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/1u2vjFhACnJOVpsQce9nLZaOZFcJHpVrs?usp=sharing (https://colab.research.google.com/drive/1u2vjFhACnJOVpsQce9nLZaOZFcJHpVrs?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_i
ndices[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 (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 ./d
ata/cifar-10-python.tar.gz


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

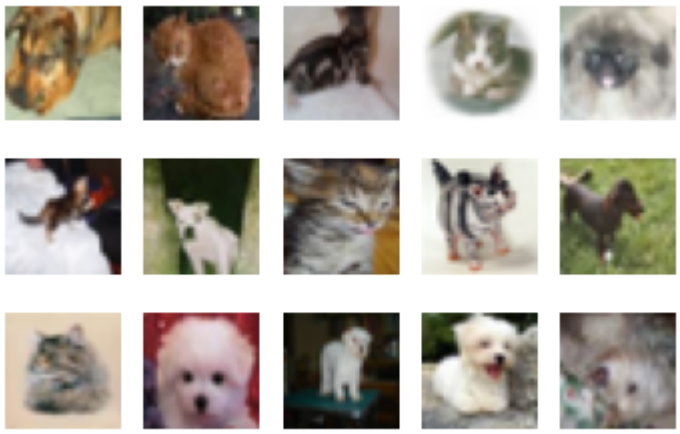
Part (a) -- 1 pt

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 [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 [6]:

```
print("Number of trainning examples: ",len(train_loader))
print("Number of validation examples: ",len(val_loader))
print("Number of test examples: ",len(test_loader))
```

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

Part (c) -- 3pt

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

A validation data set is used to verify the performance of our trained model, it is sepearate from the test data and trainning data to avoid the biases of the model. By adujusting the model using validation set we can better understand the real performance of our model and tune the hyperparameters. If we use training set instead, the model may be overfitted to the feeding data and has a bad performance when it encounter the new data set.

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?)

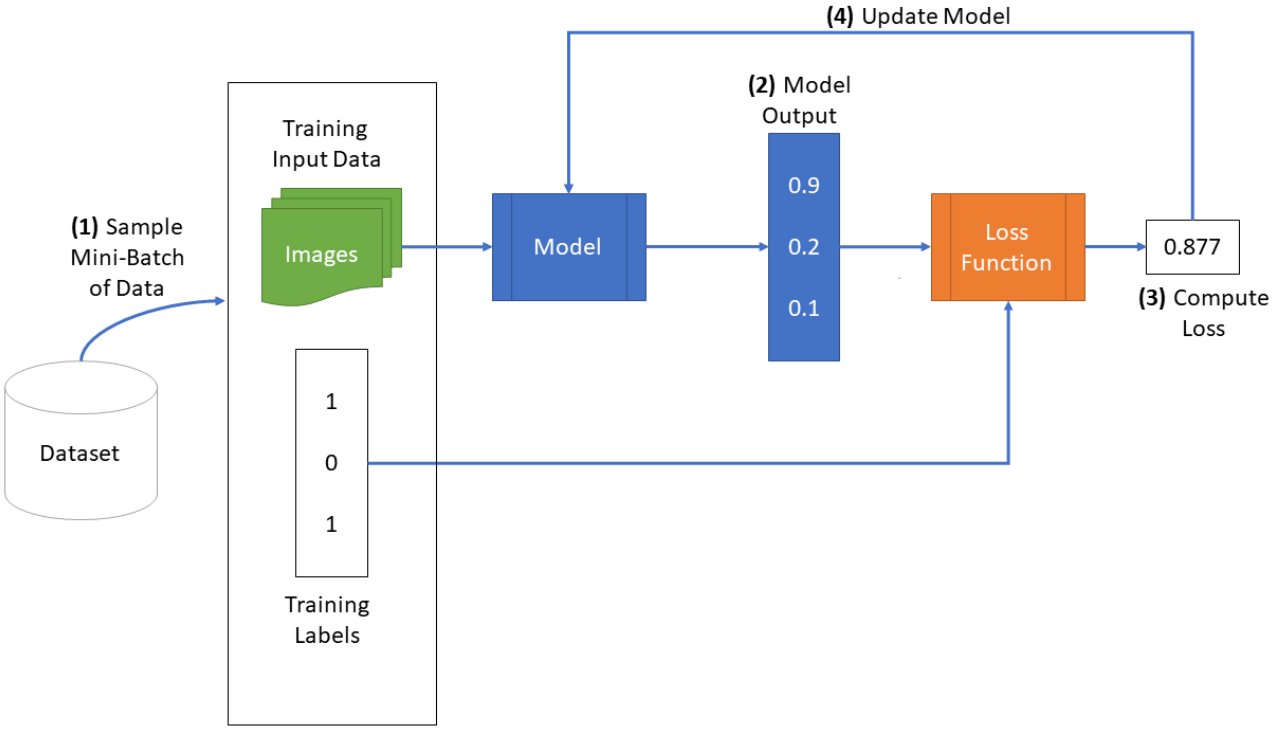
In [12]:

```
size_s = 0
size_l = 0
for param in small_net.parameters():
    print(param.shape)
    size_s += param.numel()
print("The total number of parameters in small_net is: ", size_s)

for param in large_net.parameters():
    print(param.shape)
    size_l += param.numel()
print("The total number of parameters in small_net is: ", size_l)
```

```
torch.Size([5, 3, 3, 3])
torch.Size([5])
torch.Size([1, 245])
torch.Size([1])
The total number of parameters in small_net is:  386
torch.Size([5, 3, 5, 5])
torch.Size([5])
torch.Size([10, 5, 5, 5])
torch.Size([10])
torch.Size([32, 250])
torch.Size([32])
torch.Size([1, 32])
torch.Size([1])
The total number of parameters in small_net is:  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 [31]:

```
def train_net(net, path, 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)
            model_path = "/".join([path, model_path])
            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`?

The default values of batch_size is 64, learning_rate is 0.01, num_epochs is 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 [35]:

```
from google.colab import drive
drive.mount('/content/gdrive')
small_net=SmallNet()
train_net(small_net, "/content/gdrive/My Drive/Colab Notebooks/APS360_labs/Lab2_Part2c"
, num_epochs= 5)
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount,
call drive.mount("/content/gdrive", force_remount=True).
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.446375, Train loss: 0.6813716783523559 |Validation e
rr: 0.3865, Validation loss: 0.6602997500449419
Epoch 2: Train err: 0.37325, Train loss: 0.6497629323005676 |Validation er
r: 0.3845, Validation loss: 0.6575995869934559
Epoch 3: Train err: 0.359875, Train loss: 0.6388978385925292 |Validation e
rr: 0.3495, Validation loss: 0.6291275043040514
Epoch 4: Train err: 0.346375, Train loss: 0.6246587996482849 |Validation e
rr: 0.356, Validation loss: 0.6221408396959305
Epoch 5: Train err: 0.334375, Train loss: 0.6153830280303955 |Validation e
rr: 0.3275, Validation loss: 0.6188967823982239
Finished Training
Total time elapsed: 18.70 seconds
```

The files are: model_small_bs64_lr0.01_epoch0

model_small_bs64_lr0.01_epoch1

model_small_bs64_lr0.01_epoch2

model_small_bs64_lr0.01_epoch3

model_small_bs64_lr0.01_epoch4

model_small_bs64_lr0.01_epoch4_train_err.csv

model_small_bs64_lr0.01_epoch4_train_loss.csv

model_small_bs64_lr0.01_epoch4_val_err.csv

model_small_bs64_lr0.01_epoch4_val_loss.csv

The files name end with epoch0/1/2/3/4 are stored the model of epochs 0 to 4 respectively.

The other 4 .csv files contains all the information about trainning error, trainning loss, validation error and validation loss for the models above.

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 [ ]:

```
# 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')
```

In [36]:

```
small_net = SmallNet()
train_net(small_net, "/content/gdrive/My Drive/Colab Notebooks/APS360_labs/Lab2_Part2d_
Small")
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.446375, Train loss: 0.6813716783523559 |Validation e
rr: 0.3865, Validation loss: 0.6602997500449419
Epoch 2: Train err: 0.37325, Train loss: 0.6497629323005676 |Validation er
r: 0.3845, Validation loss: 0.6575995869934559
Epoch 3: Train err: 0.359875, Train loss: 0.6388978385925292 |Validation e
rr: 0.3495, Validation loss: 0.6291275043040514
Epoch 4: Train err: 0.346375, Train loss: 0.6246587996482849 |Validation e
rr: 0.356, Validation loss: 0.6221408396959305
Epoch 5: Train err: 0.334375, Train loss: 0.6153830280303955 |Validation e
rr: 0.3275, Validation loss: 0.6188967823982239
Epoch 6: Train err: 0.318, Train loss: 0.6036732516288758 |Validation err:
0.339, Validation loss: 0.6094125052914023
Epoch 7: Train err: 0.315625, Train loss: 0.5944745948314667 |Validation e
rr: 0.329, Validation loss: 0.5974238961935043
Epoch 8: Train err: 0.3085, Train loss: 0.5829453563690186 |Validation er
r: 0.3085, Validation loss: 0.5885121468454599
Epoch 9: Train err: 0.302, Train loss: 0.5805657277107239 |Validation err:
0.3115, Validation loss: 0.5845186104997993
Epoch 10: Train err: 0.29975, Train loss: 0.573062111377716 |Validation er
r: 0.309, Validation loss: 0.5785001656040549
Epoch 11: Train err: 0.287375, Train loss: 0.5632161114215851 |Validation
err: 0.314, Validation loss: 0.5821095015853643
Epoch 12: Train err: 0.292125, Train loss: 0.5567435595989227 |Validation
err: 0.3115, Validation loss: 0.5860895598307252
Epoch 13: Train err: 0.2885, Train loss: 0.5562505607604981 |Validation er
r: 0.306, Validation loss: 0.5769414035603404
Epoch 14: Train err: 0.280375, Train loss: 0.5473350758552551 |Validation
err: 0.3115, Validation loss: 0.5721263345330954
Epoch 15: Train err: 0.285, Train loss: 0.5481121215820313 |Validation er
r: 0.305, Validation loss: 0.5623639700934291
Epoch 16: Train err: 0.2915, Train loss: 0.5539557900428772 |Validation er
r: 0.3135, Validation loss: 0.5774335078895092
Epoch 17: Train err: 0.28075, Train loss: 0.5475348830223083 |Validation e
rr: 0.2995, Validation loss: 0.5680588381364942
Epoch 18: Train err: 0.279625, Train loss: 0.5440063354969025 |Validation
err: 0.319, Validation loss: 0.576342330314219
Epoch 19: Train err: 0.27575, Train loss: 0.5402116534709931 |Validation e
rr: 0.3295, Validation loss: 0.606647988781333
Epoch 20: Train err: 0.2715, Train loss: 0.5385935208797454 |Validation er
r: 0.298, Validation loss: 0.5778946885839105
Epoch 21: Train err: 0.27575, Train loss: 0.540246558189392 |Validation er
r: 0.302, Validation loss: 0.5672952607274055
Epoch 22: Train err: 0.279, Train loss: 0.5399930019378663 |Validation er
r: 0.2895, Validation loss: 0.5702174408361316
Epoch 23: Train err: 0.27325, Train loss: 0.5354620461463928 |Validation e
rr: 0.303, Validation loss: 0.5667499387636781
Epoch 24: Train err: 0.27275, Train loss: 0.5359286315441132 |Validation e
rr: 0.301, Validation loss: 0.5878297919407487
Epoch 25: Train err: 0.27325, Train loss: 0.5346703794002533 |Validation e
rr: 0.297, Validation loss: 0.563475382514298
Epoch 26: Train err: 0.27025, Train loss: 0.5316284673213959 |Validation e
rr: 0.2985, Validation loss: 0.5694020707160234
Epoch 27: Train err: 0.270375, Train loss: 0.5298305144309997 |Validation
err: 0.301, Validation loss: 0.578824263997376
Epoch 28: Train err: 0.269625, Train loss: 0.5351403400897979 |Validation
err: 0.3005, Validation loss: 0.5655373437330127
Epoch 29: Train err: 0.271875, Train loss: 0.5319398436546325 |Validation
err: 0.2955, Validation loss: 0.5849009975790977
Epoch 30: Train err: 0.270875, Train loss: 0.5373601081371308 |Validation
```

```
err: 0.3175, Validation loss: 0.5815494349226356
Finished Training
Total time elapsed: 116.07 seconds
```

