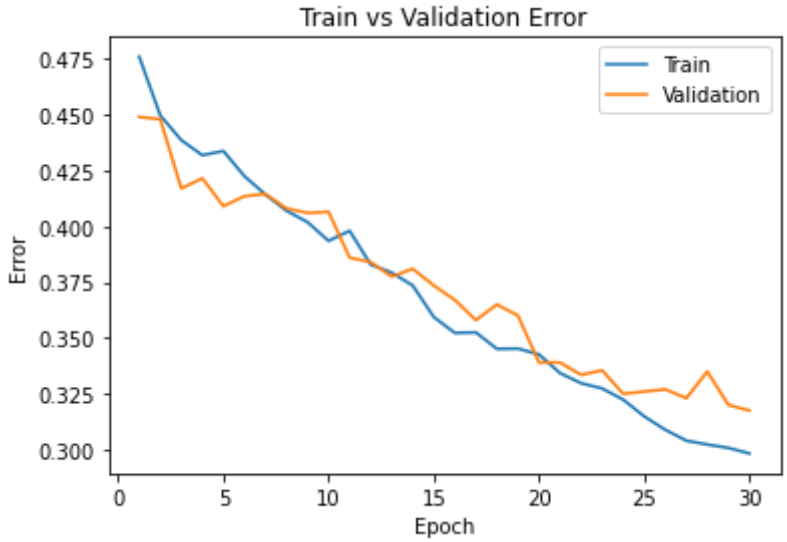
Train the model with the hyperparameters you chose in part(c), and include the training curve.