In [37]:

```
large_net = LargeNet()
train_net(large_net, "/content/gdrive/My Drive/Colab Notebooks/APS360_labs/Lab2_Part2d_
Large")
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.44475, Train loss: 0.6900203123092651 |Validation er
r: 0.4285, Validation loss: 0.6807542946189642
Epoch 2: Train err: 0.4195, Train loss: 0.67819615650177 |Validation err:
0.413, Validation loss: 0.6741204150021076
Epoch 3: Train err: 0.39875, Train loss: 0.6658886175155639 |Validation er
r: 0.3925, Validation loss: 0.6518177818506956
Epoch 4: Train err: 0.373875, Train loss: 0.6490470728874207 |Validation e
rr: 0.406, Validation loss: 0.6635930277407169
Epoch 5: Train err: 0.35375, Train loss: 0.6330237889289856 |Validation er
r: 0.353, Validation loss: 0.6284337677061558
Epoch 6: Train err: 0.339125, Train loss: 0.6160062110424042 |Validation e
rr: 0.34, Validation loss: 0.6150468923151493
Epoch 7: Train err: 0.32625, Train loss: 0.6001663441658021 |Validation er
r: 0.335, Validation loss: 0.6086486000567675
Epoch 8: Train err: 0.313875, Train loss: 0.5828366029262543 |Validation e
rr: 0.3325, Validation loss: 0.5983812175691128
Epoch 9: Train err: 0.3085, Train loss: 0.5771152818202973 |Validation er
r: 0.315, Validation loss: 0.5944901742041111
Epoch 10: Train err: 0.2935, Train loss: 0.5622252209186553 |Validation er
r: 0.3155, Validation loss: 0.5953319733962417
Epoch 11: Train err: 0.284375, Train loss: 0.5485934205055237 |Validation
err: 0.3155, Validation loss: 0.611855155788362
Epoch 12: Train err: 0.2785, Train loss: 0.5407600071430206 |Validation er
r: 0.31, Validation loss: 0.5998768676072359
Epoch 13: Train err: 0.278125, Train loss: 0.5309763443470001 |Validation
err: 0.297, Validation loss: 0.583570459857583
Epoch 14: Train err: 0.26275, Train loss: 0.5173734092712402 |Validation e
rr: 0.2985, Validation loss: 0.5952338296920061
Epoch 15: Train err: 0.2555, Train loss: 0.5114965040683747 |Validation er
r: 0.2975, Validation loss: 0.5921047143638134
Epoch 16: Train err: 0.249625, Train loss: 0.5027769944667816 |Validation
err: 0.3025, Validation loss: 0.6024623941630125
Epoch 17: Train err: 0.247125, Train loss: 0.49459292030334473 |Validation
err: 0.298, Validation loss: 0.5841752551496029
Epoch 18: Train err: 0.235, Train loss: 0.4815113615989685 |Validation er
r: 0.3055, Validation loss: 0.5952335279434919
Epoch 19: Train err: 0.234375, Train loss: 0.4761629197597504 |Validation
err: 0.3205, Validation loss: 0.6141199134290218
Epoch 20: Train err: 0.225875, Train loss: 0.4626086118221283 |Validation
err: 0.3085, Validation loss: 0.6088314475491643
Epoch 21: Train err: 0.219125, Train loss: 0.4528284652233124 |Validation
err: 0.287, Validation loss: 0.5983596304431558
Epoch 22: Train err: 0.214125, Train loss: 0.4414942805767059 |Validation
err: 0.3045, Validation loss: 0.6131745660677552
Epoch 23: Train err: 0.209125, Train loss: 0.4349266333580017 |Validation
err: 0.301, Validation loss: 0.6113477284088731
Epoch 24: Train err: 0.202625, Train loss: 0.4223342254161835 |Validation
err: 0.3195, Validation loss: 0.6655794447287917
Epoch 25: Train err: 0.194, Train loss: 0.4071682426929474 |Validation er
r: 0.3015, Validation loss: 0.6275045238435268
Epoch 26: Train err: 0.181, Train loss: 0.38940074408054354 |Validation er
r: 0.309, Validation loss: 0.662933480925858
Epoch 27: Train err: 0.1765, Train loss: 0.3803082858324051 |Validation er
r: 0.303, Validation loss: 0.6525407964363694
Epoch 28: Train err: 0.16925, Train loss: 0.37183564841747285 |Validation
err: 0.2975, Validation loss: 0.6434079697355628
Epoch 29: Train err: 0.1605, Train loss: 0.35386401546001434 |Validation e
rr: 0.319, Validation loss: 0.8143855566158891
Epoch 30: Train err: 0.1595, Train loss: 0.3514791972637176 |Validation er
```

```
r: 0.3025, Validation loss: 0.702620318159461
Finished Training
Total time elapsed: 126.17 seconds
```

As the results shown, trainning small_net used 116.07s and training large_net used 126.17s. It is obvious that trainning large_net takes a longer time since it contains much larger number of parameters compare to small_net.

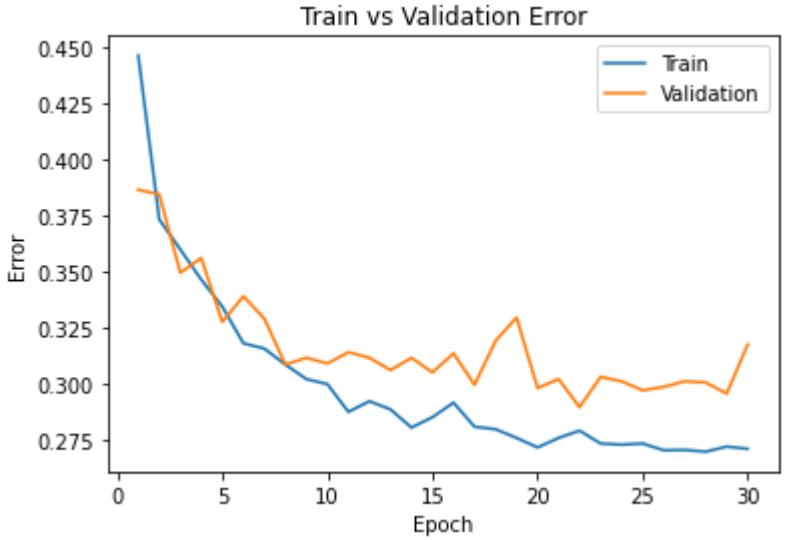
Part (e) - 2pt

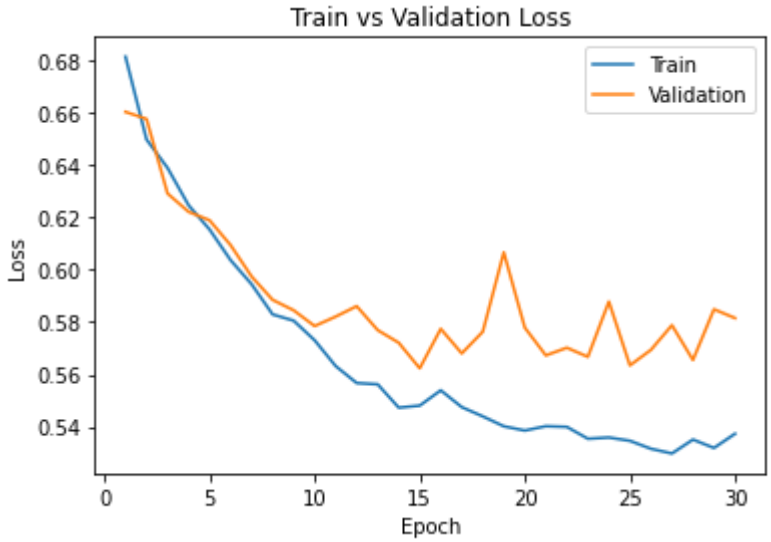
Use the function `plot_training_curve` to display the trajectory of the training/validation error and the training/validation loss. You will need to use the function `get_model_name` to generate the argument to the `plot_training_curve` function.

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

In [38]:

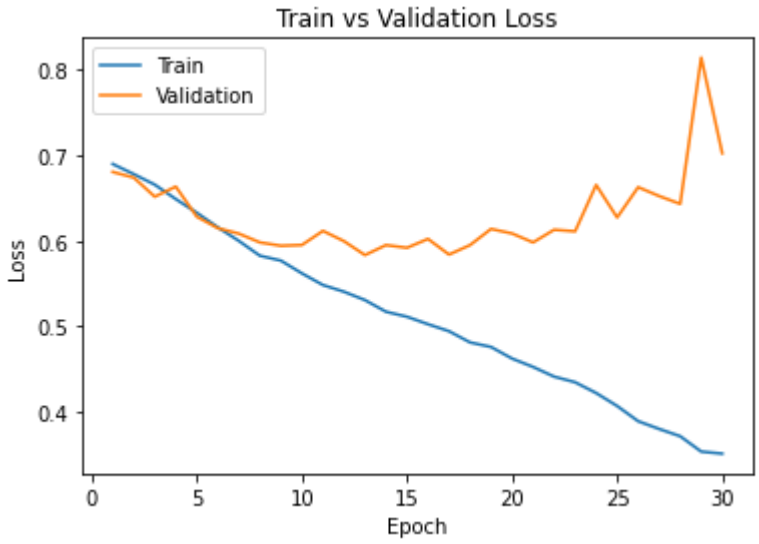
```
model_path = get_model_name("small", batch_size=64, learning_rate=0.01, epoch=29)
path = "/content/gdrive/My Drive/Colab Notebooks/APS360_labs/Lab2_Part2d_Small"
path_small = "/".join([path, model_path])
plot_training_curve(path_small)
```