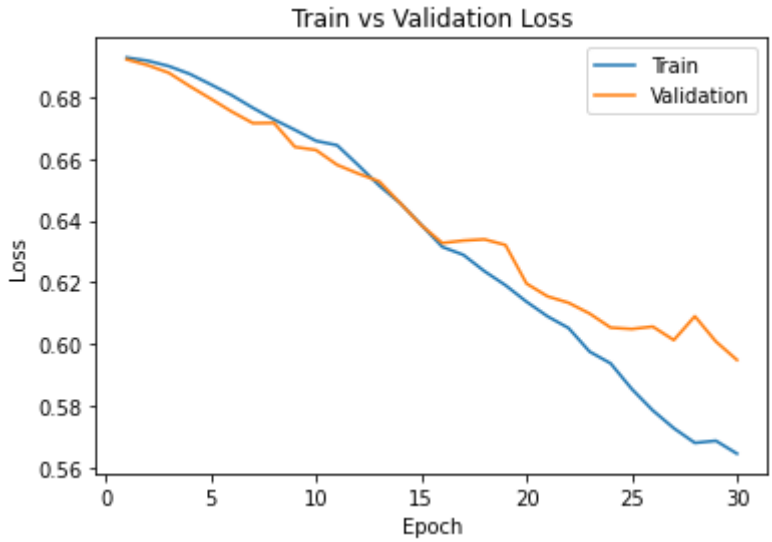
In [33]:
```
large_net = LargeNet()
train_net(large_net, 512, 0.1, 30)

plot_training_curve(get_model_name("large", batch_size=512, learning_rat
e=0.1, epoch=29))
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.477875, Train loss: 0.6929536536335945 |Validatio
n err: 0.476, Validation loss: 0.687719076871872
Epoch 2: Train err: 0.44975, Train loss: 0.6866156831383705 |Validation
err: 0.4355, Validation loss: 0.6788475513458252
Epoch 3: Train err: 0.417375, Train loss: 0.677769310772419 |Validation
err: 0.406, Validation loss: 0.6653236448764801
Epoch 4: Train err: 0.402, Train loss: 0.6664926782250404 |Validation e
rr: 0.3945, Validation loss: 0.658050611615181
Epoch 5: Train err: 0.3855, Train loss: 0.6551658473908901 |Validation
err: 0.3675, Validation loss: 0.6423593163490295
Epoch 6: Train err: 0.365, Train loss: 0.6404123604297638 |Validation e
rr: 0.364, Validation loss: 0.6320077627897263
Epoch 7: Train err: 0.356375, Train loss: 0.6332094147801399 |Validatio
n err: 0.335, Validation loss: 0.6137653887271881
Epoch 8: Train err: 0.339625, Train loss: 0.6179096810519695 |Validatio
n err: 0.3465, Validation loss: 0.6287042647600174
Epoch 9: Train err: 0.326, Train loss: 0.6031752899289131 |Validation e
rr: 0.332, Validation loss: 0.6101419627666473
Epoch 10: Train err: 0.315125, Train loss: 0.5840605087578297 |Validati
on err: 0.328, Validation loss: 0.5989724844694138
Epoch 11: Train err: 0.29225, Train loss: 0.5636394135653973 |Validatio
n err: 0.331, Validation loss: 0.6005314588546753
Epoch 12: Train err: 0.284625, Train loss: 0.5512890703976154 |Validati
on err: 0.319, Validation loss: 0.5937991738319397
Epoch 13: Train err: 0.277125, Train loss: 0.5355592481791973 |Validati
on err: 0.318, Validation loss: 0.5910229384899139
Epoch 14: Train err: 0.264625, Train loss: 0.5229382142424583 |Validati
on err: 0.33, Validation loss: 0.6232402473688126
Epoch 15: Train err: 0.264875, Train loss: 0.5216288287192583 |Validati
on err: 0.321, Validation loss: 0.6012101322412491
Epoch 16: Train err: 0.251375, Train loss: 0.5069038551300764 |Validati
on err: 0.3195, Validation loss: 0.612896278500557
Epoch 17: Train err: 0.246875, Train loss: 0.4984983056783676 |Validati
on err: 0.325, Validation loss: 0.6493691951036453
Epoch 18: Train err: 0.23375, Train loss: 0.4824167527258396 |Validatio
n err: 0.3135, Validation loss: 0.6138880699872971
Epoch 19: Train err: 0.227, Train loss: 0.4710276871919632 |Validation
err: 0.318, Validation loss: 0.6219888031482697
Epoch 20: Train err: 0.229, Train loss: 0.4701010175049305 |Validation
err: 0.3225, Validation loss: 0.6640109866857529
Epoch 21: Train err: 0.205875, Train loss: 0.4364516492933035 |Validati
on err: 0.3185, Validation loss: 0.6444135010242462
Epoch 22: Train err: 0.1935, Train loss: 0.41234126314520836 |Validatio
n err: 0.3315, Validation loss: 0.6462982892990112
Epoch 23: Train err: 0.186625, Train loss: 0.40691872499883175 |Validat
ion err: 0.3215, Validation loss: 0.693313717842102
Epoch 24: Train err: 0.17825, Train loss: 0.3924157600849867 |Validatio
n err: 0.323, Validation loss: 0.6848430931568146
Epoch 25: Train err: 0.168875, Train loss: 0.3688964154571295 |Validati
on err: 0.334, Validation loss: 0.749558687210083
Epoch 26: Train err: 0.15375, Train loss: 0.34573087841272354 |Validati
on err: 0.3295, Validation loss: 0.7886755615472794
Epoch 27: Train err: 0.14025, Train loss: 0.3199676685035229 |Validatio
n err: 0.3295, Validation loss: 0.8377431780099869
Epoch 28: Train err: 0.14, Train loss: 0.3199429716914892 |Validation e
```

```
rr: 0.33, Validation loss: 0.8318142890930176
Epoch 29: Train err: 0.13275, Train loss: 0.2944728322327137 |Validatio
n err: 0.339, Validation loss: 0.9227322489023209
Epoch 30: Train err: 0.11175, Train loss: 0.27146794740110636 |Validati
on err: 0.3245, Validation loss: 0.9187549650669098
Finished Training
Total time elapsed: 111.18 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 [34]:
```
net = LargeNet()
model_path = get_model_name(net.name, batch_size=16, learning_rate=0.001
, epoch=27)
state = torch.load(model_path)
net.load_state_dict(state)
```

Out[34]:
```
<All keys matched successfully>
```

Part (b) - 2pt

Justify your choice of model from part (a).

Large_net, batch_size = 16, learning_rate = 0.001, epoch = 27.

Becuase changing the learning_rate to 0.001 reduced the overfitting of the sets but the overall error increased for the validation, whereas changing the batch_size to 16 increased the overfitting, but decreased the overall error for the validation. Thus together, there was less overfitting and less validation error, improving the validation accuracy.

At Epoch = 27, there is the lowest validation error of 0.288.

In conclusion, I chose this model because it does not have overfitting, and the error decreased to 0.288 at the lowest, which both implies good validation accuracy.

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 [40]:
```
# If you use the `evaluate` function provided in part 0, you will need t
o
# set batch_size > 1
train_loader, val_loader, test_loader, classes = get_data_loader(
    target_classes=["cat", "dog"],
    batch_size=64)

evaluate(net, test_loader, nn.BCEWithLogitsLoss())
```

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

Out[40]:
```
(0.2855, 0.5526062799617648)
```

Test Classification Error is 0.2855.

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 [41]: evaluate(net, val_loader, nn.BCEWithLogitsLoss())
```

```
Out[41]: (0.3015, 0.58578981179744)
```

I would expect the test error to be higher than validation error because the best model for validation was chosen by adjusting hyperparameters, whereas the test model was not known for us to adjust to it. However, my test error was lower than validation error.

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?

We only used the test data set at the very end to test the model that we have already trained. It is important to use as little test data as possible because it is uesd only to verifying the results, whereas the other training data and validation data is used to train and adjust the hyperparameters of the model.

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 [44]:
```
class Cat_Dog(nn.Module):
    def __init__(self):
        super(Cat_Dog, self).__init__()
        self.name = "CatDog"
        self.layer1 = nn.Linear(3 * 32 * 32, 30)
        self.layer2 = nn.Linear(30, 1)
        #self.layer3 = nn.Linear(1500, 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 = F.relu(activation2)

        #activation3 = self.layer3(activation2)
        return activation2.squeeze()

cats_dogs = Cat_Dog()

train_net(cats_dogs, 16, 0.001, 27)


train_loader, val_loader, test_loader, classes = get_data_loader(
    target_classes=["cat", "dog"],
    batch_size=64)

test_error, test_loss = evaluate(cats_dogs, test_loader, nn.BCEWithLogit
sLoss())

print("Test err: ", test_error)
print("Test loss: ", test_loss)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.413375, Train loss: 0.6650850504040718 |Validatio
n err: 0.3925, Validation loss: 0.6526617212295532
Epoch 2: Train err: 0.37525, Train loss: 0.6436380042433739 |Validation
err: 0.392, Validation loss: 0.6516472935676575
Epoch 3: Train err: 0.3665, Train loss: 0.6341393489837647 |Validation
err: 0.382, Validation loss: 0.645181411743164
Epoch 4: Train err: 0.34975, Train loss: 0.6240796703100204 |Validation
err: 0.3945, Validation loss: 0.6618771114349365
Epoch 5: Train err: 0.342, Train loss: 0.6170741337537765 |Validation e
rr: 0.378, Validation loss: 0.6427467465400696
Epoch 6: Train err: 0.329875, Train loss: 0.60537016248703 |Validation
err: 0.3945, Validation loss: 0.6619392056465149
Epoch 7: Train err: 0.318125, Train loss: 0.596566174030304 |Validation
err: 0.375, Validation loss: 0.6482871866226196
Epoch 8: Train err: 0.314125, Train loss: 0.5859173473119735 |Validatio
n err: 0.3765, Validation loss: 0.6474692990779877
Epoch 9: Train err: 0.301375, Train loss: 0.5767931064367294 |Validatio
n err: 0.376, Validation loss: 0.6500314214229583
Epoch 10: Train err: 0.295375, Train loss: 0.5651416803598404 |Validati
on err: 0.3795, Validation loss: 0.6496774654388427
Epoch 11: Train err: 0.28925, Train loss: 0.5563273149132728 |Validatio
n err: 0.374, Validation loss: 0.6491571776866912
Epoch 12: Train err: 0.278125, Train loss: 0.5470306626558303 |Validati
on err: 0.3675, Validation loss: 0.6581876451969146
Epoch 13: Train err: 0.2715, Train loss: 0.5353209666013717 |Validation
err: 0.3775, Validation loss: 0.686122722864151
Epoch 14: Train err: 0.263625, Train loss: 0.5289770259857177 |Validati
on err: 0.366, Validation loss: 0.666825897693634
Epoch 15: Train err: 0.252375, Train loss: 0.5140348685979843 |Validati
on err: 0.3625, Validation loss: 0.6847940359115601
Epoch 16: Train err: 0.24875, Train loss: 0.5066808835268021 |Validatio
n err: 0.372, Validation loss: 0.6760222225189209
Epoch 17: Train err: 0.237875, Train loss: 0.49500395309925077 |Validat
ion err: 0.365, Validation loss: 0.6711816058158875
Epoch 18: Train err: 0.230625, Train loss: 0.48216327315568924 |Validat
ion err: 0.3645, Validation loss: 0.6872826001644134
Epoch 19: Train err: 0.2205, Train loss: 0.4732912865281105 |Validation
err: 0.3805, Validation loss: 0.7050547659397125
Epoch 20: Train err: 0.21175, Train loss: 0.4610502922832966 |Validatio
n err: 0.3525, Validation loss: 0.712874340057373
Epoch 21: Train err: 0.200875, Train loss: 0.4509826074838638 |Validati
on err: 0.369, Validation loss: 0.6958276438713074
Epoch 22: Train err: 0.20425, Train loss: 0.4426146577000618 |Validatio
n err: 0.3765, Validation loss: 0.7397264404296875
Epoch 23: Train err: 0.204, Train loss: 0.4330629488825798 |Validation
err: 0.372, Validation loss: 0.7322707440853119
Epoch 24: Train err: 0.1875, Train loss: 0.4181140091717243 |Validation
err: 0.3555, Validation loss: 0.7246655232906342
Epoch 25: Train err: 0.18125, Train loss: 0.4066800771653652 |Validatio
n err: 0.365, Validation loss: 0.728918039560318
Epoch 26: Train err: 0.173375, Train loss: 0.39572204813361167 |Validat
ion err: 0.366, Validation loss: 0.7570700402259827
Epoch 27: Train err: 0.176125, Train loss: 0.3885623929202557 |Validati
on err: 0.374, Validation loss: 0.7624980766773224
Finished Training
```

```
Total time elapsed: 129.86 seconds
Files already downloaded and verified
Files already downloaded and verified
Test err:  0.379
Test loss:  0.7712736614048481
```