In [39]:

```
model_path = get_model_name("large", batch_size=64, learning_rate=0.01, epoch=29)
path = "/content/gdrive/My Drive/Colab Notebooks/APS360_labs/Lab2_Part2d_Large"
path_large = "/".join([path, model_path])
plot_training_curve(path_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.

For small_net, underfitting occurs at smaller epochs (0-10), it can be observed that both trainning and validation error/loss decreases as the number of epoch increases. There is no significant overfitting occured in small_net.

For large_net, underfitting occurs at smaller epochs (0-10), it can be observed that both trainning and validation error/loss decreases as the number of epoch increases. Between 10-15 epoch, the large_net starts to become overfitting. It can be observed that the train error/loss decreases as the number of epoch increases, but the validation error starts to flattened and validation loss somehow increases as the number of epoch increases. And overall, the large_net result is less noisy than the small_net.

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 [40]:

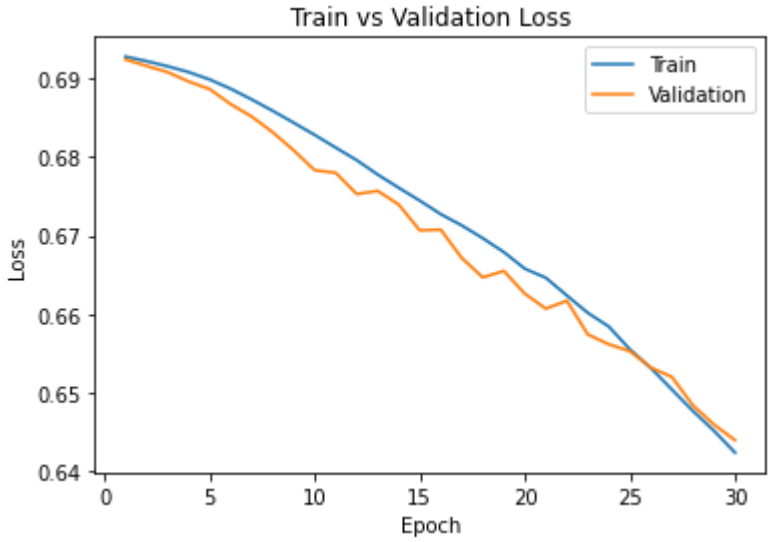
```
# Note: When we re-construct the model, we start the training
# with *random weights*. If we omit this code, the values of
# the weights will still be the previously trained values.
large_net = LargeNet()
train_net(large_net, "/content/gdrive/My Drive/Colab Notebooks/APS360_labs/Lab2_Part3_L
arge", learning_rate=0.001)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.47625, Train loss: 0.6928360013961792 |Validation er
r: 0.467, Validation loss: 0.6924686580896378
Epoch 2: Train err: 0.448625, Train loss: 0.6922589712142945 |Validation e
rr: 0.4305, Validation loss: 0.691649341955781
Epoch 3: Train err: 0.43575, Train loss: 0.6916067280769348 |Validation er
r: 0.4285, Validation loss: 0.690854424610734
Epoch 4: Train err: 0.43, Train loss: 0.690861343383789 |Validation err:
0.424, Validation loss: 0.6896595880389214
Epoch 5: Train err: 0.434125, Train loss: 0.6899195008277893 |Validation e
rr: 0.4195, Validation loss: 0.6886935643851757
Epoch 6: Train err: 0.43575, Train loss: 0.6887411961555481 |Validation er
r: 0.4195, Validation loss: 0.6867824867367744
Epoch 7: Train err: 0.437125, Train loss: 0.6873774147033691 |Validation e
rr: 0.4185, Validation loss: 0.6851982977241278
Epoch 8: Train err: 0.4375, Train loss: 0.6859278454780579 |Validation er
r: 0.412, Validation loss: 0.683199780061841
Epoch 9: Train err: 0.424375, Train loss: 0.6844058036804199 |Validation e
rr: 0.411, Validation loss: 0.6808880660682917
Epoch 10: Train err: 0.424, Train loss: 0.6828502931594849 |Validation er
r: 0.408, Validation loss: 0.6783502567559481
Epoch 11: Train err: 0.425375, Train loss: 0.6812348766326904 |Validation
err: 0.4125, Validation loss: 0.6780214440077543
Epoch 12: Train err: 0.42, Train loss: 0.6796319708824158 |Validation err:
0.4125, Validation loss: 0.6753159202635288
Epoch 13: Train err: 0.414875, Train loss: 0.6777918744087219 |Validation
err: 0.415, Validation loss: 0.6757059413939714
Epoch 14: Train err: 0.412375, Train loss: 0.6761112003326416 |Validation
err: 0.412, Validation loss: 0.6739734839648008
Epoch 15: Train err: 0.40925, Train loss: 0.674472680568695 |Validation er
r: 0.415, Validation loss: 0.6706762500107288
Epoch 16: Train err: 0.406375, Train loss: 0.6727448840141297 |Validation
err: 0.4105, Validation loss: 0.6707733049988747
Epoch 17: Train err: 0.4015, Train loss: 0.6713076601028443 |Validation er
r: 0.4045, Validation loss: 0.6671545393764973
Epoch 18: Train err: 0.3995, Train loss: 0.6696742882728577 |Validation er
r: 0.4055, Validation loss: 0.6646782550960779
Epoch 19: Train err: 0.40075, Train loss: 0.6679086356163025 |Validation e
rr: 0.396, Validation loss: 0.6655019577592611
Epoch 20: Train err: 0.392375, Train loss: 0.665787980556488 |Validation e
rr: 0.405, Validation loss: 0.6626011095941067
Epoch 21: Train err: 0.38975, Train loss: 0.6646300601959229 |Validation e
rr: 0.394, Validation loss: 0.660687854513526
Epoch 22: Train err: 0.388875, Train loss: 0.662373058795929 |Validation e
rr: 0.393, Validation loss: 0.6616998575627804
Epoch 23: Train err: 0.38425, Train loss: 0.6601516346931458 |Validation e
rr: 0.3975, Validation loss: 0.6573981791734695
Epoch 24: Train err: 0.382375, Train loss: 0.6584009389877319 |Validation
err: 0.386, Validation loss: 0.6561364810913801
Epoch 25: Train err: 0.37875, Train loss: 0.6554971766471863 |Validation e
rr: 0.388, Validation loss: 0.6552744228392839
Epoch 26: Train err: 0.376625, Train loss: 0.6531173253059387 |Validation
err: 0.3875, Validation loss: 0.6531743723899126
Epoch 27: Train err: 0.375, Train loss: 0.6503696331977844 |Validation er
r: 0.387, Validation loss: 0.6519789285957813
Epoch 28: Train err: 0.371375, Train loss: 0.6476435809135437 |Validation
err: 0.3875, Validation loss: 0.6483502741903067
Epoch 29: Train err: 0.368375, Train loss: 0.6451257643699646 |Validation
err: 0.3825, Validation loss: 0.6459067314863205
Epoch 30: Train err: 0.362625, Train loss: 0.6423329524993896 |Validation
```

```
err: 0.3785, Validation loss: 0.6439237017184496
Finished Training
Total time elapsed: 129.17 seconds
```

In [42]:

```
model_path = get_model_name("large", batch_size=64, learning_rate=0.001, epoch=29)
path = "/content/gdrive/My Drive/Colab Notebooks/APS360_labs/Lab2_Part3_Large"
path_large = "/".join([path, model_path])
plot_training_curve(path_large)
```



As the learning rate decreased from 0.01 to 0.001, the trainning time increases from 126.17s to 129.17s. As we decreased the learning rate, the validation error/loss curve is better fitted with the trainning error/loss curve, the overfitting is no longer happened compare to the default.

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 [43]:

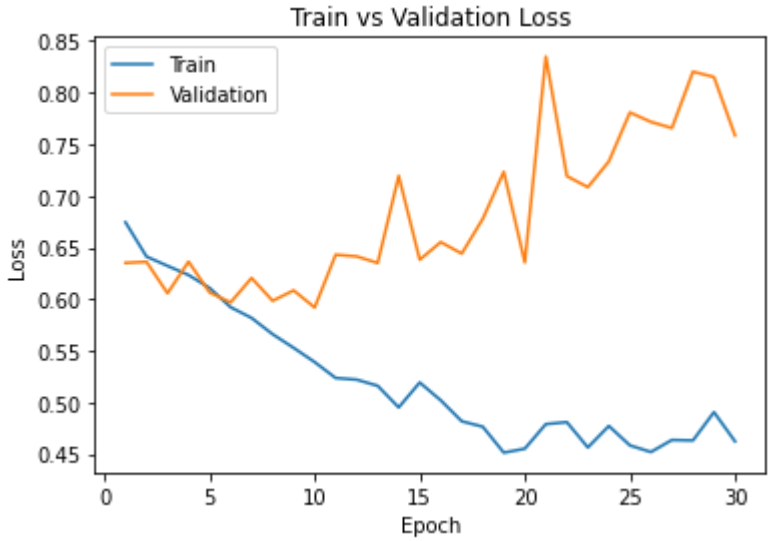
```
large_net = LargeNet()
train_net(large_net, "/content/gdrive/My Drive/Colab Notebooks/APS360_labs/Lab2_Part3b"
, learning_rate=0.1)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.4295, Train loss: 0.67437779712677 |Validation err:
0.3595, Validation loss: 0.6350857093930244
Epoch 2: Train err: 0.36075, Train loss: 0.6411805458068848 |Validation er
r: 0.3535, Validation loss: 0.6361209936439991
Epoch 3: Train err: 0.365125, Train loss: 0.6321813461780548 |Validation e
rr: 0.3385, Validation loss: 0.6056603882461786
Epoch 4: Train err: 0.352625, Train loss: 0.6233456182479858 |Validation e
rr: 0.3575, Validation loss: 0.6362800188362598
Epoch 5: Train err: 0.34075, Train loss: 0.6108013873100281 |Validation er
r: 0.3305, Validation loss: 0.6064918786287308
Epoch 6: Train err: 0.323375, Train loss: 0.5921835997104645 |Validation e
rr: 0.317, Validation loss: 0.5967769594863057
Epoch 7: Train err: 0.3145, Train loss: 0.5817317583560944 |Validation er
r: 0.3365, Validation loss: 0.6204487886279821
Epoch 8: Train err: 0.29825, Train loss: 0.5660300073623658 |Validation er
r: 0.3285, Validation loss: 0.5983372200280428
Epoch 9: Train err: 0.290875, Train loss: 0.552809501171112 |Validation er
r: 0.3315, Validation loss: 0.6084455158561468
Epoch 10: Train err: 0.278625, Train loss: 0.539032607793808 |Validation e
rr: 0.306, Validation loss: 0.5918631898239255
Epoch 11: Train err: 0.272375, Train loss: 0.5236025826931 |Validation er
r: 0.33, Validation loss: 0.6430060230195522
Epoch 12: Train err: 0.267375, Train loss: 0.5220149435997009 |Validation
err: 0.2925, Validation loss: 0.6413561534136534
Epoch 13: Train err: 0.266, Train loss: 0.5160510110855102 |Validation er
r: 0.3125, Validation loss: 0.6349832843989134
Epoch 14: Train err: 0.24875, Train loss: 0.4951590054035187 |Validation e
rr: 0.3145, Validation loss: 0.7193072671070695
Epoch 15: Train err: 0.264625, Train loss: 0.519231944322586 |Validation e
rr: 0.314, Validation loss: 0.6381420725956559
Epoch 16: Train err: 0.252625, Train loss: 0.5020012385845184 |Validation
err: 0.3225, Validation loss: 0.6551959458738565
Epoch 17: Train err: 0.23875, Train loss: 0.481714787364006 |Validation er
r: 0.357, Validation loss: 0.6440742611885071
Epoch 18: Train err: 0.23375, Train loss: 0.47645506453514097 |Validation
err: 0.3375, Validation loss: 0.6777342790737748
Epoch 19: Train err: 0.218125, Train loss: 0.45134368968009947 |Validation
err: 0.3445, Validation loss: 0.7232250478118658
Epoch 20: Train err: 0.217875, Train loss: 0.45516350817680357 |Validation
err: 0.3245, Validation loss: 0.6354950983077288
Epoch 21: Train err: 0.23275, Train loss: 0.47897080445289614 |Validation
err: 0.3255, Validation loss: 0.8348110988736153
Epoch 22: Train err: 0.234875, Train loss: 0.4808810565471649 |Validation
err: 0.334, Validation loss: 0.7191346418112516
Epoch 23: Train err: 0.21575, Train loss: 0.4563647754192352 |Validation e
rr: 0.316, Validation loss: 0.7083508176729083
Epoch 24: Train err: 0.2355, Train loss: 0.47718250966072084 |Validation e
rr: 0.327, Validation loss: 0.7333047650754452
Epoch 25: Train err: 0.22025, Train loss: 0.4583414270877838 |Validation e
rr: 0.3315, Validation loss: 0.7806987538933754
Epoch 26: Train err: 0.209625, Train loss: 0.4519626965522766 |Validation
err: 0.3435, Validation loss: 0.7715998776257038
Epoch 27: Train err: 0.22175, Train loss: 0.4636160457134247 |Validation e
rr: 0.3215, Validation loss: 0.7656293725594878
Epoch 28: Train err: 0.219375, Train loss: 0.46314777398109436 |Validation
err: 0.348, Validation loss: 0.8202023077756166
Epoch 29: Train err: 0.235875, Train loss: 0.49053542733192446 |Validation
err: 0.326, Validation loss: 0.8150460105389357
Epoch 30: Train err: 0.22, Train loss: 0.4623157248497009 |Validation err:
```

```
0.3165, Validation loss: 0.7585078496485949
Finished Training
Total time elapsed: 131.20 seconds
```

In [44]:

```
model_path = get_model_name("large", batch_size=64, learning_rate=0.1, epoch=29)
path = "/content/gdrive/My Drive/Colab Notebooks/APS360_labs/Lab2_Part3b"
path_large = "/".join([path, model_path])
plot_training_curve(path_large)
```



As the learning rate increased from 0.01 to 0.1, the trainning time increases from 126.17s to 131.2s. As we increased the learning rate, the size of gradient descent step become larger, the validation and trainning error/loss decreases faster than default. The overfitting is also happened earlier at around epoch 5-10 compared to the epoch 10-15 in the default result.

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 [50]:

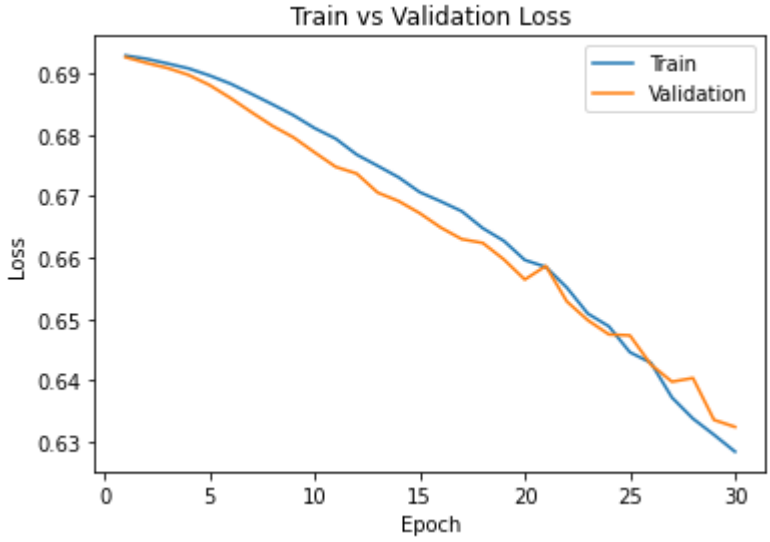
```
large_net = LargeNet()
train_net(large_net, "/content/gdrive/My Drive/Colab Notebooks/APS360_labs/Lab2_Part3c"
, batch_size=512, learning_rate=0.01, num_epochs=30)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.48175, Train loss: 0.6929379552602768 |Validation er
r: 0.478, Validation loss: 0.6926824003458023
Epoch 2: Train err: 0.457625, Train loss: 0.6924104019999504 |Validation e
rr: 0.434, Validation loss: 0.6917425245046616
Epoch 3: Train err: 0.437, Train loss: 0.6916500590741634 |Validation err:
0.4265, Validation loss: 0.6909129917621613
Epoch 4: Train err: 0.433625, Train loss: 0.6908449940383434 |Validation e
rr: 0.424, Validation loss: 0.6897870451211929
Epoch 5: Train err: 0.434, Train loss: 0.6896935552358627 |Validation err:
0.424, Validation loss: 0.6881355047225952
Epoch 6: Train err: 0.438, Train loss: 0.688353206962347 |Validation err:
0.4285, Validation loss: 0.686011865735054
Epoch 7: Train err: 0.439375, Train loss: 0.6866871677339077 |Validation e
rr: 0.426, Validation loss: 0.6836968809366226
Epoch 8: Train err: 0.43525, Train loss: 0.6849770769476891 |Validation er
r: 0.4115, Validation loss: 0.6814671903848648
Epoch 9: Train err: 0.42375, Train loss: 0.6832009293138981 |Validation er
r: 0.414, Validation loss: 0.679591491818428
Epoch 10: Train err: 0.421, Train loss: 0.6811089366674423 |Validation er
r: 0.416, Validation loss: 0.6771548539400101
Epoch 11: Train err: 0.420875, Train loss: 0.6794026419520378 |Validation
err: 0.4095, Validation loss: 0.6748111099004745
Epoch 12: Train err: 0.41475, Train loss: 0.6768048219382763 |Validation e
rr: 0.412, Validation loss: 0.6737060546875
Epoch 13: Train err: 0.4105, Train loss: 0.6749702803790569 |Validation er
r: 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: 0.6706806942820549 |Validation er
r: 0.4105, Validation loss: 0.667252704501152
Epoch 16: Train err: 0.397625, Train loss: 0.6691771410405636 |Validation
err: 0.405, Validation loss: 0.6649097055196762
Epoch 17: Train err: 0.393875, Train loss: 0.6675694733858109 |Validation
err: 0.401, Validation loss: 0.6630224883556366
Epoch 18: Train err: 0.393, Train loss: 0.6648042872548103 |Validation er
r: 0.3945, Validation loss: 0.6624014377593994
Epoch 19: Train err: 0.38625, Train loss: 0.662746611982584 |Validation er
r: 0.388, Validation loss: 0.6597220152616501
Epoch 20: Train err: 0.38175, Train loss: 0.6596181839704514 |Validation e
rr: 0.4005, Validation loss: 0.6564337313175201
Epoch 21: Train err: 0.38575, Train loss: 0.6584899798035622 |Validation e
rr: 0.3885, Validation loss: 0.6586423963308334
Epoch 22: Train err: 0.378125, Train loss: 0.655123382806778 |Validation e
rr: 0.3855, Validation loss: 0.6528600305318832
Epoch 23: Train err: 0.372125, Train loss: 0.6508794128894806 |Validation
err: 0.3835, Validation loss: 0.6497963815927505
Epoch 24: Train err: 0.37675, Train loss: 0.6488028429448605 |Validation e
rr: 0.385, Validation loss: 0.6474899500608444
Epoch 25: Train err: 0.368625, Train loss: 0.6445869170129299 |Validation
err: 0.382, Validation loss: 0.6473268568515778
Epoch 26: Train err: 0.372625, Train loss: 0.6428566053509712 |Validation
err: 0.3745, Validation loss: 0.6425703465938568
Epoch 27: Train err: 0.359375, Train loss: 0.6372117549180984 |Validation
err: 0.379, Validation loss: 0.6397799849510193
Epoch 28: Train err: 0.35425, Train loss: 0.6337667480111122 |Validation e
rr: 0.3695, Validation loss: 0.6403783112764359
Epoch 29: Train err: 0.3535, Train loss: 0.6311353109776974 |Validation er
r: 0.366, Validation loss: 0.6335585117340088
Epoch 30: Train err: 0.353, Train loss: 0.6283832415938377 |Validation er
```

```
r: 0.3675, Validation loss: 0.6324127316474915
Finished Training
Total time elapsed: 118.72 seconds
```

In [51]:

```
model_path = get_model_name("large", batch_size=512, learning_rate=0.01, epoch=29)
path = "/content/gdrive/My Drive/Colab Notebooks/APS360_labs/Lab2_Part3c"
path_large = "/".join([path, model_path])
plot_training_curve(path_large)
```



As the batch_size increased from 64 to 512, the trainning time decreases from 126.17s to 118.72s. As we increased the batch_size, the trainning error/loss graph is similar with the validation error/loss graph.

It has a similar result as learning_rate = 0.01 and there is no significant overfitting happened.

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 [52]:

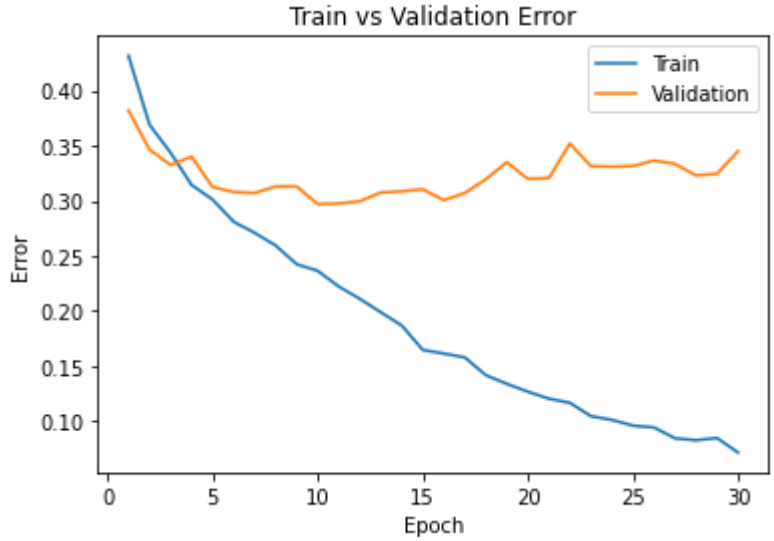
```
large_net = LargeNet()
train_net(large_net, "/content/gdrive/My Drive/Colab Notebooks/APS360_labs/Lab2_Part3d"
, batch_size=16, learning_rate=0.01, num_epochs=30)
```

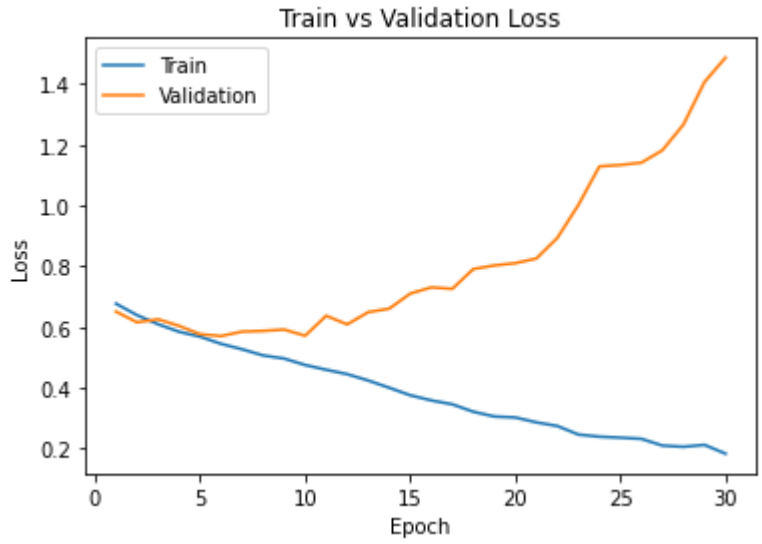
```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.43175, Train loss: 0.6774994022846222 |Validation er
r: 0.382, Validation loss: 0.6513170118331909
Epoch 2: Train err: 0.369, Train loss: 0.639639899969101 |Validation err:
0.3465, Validation loss: 0.6161113576889038
Epoch 3: Train err: 0.34375, Train loss: 0.6098222947120666 |Validation er
r: 0.3325, Validation loss: 0.6260210764408112
Epoch 4: Train err: 0.314375, Train loss: 0.5849691489338875 |Validation e
rr: 0.34, Validation loss: 0.6044013917446136
Epoch 5: Train err: 0.301125, Train loss: 0.5689119303822517 |Validation e
rr: 0.3125, Validation loss: 0.576918310880661
Epoch 6: Train err: 0.281, Train loss: 0.5452213581204415 |Validation err:
0.308, Validation loss: 0.5708447456359863
Epoch 7: Train err: 0.270875, Train loss: 0.5272981298565864 |Validation e
rr: 0.307, Validation loss: 0.5854293291568756
Epoch 8: Train err: 0.259375, Train loss: 0.5070905526578426 |Validation e
rr: 0.313, Validation loss: 0.5877130818367005
Epoch 9: Train err: 0.242375, Train loss: 0.4968344421982765 |Validation e
rr: 0.313, Validation loss: 0.5922425072193146
Epoch 10: Train err: 0.236375, Train loss: 0.4756101597249508 |Validation
err: 0.297, Validation loss: 0.5718690166473389
Epoch 11: Train err: 0.222125, Train loss: 0.4599769461452961 |Validation
err: 0.2975, Validation loss: 0.6376970833539963
Epoch 12: Train err: 0.211, Train loss: 0.4454492371380329 |Validation er
r: 0.2995, Validation loss: 0.609202565908432
Epoch 13: Train err: 0.19875, Train loss: 0.4245421719551086 |Validation e
rr: 0.3075, Validation loss: 0.6494987765550614
Epoch 14: Train err: 0.18675, Train loss: 0.4007472907453775 |Validation e
rr: 0.3085, Validation loss: 0.6610016552209854
Epoch 15: Train err: 0.1645, Train loss: 0.3759974058121443 |Validation er
r: 0.3105, Validation loss: 0.7106090537309646
Epoch 16: Train err: 0.16125, Train loss: 0.3591455406397581 |Validation e
rr: 0.3005, Validation loss: 0.7310364942550659
Epoch 17: Train err: 0.15775, Train loss: 0.3463234790861607 |Validation e
rr: 0.307, Validation loss: 0.7263009325265884
Epoch 18: Train err: 0.141625, Train loss: 0.32175366275012496 |Validation
err: 0.3195, Validation loss: 0.7913952842950821
Epoch 19: Train err: 0.13375, Train loss: 0.30618105667084455 |Validation
err: 0.335, Validation loss: 0.8032052783966065
Epoch 20: Train err: 0.126625, Train loss: 0.3029071792438626 |Validation
err: 0.32, Validation loss: 0.8106685240268707
Epoch 21: Train err: 0.12025, Train loss: 0.28682796490937473 |Validation
err: 0.3205, Validation loss: 0.8259474284648896
Epoch 22: Train err: 0.1165, Train loss: 0.27489088076353074 |Validation e
rr: 0.352, Validation loss: 0.8937610774040222
Epoch 23: Train err: 0.104375, Train loss: 0.2467898527495563 |Validation
err: 0.3315, Validation loss: 1.0021928198337555
Epoch 24: Train err: 0.101, Train loss: 0.23970085787773132 |Validation er
r: 0.331, Validation loss: 1.1290796399116516
Epoch 25: Train err: 0.09575, Train loss: 0.23643119425699116 |Validation
err: 0.3315, Validation loss: 1.1338514368534087
Epoch 26: Train err: 0.094125, Train loss: 0.2325953512713313 |Validation
err: 0.3365, Validation loss: 1.1414263204336166
Epoch 27: Train err: 0.08425, Train loss: 0.21040759468451142 |Validation
err: 0.3335, Validation loss: 1.1823678107261657
Epoch 28: Train err: 0.0825, Train loss: 0.20643112615589052 |Validation e
rr: 0.323, Validation loss: 1.266836181640625
Epoch 29: Train err: 0.0845, Train loss: 0.21273409337876364 |Validation e
rr: 0.3245, Validation loss: 1.406717705130577
Epoch 30: Train err: 0.071375, Train loss: 0.18387044295761734 |Validation
```

```
err: 0.345, Validation loss: 1.4871552000045776
Finished Training
Total time elapsed: 179.30 seconds
```

In [53]:

```
model_path = get_model_name("large", batch_size=16, learning_rate=0.01, epoch=29)
path = "/content/gdrive/My Drive/Colab Notebooks/APS360_labs/Lab2_Part3d"
path_large = "/".join([path, model_path])
plot_training_curve(path_large)
```





As the batch_size decreased from 64 to 16, the trainning time increases from 126.17s to 179.3s.

The overfitting is happened earlier at around epoch 5 compared to the epoch 10-15 in the default result.

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.

The hyperparameters I chose are large_net, batch_size = 512, learning_rate = 0.001. The large_net will get a less noisy result than small_net and by observing the result above, larger batch_size and lower learning rate can best avoid the occurance of overfitting.

Part (b) - 1pt

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

In [54]:

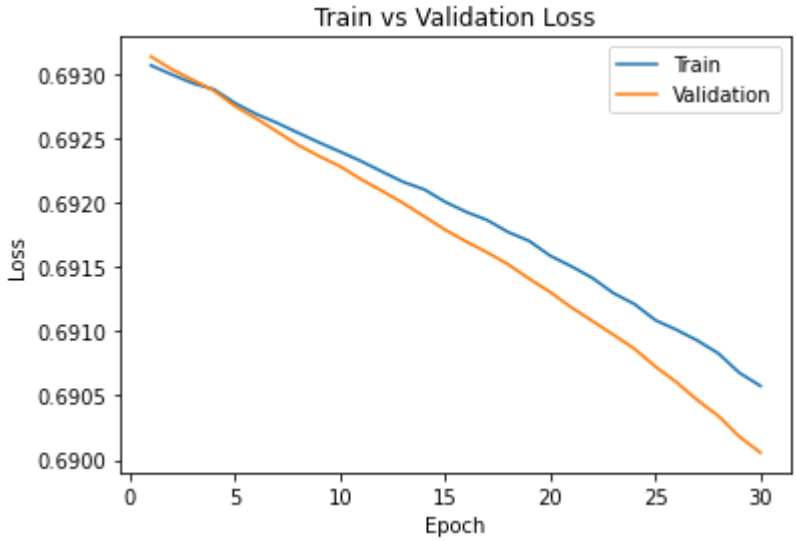
```
large_net = LargeNet()
train_net(large_net, "/content/gdrive/My Drive/Colab Notebooks/APS360_labs/Lab2_Part4b"
, batch_size=512, learning_rate=0.001, num_epochs=30)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.48825, Train loss: 0.6930677480995655 |Validation er
r: 0.4955, Validation loss: 0.6931362152099609
Epoch 2: Train err: 0.483125, Train loss: 0.692995510995388 |Validation er
r: 0.4945, Validation loss: 0.6930360496044159
Epoch 3: Train err: 0.480375, Train loss: 0.6929280497133732 |Validation e
rr: 0.493, Validation loss: 0.6929539889097214
Epoch 4: Train err: 0.477, Train loss: 0.6928808391094208 |Validation err:
0.4885, Validation loss: 0.692870706319809
Epoch 5: Train err: 0.473375, Train loss: 0.692774411290884 |Validation er
r: 0.4835, Validation loss: 0.6927504986524582
Epoch 6: Train err: 0.469, Train loss: 0.6926896274089813 |Validation err:
0.472, Validation loss: 0.6926551759243011
Epoch 7: Train err: 0.46325, Train loss: 0.692620363086462 |Validation er
r: 0.47, Validation loss: 0.6925524920225143
Epoch 8: Train err: 0.46225, Train loss: 0.6925435550510883 |Validation er
r: 0.463, Validation loss: 0.6924485266208649
Epoch 9: Train err: 0.459625, Train loss: 0.6924680322408676 |Validation e
rr: 0.457, Validation loss: 0.6923621594905853
Epoch 10: Train err: 0.458, Train loss: 0.6923965662717819 |Validation er
r: 0.4555, Validation loss: 0.6922826021909714
Epoch 11: Train err: 0.454875, Train loss: 0.6923230737447739 |Validation
err: 0.4505, Validation loss: 0.6921818852424622
Epoch 12: Train err: 0.4535, Train loss: 0.6922412514686584 |Validation er
r: 0.441, Validation loss: 0.6920914500951767
Epoch 13: Train err: 0.450375, Train loss: 0.6921614557504654 |Validation
err: 0.437, Validation loss: 0.691996842622757
Epoch 14: Train err: 0.44725, Train loss: 0.6921032443642616 |Validation e
rr: 0.433, Validation loss: 0.6918932348489761
Epoch 15: Train err: 0.449375, Train loss: 0.6920064650475979 |Validation
err: 0.432, Validation loss: 0.6917892098426819
Epoch 16: Train err: 0.44425, Train loss: 0.6919283680617809 |Validation e
rr: 0.432, Validation loss: 0.6916972398757935
Epoch 17: Train err: 0.441375, Train loss: 0.6918644718825817 |Validation
err: 0.431, Validation loss: 0.6916135102510452
Epoch 18: Train err: 0.438125, Train loss: 0.6917712315917015 |Validation
err: 0.4295, Validation loss: 0.6915201395750046
Epoch 19: Train err: 0.436375, Train loss: 0.6917018257081509 |Validation
err: 0.428, Validation loss: 0.6914086490869522
Epoch 20: Train err: 0.436375, Train loss: 0.6915871091187 |Validation er
r: 0.4275, Validation loss: 0.6913044154644012
Epoch 21: Train err: 0.437, Train loss: 0.6915052235126495 |Validation er
r: 0.4285, Validation loss: 0.6911860555410385
Epoch 22: Train err: 0.438625, Train loss: 0.6914149634540081 |Validation
err: 0.428, Validation loss: 0.6910803616046906
Epoch 23: Train err: 0.436875, Train loss: 0.6912974379956722 |Validation
err: 0.428, Validation loss: 0.6909734457731247
Epoch 24: Train err: 0.436875, Train loss: 0.6912120543420315 |Validation
err: 0.425, Validation loss: 0.6908644735813141
Epoch 25: Train err: 0.435125, Train loss: 0.6910865269601345 |Validation
err: 0.4255, Validation loss: 0.6907256692647934
Epoch 26: Train err: 0.434625, Train loss: 0.6910119205713272 |Validation
err: 0.4245, Validation loss: 0.6906051337718964
Epoch 27: Train err: 0.43675, Train loss: 0.6909283325076103 |Validation e
rr: 0.4265, Validation loss: 0.6904648989439011
Epoch 28: Train err: 0.43575, Train loss: 0.6908275187015533 |Validation e
rr: 0.4265, Validation loss: 0.6903413087129593
Epoch 29: Train err: 0.436375, Train loss: 0.6906765103340149 |Validation
err: 0.423, Validation loss: 0.6901802867650986
Epoch 30: Train err: 0.435625, Train loss: 0.6905755028128624 |Validation
```

```
err: 0.4235, Validation loss: 0.6900565475225449
Finished Training
Total time elapsed: 113.93 seconds
```

In [55]:

```
model_path = get_model_name("large", batch_size=512, learning_rate=0.001, epoch=29)
path = "/content/gdrive/My Drive/Colab Notebooks/APS360_labs/Lab2_Part4b"
path_large = "/".join([path, model_path])
plot_training_curve(path_large)
```



Part (c) - 2pt

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

The new set of hyperparameter values I chose are large_net, batch_size = 512, learning_rate = 0.05.

The large_net will get a less noisy result than small_net and by observing the result above, larger batch_size and lower learning rate can best avoid the occurance of overfitting.

I want to keep these characteristic and lowering the trainning time of the model, so I decide to keep a low learning rate but increases a bit to make the error/loss decreases faster.

Part (d) - 1pt

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

In [66]:

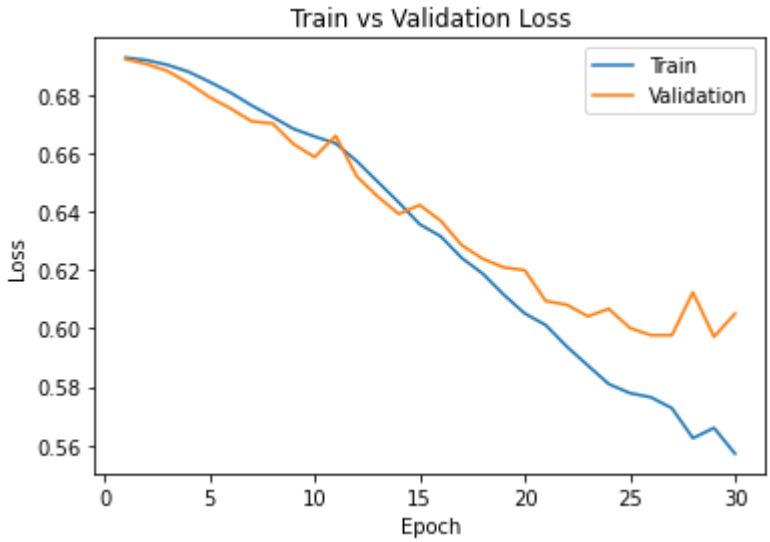
```
large_net = LargeNet()
train_net(large_net, "/content/gdrive/My Drive/Colab Notebooks/APS360_labs/Lab2_Part4d"
, batch_size=1024, learning_rate=0.05, num_epochs=30)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.48025, Train loss: 0.692861519753933 |Validation er
r: 0.4575, Validation loss: 0.6923681497573853
Epoch 2: Train err: 0.4515, Train loss: 0.6919781863689423 |Validation er
r: 0.4475, Validation loss: 0.690619170665741
Epoch 3: Train err: 0.448625, Train loss: 0.6903635263442993 |Validation e
rr: 0.4305, Validation loss: 0.6882481575012207
Epoch 4: Train err: 0.433375, Train loss: 0.6879977062344551 |Validation e
rr: 0.421, Validation loss: 0.6841238141059875
Epoch 5: Train err: 0.43375, Train loss: 0.6845527961850166 |Validation er
r: 0.414, Validation loss: 0.6792883276939392
Epoch 6: Train err: 0.425125, Train loss: 0.6808486878871918 |Validation e
rr: 0.4185, Validation loss: 0.6754190325737
Epoch 7: Train err: 0.41725, Train loss: 0.6764698624610901 |Validation er
r: 0.4105, Validation loss: 0.6710297763347626
Epoch 8: Train err: 0.408875, Train loss: 0.6725167036056519 |Validation e
rr: 0.4065, Validation loss: 0.6702995002269745
Epoch 9: Train err: 0.400125, Train loss: 0.6684329062700272 |Validation e
rr: 0.402, Validation loss: 0.6631980240345001
Epoch 10: Train err: 0.393125, Train loss: 0.6658134162425995 |Validation
err: 0.397, Validation loss: 0.658738762140274
Epoch 11: Train err: 0.39625, Train loss: 0.6634786501526833 |Validation e
rr: 0.397, Validation loss: 0.6660594642162323
Epoch 12: Train err: 0.38725, Train loss: 0.6574025899171829 |Validation e
rr: 0.3755, Validation loss: 0.6521849036216736
Epoch 13: Train err: 0.37575, Train loss: 0.6503145918250084 |Validation e
rr: 0.3825, Validation loss: 0.6452338695526123
Epoch 14: Train err: 0.36675, Train loss: 0.643231600522995 |Validation er
r: 0.3735, Validation loss: 0.6392670273780823
Epoch 15: Train err: 0.36025, Train loss: 0.6357321217656136 |Validation e
rr: 0.3745, Validation loss: 0.642320454120636
Epoch 16: Train err: 0.3535, Train loss: 0.6315861791372299 |Validation er
r: 0.366, Validation loss: 0.63691446185112
Epoch 17: Train err: 0.349875, Train loss: 0.6241620779037476 |Validation
err: 0.3515, Validation loss: 0.6285594999790192
Epoch 18: Train err: 0.3425, Train loss: 0.618795245885849 |Validation er
r: 0.342, Validation loss: 0.6238750517368317
Epoch 19: Train err: 0.336, Train loss: 0.611568033695221 |Validation err:
0.3415, Validation loss: 0.620937317609787
Epoch 20: Train err: 0.33525, Train loss: 0.605135902762413 |Validation er
r: 0.345, Validation loss: 0.6199671626091003
Epoch 21: Train err: 0.327625, Train loss: 0.6011437997221947 |Validation
err: 0.3385, Validation loss: 0.6093878448009491
Epoch 22: Train err: 0.322375, Train loss: 0.5937773063778877 |Validation
err: 0.3315, Validation loss: 0.6081297397613525
Epoch 23: Train err: 0.316625, Train loss: 0.5873182564973831 |Validation
err: 0.3295, Validation loss: 0.6041537225246429
Epoch 24: Train err: 0.3115, Train loss: 0.5809303000569344 |Validation er
r: 0.33, Validation loss: 0.606731116771698
Epoch 25: Train err: 0.309, Train loss: 0.5778397247195244 |Validation er
r: 0.3185, Validation loss: 0.6001583635807037
Epoch 26: Train err: 0.30825, Train loss: 0.5764101147651672 |Validation e
rr: 0.307, Validation loss: 0.5977014303207397
Epoch 27: Train err: 0.30375, Train loss: 0.5726678594946861 |Validation e
rr: 0.314, Validation loss: 0.5976929664611816
Epoch 28: Train err: 0.299625, Train loss: 0.5623890832066536 |Validation
err: 0.34, Validation loss: 0.6123574674129486
Epoch 29: Train err: 0.297625, Train loss: 0.5659100711345673 |Validation
err: 0.3255, Validation loss: 0.5971436202526093
Epoch 30: Train err: 0.29875, Train loss: 0.5570858344435692 |Validation e
```

```
rr: 0.306, Validation loss: 0.6050869226455688
Finished Training
Total time elapsed: 121.19 seconds
```

In [67]:

```
model_path = get_model_name("large", batch_size=1024, learning_rate=0.05, epoch=29)
path = "/content/gdrive/My Drive/Colab Notebooks/APS360_labs/Lab2_Part4d"
path_large = "/".join([path, model_path])
plot_training_curve(path_large)
```



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 [88]:

```
net = LargeNet()
model_path = get_model_name("large", batch_size=1024, learning_rate=0.05, epoch=19)
model_path = "/".join(["/content/gdrive/My Drive/Colab Notebooks/APS360_labs/Lab2_Part5
a", model_path])
state = torch.load(model_path)
net.load_state_dict(state)
```

Out[88]:

```
<All keys matched successfully>
```

Part (b) - 2pt

Justify your choice of model from part (a).

I chose large_net because it is observed to be less noisy than small_net, but it will cause overfitting so the other parameter should be adjusted to solve this problem.

I chose batch_size=1024 since increases in batch_size can effectively avoid the occurrence of overfitting when epoch become larger.

I chose learning_rate=0.05 so it can make the err/lose decreases faster

I chose num_epochs=19 because the model tends to be overfitted when the number of epoch become larger.

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 [90]:

```
# 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=1024)
criterion = nn.BCEWithLogitsLoss()
err, loss = evaluate(net, test_loader, criterion)
print("Test classification error: ", err, "Test loss: ", loss)
```

```
Files already downloaded and verified
Files already downloaded and verified
Test classification error:  0.341 Test loss:  0.6156387627124786
```

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.

The test classification error(0.341) appears to be slightly lower than the validation error(0.345), but they are very similar.

It is expected that the test error will be higher than the validation error because we tuned the hyparameters of the model based on the validation data set, it is normal to think that the model will have a better peroformance in validation.

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?

The test data used only at the very end because we want the model not be biased from the test data, otherwise the model will adjust its performance and make it more fit to the test result and turned to be perform bad on the new data, which is we do not desire to be.

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 [94]:

```
class Pigeon(nn.Module):
    def __init__(self):
        super(Pigeon, self).__init__()
        self.name = "pigeon"
        #change the # of hidden units from 30 to 50/100/200/500/1000 respectively
        self.layer1 = nn.Linear(3 * 32 * 32, 30)
        self.layer2 = nn.Linear(30, 1)
    def forward(self, img):
        flattened = img.view(-1, 3 * 32 * 32)
        activation1 = self.layer1(flattened)
        activation1 = F.relu(activation1)
        activation2 = self.layer2(activation1)
        activation2 = activation2.squeeze(1)
        return activation2
```

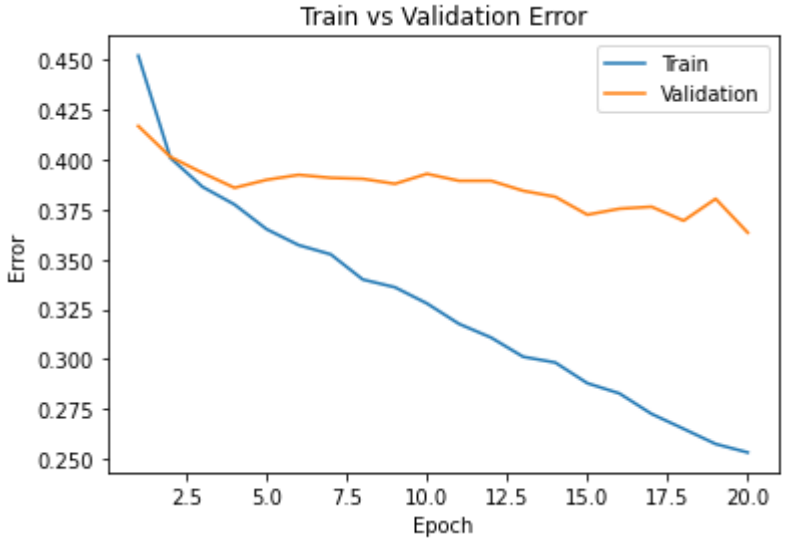
In [98]:

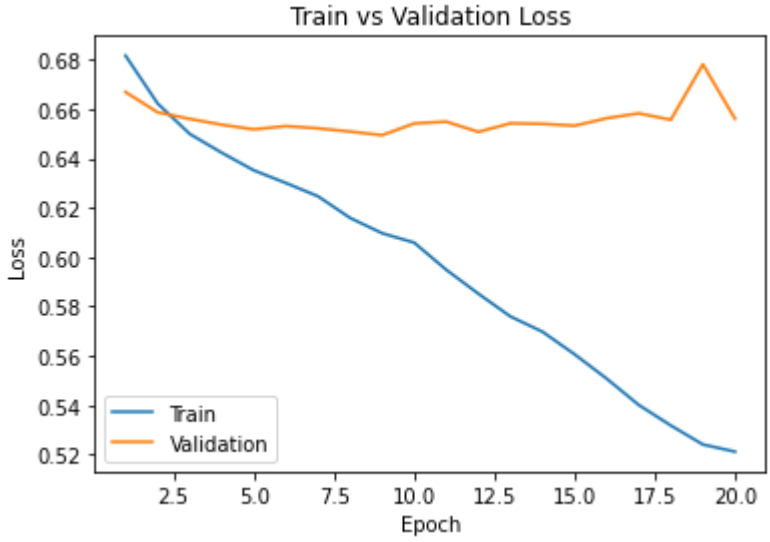
```
pigeon = Pigeon()
train_net(pigeon, "/content/gdrive/My Drive/Colab Notebooks/APS360_labs/Lab2_Part5f", b
atch_size=1024, learning_rate=0.05, num_epochs=20)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.45225, Train loss: 0.6817076131701469 |Validation er
r: 0.417, Validation loss: 0.6669414043426514
Epoch 2: Train err: 0.401, Train loss: 0.6623173803091049 |Validation err:
0.4015, Validation loss: 0.658717691898346
Epoch 3: Train err: 0.3865, Train loss: 0.6499697342514992 |Validation er
r: 0.3935, Validation loss: 0.6560535430908203
Epoch 4: Train err: 0.377625, Train loss: 0.6423265263438225 |Validation e
rr: 0.386, Validation loss: 0.6536880433559418
Epoch 5: Train err: 0.36525, Train loss: 0.6351850032806396 |Validation er
r: 0.39, Validation loss: 0.6518155634403229
Epoch 6: Train err: 0.35725, Train loss: 0.6300496086478233 |Validation er
r: 0.3925, Validation loss: 0.6531242430210114
Epoch 7: Train err: 0.352625, Train loss: 0.6246740072965622 |Validation e
rr: 0.391, Validation loss: 0.6522330045700073
Epoch 8: Train err: 0.34, Train loss: 0.615900307893753 |Validation err:
0.3905, Validation loss: 0.6509090065956116
Epoch 9: Train err: 0.336125, Train loss: 0.6096850708127022 |Validation e
rr: 0.388, Validation loss: 0.6494366228580475
Epoch 10: Train err: 0.328, Train loss: 0.6058643683791161 |Validation er
r: 0.393, Validation loss: 0.6541977524757385
Epoch 11: Train err: 0.31775, Train loss: 0.5948565155267715 |Validation e
rr: 0.3895, Validation loss: 0.6549278497695923
Epoch 12: Train err: 0.310875, Train loss: 0.585158184170723 |Validation e
rr: 0.3895, Validation loss: 0.6507782340049744
Epoch 13: Train err: 0.30125, Train loss: 0.5759217962622643 |Validation e
rr: 0.3845, Validation loss: 0.6542501747608185
Epoch 14: Train err: 0.298375, Train loss: 0.5697237998247147 |Validation
err: 0.3815, Validation loss: 0.6540387868881226
Epoch 15: Train err: 0.288, Train loss: 0.5605904906988144 |Validation er
r: 0.3725, Validation loss: 0.6532819867134094
Epoch 16: Train err: 0.282875, Train loss: 0.5507166758179665 |Validation
err: 0.3755, Validation loss: 0.6563006341457367
Epoch 17: Train err: 0.272625, Train loss: 0.5400434955954552 |Validation
err: 0.3765, Validation loss: 0.658290833234787
Epoch 18: Train err: 0.265125, Train loss: 0.5317176207900047 |Validation
err: 0.3695, Validation loss: 0.6556809544563293
Epoch 19: Train err: 0.2575, Train loss: 0.5239980965852737 |Validation er
r: 0.3805, Validation loss: 0.6781530678272247
Epoch 20: Train err: 0.25325, Train loss: 0.5211120545864105 |Validation e
rr: 0.3635, Validation loss: 0.6561074256896973
Finished Training
Total time elapsed: 64.00 seconds
```

In [99]:

```
model_path = get_model_name("pigeon", batch_size=1024, learning_rate=0.05, epoch=19)
path = "/content/gdrive/My Drive/Colab Notebooks/APS360_labs/Lab2_Part5f"
path_large = "/".join([path, model_path])
plot_training_curve(path_large)
```





In [100]:

```
train_loader, val_loader, test_loader, classes = get_data_loader(
    target_classes=["cat", "dog"],
    batch_size=1024)
criterion = nn.BCEWithLogitsLoss()
err, loss = evaluate(pigeon, test_loader, criterion)
print("Test classification error: ", err, "Test loss: ", loss)
```

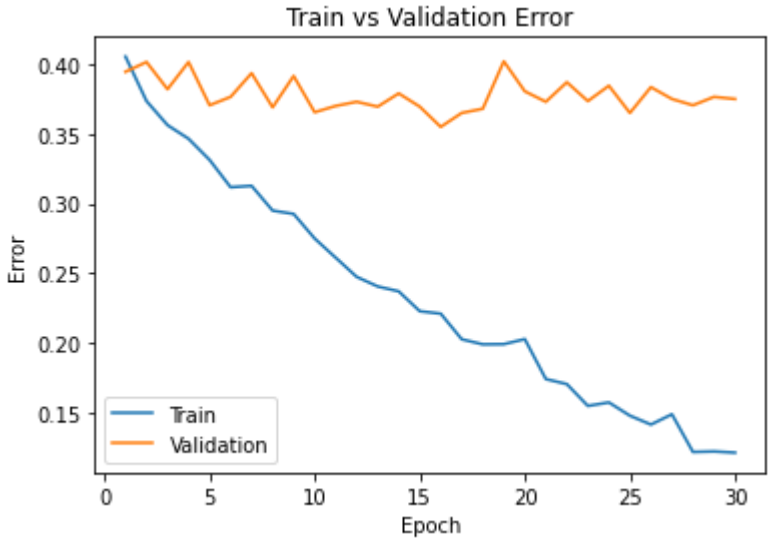
```
Files already downloaded and verified
Files already downloaded and verified
Test classification error:  0.359 Test loss:  0.6557274460792542
```

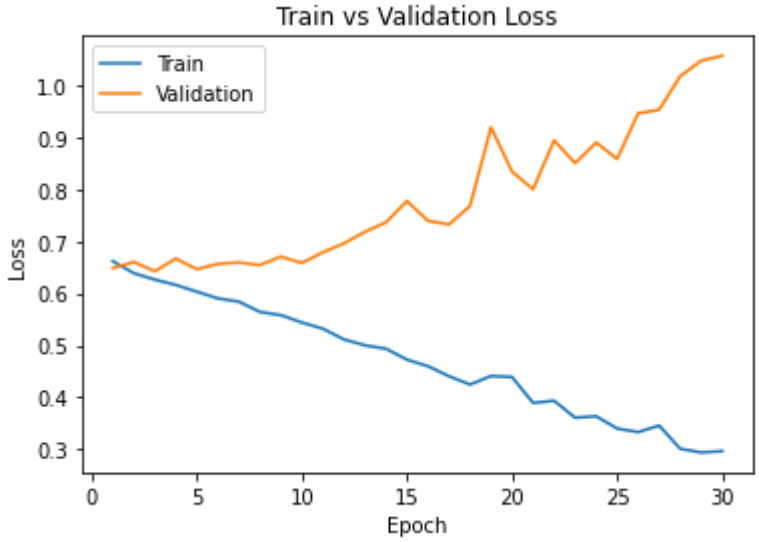
In [102]:

```
pigeon = Pigeon()
train_net(pigeon, "/content/gdrive/My Drive/Colab Notebooks/APS360_labs/Lab2_Part5f1",
batch_size=64, learning_rate=0.01, num_epochs=30)

model_path = get_model_name("pigeon", batch_size=64, learning_rate=0.01, epoch=29)
path = "/content/gdrive/My Drive/Colab Notebooks/APS360_labs/Lab2_Part5f1"
path_large = "/".join([path, model_path])
plot_training_curve(path_large)

train_loader, val_loader, test_loader, classes = get_data_loader(
    target_classes=["cat", "dog"],
    batch_size=64)
criterion = nn.BCEWithLogitsLoss()
err, loss = evaluate(pigeon, test_loader, criterion)
print("Test classification error: ", err, "Test loss: ", loss)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.405375, Train loss: 0.6620588073730469 |Validation e
rr: 0.3945, Validation loss: 0.6485726945102215
Epoch 2: Train err: 0.3735, Train loss: 0.6388178849220276 |Validation er
r: 0.4015, Validation loss: 0.6603791806846857
Epoch 3: Train err: 0.35625, Train loss: 0.6265799717903138 |Validation er
r: 0.382, Validation loss: 0.6425359733402729
Epoch 4: Train err: 0.3465, Train loss: 0.6162636213302612 |Validation er
r: 0.4015, Validation loss: 0.6670328862965107
Epoch 5: Train err: 0.331375, Train loss: 0.6033147349357605 |Validation e
rr: 0.3705, Validation loss: 0.646591255441308
Epoch 6: Train err: 0.31175, Train loss: 0.5903615536689758 |Validation er
r: 0.3765, Validation loss: 0.656750975176692
Epoch 7: Train err: 0.31275, Train loss: 0.5841613943576813 |Validation er
r: 0.3935, Validation loss: 0.659564808011055
Epoch 8: Train err: 0.295, Train loss: 0.5643694820404053 |Validation err:
0.369, Validation loss: 0.6544384583830833
Epoch 9: Train err: 0.292625, Train loss: 0.5581223332881927 |Validation e
rr: 0.3915, Validation loss: 0.670752914622426
Epoch 10: Train err: 0.275, Train loss: 0.5440180177688598 |Validation er
r: 0.3655, Validation loss: 0.6585566084831953
Epoch 11: Train err: 0.26125, Train loss: 0.5320472874641419 |Validation e
rr: 0.37, Validation loss: 0.6790587101131678
Epoch 12: Train err: 0.2475, Train loss: 0.5114462027549743 |Validation er
r: 0.373, Validation loss: 0.6962906178086996
Epoch 13: Train err: 0.240625, Train loss: 0.5001414263248444 |Validation
err: 0.3695, Validation loss: 0.7184182889759541
Epoch 14: Train err: 0.237125, Train loss: 0.4934847273826599 |Validation
err: 0.379, Validation loss: 0.736547733657062
Epoch 15: Train err: 0.222875, Train loss: 0.47256288361549376 |Validation
err: 0.3695, Validation loss: 0.7776702400296926
Epoch 16: Train err: 0.221125, Train loss: 0.45990034461021423 |Validation
err: 0.355, Validation loss: 0.7396496422588825
Epoch 17: Train err: 0.20275, Train loss: 0.4406240692138672 |Validation e
rr: 0.365, Validation loss: 0.7325921822339296
Epoch 18: Train err: 0.199125, Train loss: 0.4246277847290039 |Validation
err: 0.368, Validation loss: 0.7683038730174303
Epoch 19: Train err: 0.19925, Train loss: 0.44107814073562623 |Validation
err: 0.402, Validation loss: 0.9193822182714939
Epoch 20: Train err: 0.202875, Train loss: 0.43910927057266236 |Validation
err: 0.3805, Validation loss: 0.8345869425684214
Epoch 21: Train err: 0.17425, Train loss: 0.3892930691242218 |Validation e
rr: 0.373, Validation loss: 0.8001550659537315
Epoch 22: Train err: 0.170625, Train loss: 0.39342616057395935 |Validation
err: 0.387, Validation loss: 0.8944555316120386
Epoch 23: Train err: 0.155, Train loss: 0.3608554136753082 |Validation er
r: 0.3735, Validation loss: 0.8506541401147842
Epoch 24: Train err: 0.1575, Train loss: 0.3635837936401367 |Validation er
r: 0.3845, Validation loss: 0.8902523461729288
Epoch 25: Train err: 0.148, Train loss: 0.33989807045459747 |Validation er
r: 0.365, Validation loss: 0.8587767006829381
Epoch 26: Train err: 0.141625, Train loss: 0.33313262128829957 |Validation
err: 0.3835, Validation loss: 0.946588534861803
Epoch 27: Train err: 0.149, Train loss: 0.34568459177017213 |Validation er
r: 0.375, Validation loss: 0.953350642696023
Epoch 28: Train err: 0.122, Train loss: 0.30099490118026734 |Validation er
r: 0.3705, Validation loss: 1.0178348310291767
Epoch 29: Train err: 0.122375, Train loss: 0.2939163398146629 |Validation
err: 0.3765, Validation loss: 1.0477196499705315
Epoch 30: Train err: 0.1215, Train loss: 0.29651726484298707 |Validation e
```

```
rr: 0.375, Validation loss: 1.0573083329945803
Finished Training
Total time elapsed: 101.46 seconds
```





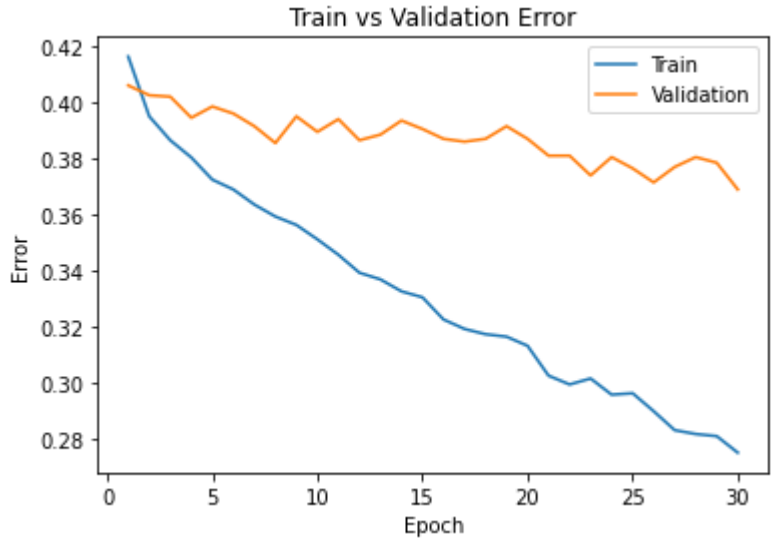

```
Files already downloaded and verified
Files already downloaded and verified
Test classification error:  0.374 Test loss:  1.0165240038186312
```

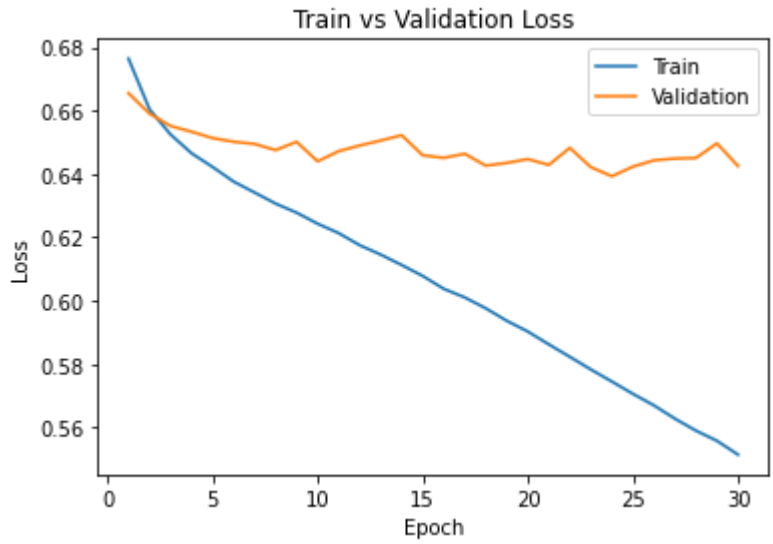
In [105]:

```
pigeon = Pigeon()
train_net(pigeon, "/content/gdrive/My Drive/Colab Notebooks/APS360_labs/Lab2_Part5f2",
batch_size=64, learning_rate=0.001, num_epochs=30)

model_path = get_model_name("pigeon", batch_size=64, learning_rate=0.001, epoch=29)
path = "/content/gdrive/My Drive/Colab Notebooks/APS360_labs/Lab2_Part5f2"
path_large = "/".join([path, model_path])
plot_training_curve(path_large)

train_loader, val_loader, test_loader, classes = get_data_loader(
    target_classes=["cat", "dog"],
    batch_size=64)
criterion = nn.BCEWithLogitsLoss()
err, loss = evaluate(pigeon, test_loader, criterion)
print("Test classification error: ", err, "Test loss: ", loss)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.41625, Train loss: 0.6764450631141663 |Validation er
r: 0.406, Validation loss: 0.665509482845664
Epoch 2: Train err: 0.395, Train loss: 0.660437177181244 |Validation err:
0.4025, Validation loss: 0.6590640060603619
Epoch 3: Train err: 0.3865, Train loss: 0.6524986891746521 |Validation er
r: 0.402, Validation loss: 0.655168367549777
Epoch 4: Train err: 0.380375, Train loss: 0.6465515723228454 |Validation e
rr: 0.3945, Validation loss: 0.6533493902534246
Epoch 5: Train err: 0.3725, Train loss: 0.6422333335876464 |Validation er
r: 0.3985, Validation loss: 0.6513529289513826
Epoch 6: Train err: 0.369, Train loss: 0.6376132864952088 |Validation err:
0.396, Validation loss: 0.6502019558101892
Epoch 7: Train err: 0.363625, Train loss: 0.6341570353507996 |Validation e
rr: 0.3915, Validation loss: 0.6494879070669413
Epoch 8: Train err: 0.359375, Train loss: 0.6306609735488892 |Validation e
rr: 0.3855, Validation loss: 0.6475583929568529
Epoch 9: Train err: 0.356375, Train loss: 0.6277895903587342 |Validation e
rr: 0.395, Validation loss: 0.650152612477541
Epoch 10: Train err: 0.35125, Train loss: 0.6243016796112061 |Validation e
rr: 0.3895, Validation loss: 0.6439921893179417
Epoch 11: Train err: 0.34575, Train loss: 0.6213266921043396 |Validation e
rr: 0.394, Validation loss: 0.6472037974745035
Epoch 12: Train err: 0.339375, Train loss: 0.6175209088325501 |Validation
err: 0.3865, Validation loss: 0.6489908918738365
Epoch 13: Train err: 0.337, Train loss: 0.6145457816123963 |Validation er
r: 0.3885, Validation loss: 0.6505636982619762
Epoch 14: Train err: 0.33275, Train loss: 0.6112585949897766 |Validation e
rr: 0.3935, Validation loss: 0.6522505097091198
Epoch 15: Train err: 0.330625, Train loss: 0.6078039398193359 |Validation
err: 0.3905, Validation loss: 0.6459012012928724
Epoch 16: Train err: 0.32275, Train loss: 0.6037393288612366 |Validation e
rr: 0.387, Validation loss: 0.6451230645179749
Epoch 17: Train err: 0.319375, Train loss: 0.6010189573764801 |Validation
err: 0.386, Validation loss: 0.6463730745017529
Epoch 18: Train err: 0.3175, Train loss: 0.597583294391632 |Validation er
r: 0.387, Validation loss: 0.6426416672766209
Epoch 19: Train err: 0.316625, Train loss: 0.5936384677886963 |Validation
err: 0.3915, Validation loss: 0.6435028482228518
Epoch 20: Train err: 0.313375, Train loss: 0.5902498784065247 |Validation
err: 0.387, Validation loss: 0.6447047479450703
Epoch 21: Train err: 0.30275, Train loss: 0.5861937024593353 |Validation e
rr: 0.381, Validation loss: 0.6428706608712673
Epoch 22: Train err: 0.299625, Train loss: 0.5822605903148651 |Validation
err: 0.381, Validation loss: 0.648263230919838
Epoch 23: Train err: 0.30175, Train loss: 0.5782670044898987 |Validation e
rr: 0.374, Validation loss: 0.6421573963016272
Epoch 24: Train err: 0.296, Train loss: 0.5744593787193298 |Validation er
r: 0.3805, Validation loss: 0.6392866000533104
Epoch 25: Train err: 0.2965, Train loss: 0.570579559803009 |Validation er
r: 0.3765, Validation loss: 0.6423205621540546
Epoch 26: Train err: 0.290125, Train loss: 0.5669271087646485 |Validation
err: 0.3715, Validation loss: 0.644320173189044
Epoch 27: Train err: 0.283375, Train loss: 0.5627356112003327 |Validation
err: 0.377, Validation loss: 0.6448737625032663
Epoch 28: Train err: 0.282, Train loss: 0.5589476592540741 |Validation er
r: 0.3805, Validation loss: 0.6450648698955774
Epoch 29: Train err: 0.28125, Train loss: 0.5557402448654175 |Validation e
rr: 0.3785, Validation loss: 0.649680720642209
Epoch 30: Train err: 0.275375, Train loss: 0.5513773720264434 |Validation
```

```
err: 0.369, Validation loss: 0.6425056587904692
Finished Training
Total time elapsed: 106.81 seconds
```





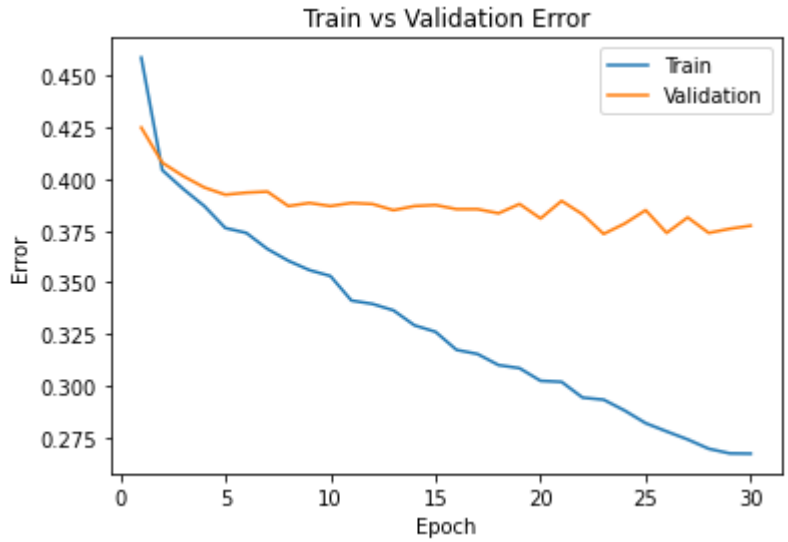

```
Files already downloaded and verified
Files already downloaded and verified
Test classification error:  0.3565 Test loss:  0.6349616460502148
```

In [106]:

```
pigeon = Pigeon()
train_net(pigeon, "/content/gdrive/My Drive/Colab Notebooks/APS360_labs/Lab2_Part5f3",
batch_size=512, learning_rate=0.01, num_epochs=30)

model_path = get_model_name("pigeon", batch_size=512, learning_rate=0.01, epoch=29)
path = "/content/gdrive/My Drive/Colab Notebooks/APS360_labs/Lab2_Part5f3"
path_large = "/".join([path, model_path])
plot_training_curve(path_large)

train_loader, val_loader, test_loader, classes = get_data_loader(
    target_classes=["cat", "dog"],
    batch_size=512)
criterion = nn.BCEWithLogitsLoss()
err, loss = evaluate(pigeon, test_loader, criterion)
print("Test classification error: ", err, "Test loss: ", loss)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.458875, Train loss: 0.6846102252602577 |Validation e
rr: 0.425, Validation loss: 0.6698803454637527
Epoch 2: Train err: 0.404375, Train loss: 0.6647883430123329 |Validation e
rr: 0.408, Validation loss: 0.6621970534324646
Epoch 3: Train err: 0.39525, Train loss: 0.6561560295522213 |Validation er
r: 0.4015, Validation loss: 0.6591369658708572
Epoch 4: Train err: 0.387, Train loss: 0.6501622721552849 |Validation err:
0.396, Validation loss: 0.6558480560779572
Epoch 5: Train err: 0.376375, Train loss: 0.6448182426393032 |Validation e
rr: 0.3925, Validation loss: 0.6536340117454529
Epoch 6: Train err: 0.374, Train loss: 0.6414279527962208 |Validation err:
0.3935, Validation loss: 0.6541043221950531
Epoch 7: Train err: 0.36625, Train loss: 0.6365129575133324 |Validation er
r: 0.394, Validation loss: 0.6519776880741119
Epoch 8: Train err: 0.3605, Train loss: 0.6315649561583996 |Validation er
r: 0.387, Validation loss: 0.6507265120744705
Epoch 9: Train err: 0.356, Train loss: 0.6282427720725536 |Validation err:
0.3885, Validation loss: 0.6494091302156448
Epoch 10: Train err: 0.353125, Train loss: 0.6249721609055996 |Validation
err: 0.387, Validation loss: 0.6504352688789368
Epoch 11: Train err: 0.34125, Train loss: 0.6205920353531837 |Validation e
rr: 0.3885, Validation loss: 0.6483761519193649
Epoch 12: Train err: 0.339625, Train loss: 0.6157152615487576 |Validation
err: 0.388, Validation loss: 0.6486307084560394
Epoch 13: Train err: 0.3365, Train loss: 0.6126304492354393 |Validation er
r: 0.385, Validation loss: 0.6473622024059296
Epoch 14: Train err: 0.32925, Train loss: 0.6073738671839237 |Validation e
rr: 0.387, Validation loss: 0.6483979374170303
Epoch 15: Train err: 0.326125, Train loss: 0.6038428582251072 |Validation
err: 0.3875, Validation loss: 0.6470333784818649
Epoch 16: Train err: 0.317375, Train loss: 0.6000079661607742 |Validation
err: 0.3855, Validation loss: 0.6460998058319092
Epoch 17: Train err: 0.315375, Train loss: 0.5960968025028706 |Validation
err: 0.3855, Validation loss: 0.6472584903240204
Epoch 18: Train err: 0.31, Train loss: 0.5906448103487492 |Validation err:
0.3835, Validation loss: 0.6470160335302353
Epoch 19: Train err: 0.3085, Train loss: 0.5873539298772812 |Validation er
r: 0.388, Validation loss: 0.6454845517873764
Epoch 20: Train err: 0.302375, Train loss: 0.5827141664922237 |Validation
err: 0.381, Validation loss: 0.6458163559436798
Epoch 21: Train err: 0.301875, Train loss: 0.5787002071738243 |Validation
err: 0.3895, Validation loss: 0.6502161473035812
Epoch 22: Train err: 0.29425, Train loss: 0.5759879983961582 |Validation e
rr: 0.383, Validation loss: 0.6513129770755768
Epoch 23: Train err: 0.29325, Train loss: 0.5682170391082764 |Validation e
rr: 0.3735, Validation loss: 0.6469174474477768
Epoch 24: Train err: 0.288, Train loss: 0.5627313666045666 |Validation er
r: 0.3785, Validation loss: 0.6459511816501617
Epoch 25: Train err: 0.281875, Train loss: 0.5592847689986229 |Validation
err: 0.385, Validation loss: 0.6529080420732498
Epoch 26: Train err: 0.277875, Train loss: 0.5547955296933651 |Validation
err: 0.374, Validation loss: 0.647481232881546
Epoch 27: Train err: 0.274, Train loss: 0.5487863719463348 |Validation er
r: 0.3815, Validation loss: 0.6501641422510147
Epoch 28: Train err: 0.2695, Train loss: 0.5457294136285782 |Validation er
r: 0.374, Validation loss: 0.6527189910411835
Epoch 29: Train err: 0.26725, Train loss: 0.5397161208093166 |Validation e
rr: 0.376, Validation loss: 0.6554051488637924
Epoch 30: Train err: 0.267125, Train loss: 0.5372924357652664 |Validation
```

```
err: 0.3775, Validation loss: 0.6504351943731308
Finished Training
Total time elapsed: 92.95 seconds
```




```
Files already downloaded and verified
Files already downloaded and verified
Test classification error:  0.3655 Test loss:  0.6465431749820709
```

If I apply the ANN model, by changing different parameters the best test classification error I can get is 0.3565, which is higher than the result obtained using CNN model (0.341). So it is showed the ANN model does not work as well as the CNN model in classifying dog and cat, CNN is the more suitable model to deal with this type of problem.

In [111]:

```
%%shell
jupyter nbconvert --to html "/Lab2_Cats_vs_Dogs.ipynb"
```

```
[NbConvertApp] Converting notebook /Lab2_Cats_vs_Dogs.ipynb to html
[NbConvertApp] Writing 1133137 bytes to /Lab2_Cats_vs_Dogs.html
```

Out[111]: