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/1ucApqnMP7FwscKX1wRMTfhTon_wsUDWO

```
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.

```
###############################################################################
# Data Loading

def get_relevant_indices(dataset, classes, target_classes):
    """ Return the indices for datapoints in the dataset that belongs to the
    desired target classes, a subset of all possible classes.

    Args:
        dataset: Dataset object
        classes: A list of strings denoting the name of each class
        target_classes: A list of strings denoting the name of desired classes
                        Should be a subset of the 'classes'
    Returns:
        indices: list of indices that have labels corresponding to one of the
                 target classes
    """
    indices = []
    for i in range(len(dataset)):
        # Check if the label is in the target classes
        label_index = dataset[i][1] # ex: 3
        label_class = classes[label_index] # ex: 'cat'
        if label_class in target_classes:
            indices.append(i)
    return indices

def get_data_loader(target_classes, batch_size):
    """ Loads images of cats and dogs, splits the data into training, validation
    and testing datasets. Returns data loaders for the three preprocessed datasets.

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

    Returns:
        train_loader: iterable training dataset organized according to batch size
        val_loader: iterable validation dataset organized according to batch size
        test_loader: iterable testing dataset organized according to batch size
        classes: A list of strings denoting the name of each class
    """

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

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

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

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

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

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

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

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

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

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

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

Part 1. Visualizing the Data [7 pt]

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

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

```
# This will download the CIFAR-10 dataset to a folder called "data"
# the first time you run this code.
train_loader, val_loader, test_loader, classes = get_data_loader(
    target_classes=["cat", "dog"],
    batch_size=1) # One image per batch
```

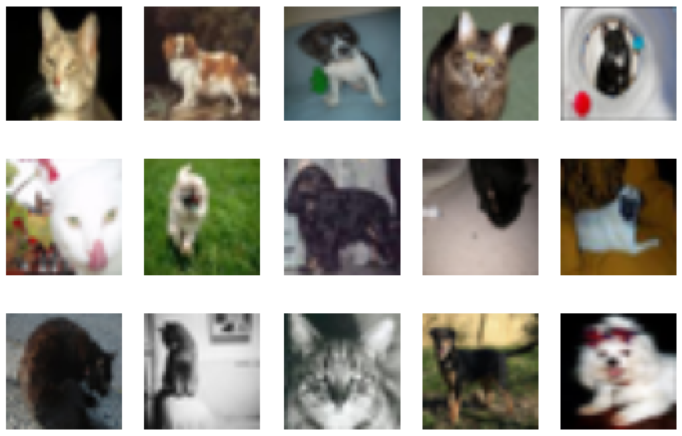
```
Files already downloaded and verified
Files already downloaded and verified
```

Part (a) -- 1 pt

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

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

```
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?

```
# Find individual cat and dog classes for every example
cat_train, cat_test, cat_validate = 0, 0, 0
dog_train, dog_test, dog_validate = 0, 0, 0

for images, labels in train_loader:
  for label in labels:
    if label == 5:
      dog_train = dog_train + 1
    elif label == 3:
      cat_train = cat_train + 1

for images, labels in test_loader:
  for label in labels:
    if label == 5:
      dog_test = dog_test + 1
    elif label == 3:
      cat_test = cat_test + 1

for images, labels in val_loader:
  for label in labels:
    if label == 5:
      dog_validate = dog_validate + 1
    elif label == 3:
      cat_validate = cat_validate + 1

# The number of training cat and dog examples
print("The number of training examples for the cat classes is: " + str(cat_train))
print("The number of training examples for the dog classes is: " + str(dog_train))

# The number of validation cat and dog examples
print("The number of validation examples for the cat classes is: " + str(cat_validate))
print("The number of validation examples for the dog classes is: " + str(dog_validate))

# The number of test cat and dog examples
print("The number of test examples for the cat classes is: " + str(cat_test))
print("The number of test examples for the dog classes is: " + str(dog_test))

# The number of training examples
print("The number of training examples for the combined cat and dog classes is: " + str(len(t
# The number of validation examples
print("The number of validation examples for the combined cat and dog classes is: " + str(len
# The number of test examples
print("The number of validation examples for the combined cat and dog classes is: " + str(len
```

```
The number of training examples for the cat classes is: 4018
The number of training examples for the dog classes is: 3982
The number of validation examples for the cat classes is: 982
The number of validation examples for the dog classes is: 1018
The number of test examples for the cat classes is: 1000
The number of test examples for the dog classes is: 1000
The number of training examples for the combined cat and dog classes is: 8000
The number of validation examples for the combined cat and dog classes is: 2000
The number of validation examples for the combined cat and dog classes is: 2000
```

Part (c) -- 3pt

Why do we need a validation set when training our model? What happens if we judge the

The validation set is needed to judge the performance of our models. The validation set is new data as opposed to the training set which is used to train the data. Thus, we're able to test our model more, and tune the hyperparameters to get more optimal results. We can also ensure that the model isn't overfitted or underfitted to the training set with the performance we obtain from the validation set. Judging the performance of our models using the training set instead of the validation set doesn't give us an accurate representation of how our model is actually performing. In reality, our model could have a training set loss/error of 0, which implies that our model is perfect, but this could just mean that our model is just memorizing the training set instead of finding patterns within the data. All of these concerns can be debunked when testing the model on the validation 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.

```
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


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


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?)

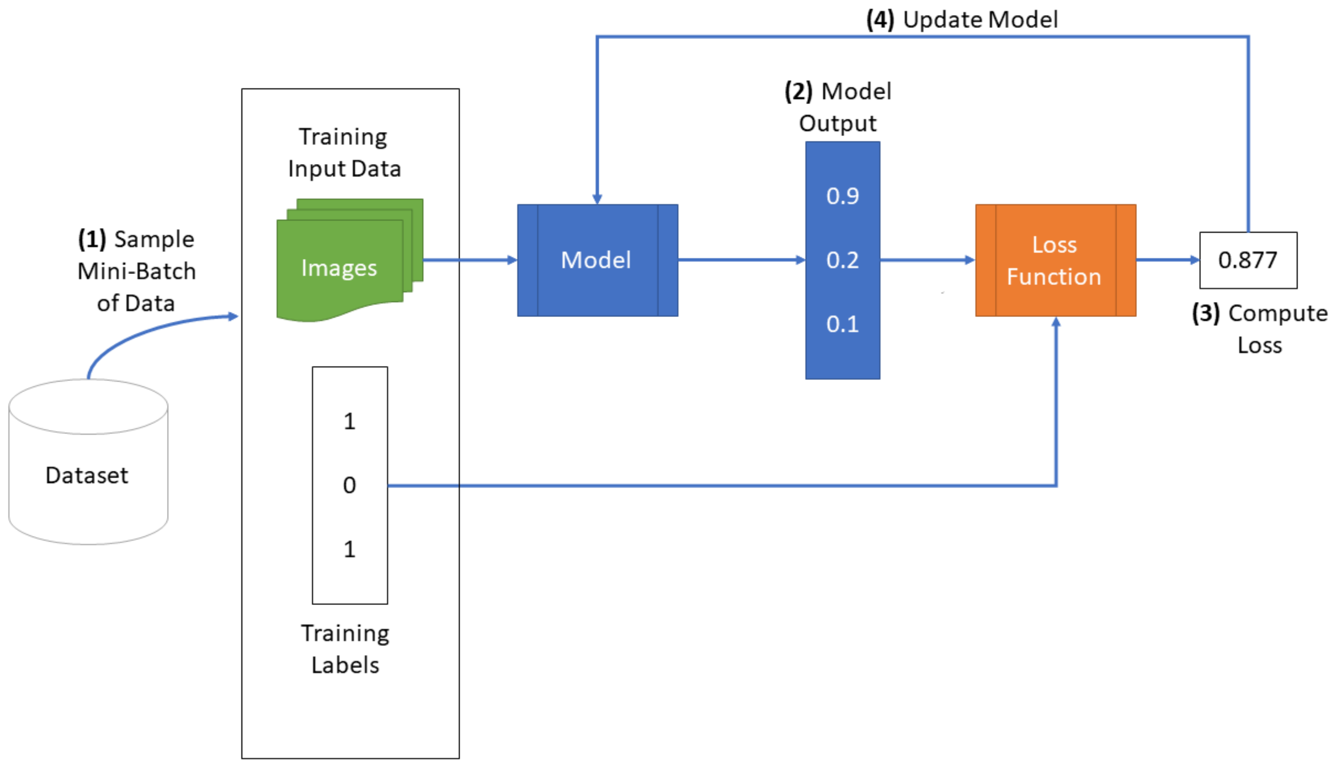
```
small_parameters = 0
large_parameters = 0

# Small net parameters
for param in small_net.parameters():
  print(param.shape)
  temp = 1
  for x in param.shape:
    temp = temp * x
  small_parameters = small_parameters + temp

print("The total number of parameters in small_net are: " + str(small_parameters))

print("\n")

# Large net parameters
for param in large_net.parameters():
  print(param.shape)
  temp = 1
```

```
    for x in param.shape:
      temp = temp * x
    large_parameters = large_parameters + temp

  print("The total number of parameters in large_net are: " + str(large_parameters))
```

```
torch.Size([5, 3, 3, 3])
torch.Size([5])
torch.Size([1, 245])
torch.Size([1])
The total number of parameters in small_net are: 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 large_net are: 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:



```
def train_net(net, batch_size=64, learning_rate=0.01, num_epochs=30):
    ########################################################################
    # Train a classifier on cats vs dogs
    target_classes = ["cat", "dog"]
    ########################################################################
    # Fixed PyTorch random seed for reproducible result
    torch.manual_seed(1000)
    ########################################################################
    # Obtain the PyTorch data loader objects to load batches of the datasets
    train_loader, val_loader, test_loader, classes = get_data_loader(
            target_classes, batch_size)
    ########################################################################
    # Define the Loss function and optimizer
    # The loss function will be Binary Cross Entropy (BCE). In this case we
    # will use the BCEWithLogitsLoss which takes unnormalized output from
    # the neural network and scalar label.
    # Optimizer will be SGD with Momentum.
    criterion = nn.BCEWithLogitsLoss()
    optimizer = optim.SGD(net.parameters(), lr=learning_rate, momentum=0.9)
    ########################################################################
    # Set up some numpy arrays to store the training/test loss/erruracy
    train_err = np.zeros(num_epochs)
    train_loss = np.zeros(num_epochs)
    val_err = np.zeros(num_epochs)
    val_loss = np.zeros(num_epochs)
    ########################################################################
    # Train the network
    # Loop over the data iterator and sample a new batch of training data
    # Get the output from the network, and optimize our loss function.
    start_time = time.time()
    for epoch in range(num_epochs):  # loop over the dataset multiple times
        total_train_loss = 0.0
        total_train_err = 0.0
        total_epoch = 0
        for i, data in enumerate(train_loader, 0):
            # Get the inputs
            inputs, labels = data
            labels = normalize_label(labels) # Convert labels to 0/1
            # Zero the parameter gradients
            optimizer.zero_grad()
            # Forward pass, backward pass, and optimize
            outputs = net(inputs)
            loss = criterion(outputs, labels.float())
            loss.backward()
            optimizer.step()
            # Calculate the statistics
            corr = (outputs > 0.0).squeeze().long() != labels
            total_train_err += int(corr.sum())
            total_train_loss += loss.item()
            total_epoch += len(labels)
        train_err[epoch] = float(total_train_err) / total_epoch
        train_loss[epoch] = float(total_train_loss) / (i+1)
        val_err[epoch], val_loss[epoch] = evaluate(net, val_loader, criterion)
        print(("Epoch {}: Train err: {}, Train loss: {} |"+
               "Validation err: {}, Validation loss: {}").format(
                   epoch + 1,
                   train_err[epoch],
                   train_loss[epoch],
                   val_err[epoch],
                   val_loss[epoch]))
        # Save the current model (checkpoint) to a file
        model_path = get_model_name(net.name, batch_size, learning_rate, epoch)
        torch.save(net.state_dict(), model_path)
    print('Finished Training')
    end_time = time.time()
    elapsed_time = end_time - start_time
    print("Total time elapsed: {:.2f} seconds".format(elapsed_time))
    # Write the train/test loss/err into CSV file for plotting later
    epochs = np.arange(1, num_epochs + 1)
    np.savetxt("{}_train_err.csv".format(model_path), train_err)
    np.savetxt("{}_train_loss.csv".format(model_path), train_loss)
    np.savetxt("{}_val_err.csv".format(model_path), val_err)
    np.savetxt("{}_val_loss.csv".format(model_path), val_loss)
```

Part (b) -- 1pt

The parameters to the function `train_net` are hyperparameters of our neural network. We made these hyperparameters easy to modify so that we can tune them later on.

What are the default values of the parameters `batch_size`, `learning_rate`, and `num_epochs`?

The default value of the batch_size is 64. The default value of the learning_rate is 0.01. The default value of the 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.

```
train_net(small_net, batch_size=64, learning_rate=0.01, num_epochs=5)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.431, Train loss: 0.674514232635498 |Validation err: 0.3765, Valid
Epoch 2: Train err: 0.368625, Train loss: 0.6483634223937988 |Validation err: 0.3895, Va
Epoch 3: Train err: 0.349375, Train loss: 0.6327687554359436 |Validation err: 0.3535, Va
```

```
    Epoch 4: Train err: 0.334125, Train loss: 0.6138443050384521 |Validation err: 0.357, Va
    Epoch 5: Train err: 0.322, Train loss: 0.6009030966758728 |Validation err: 0.3315, Valid
    Finished Training
    Total time elapsed: 23.89 seconds
```

- 5 model checkpoints are saved after the completion of each epoch
- A file with the model's training set error values after each epoch will be saved
- A file with the model's validation set error values after each epoch will be saved
- A file with the model's training set loss values after each epoch will be saved
- A file with the model's validation set loss values after each epoch will be saved

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?

```
# 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')
```

```
    Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mou
```

```
train_net(small_net)
train_net(large_net)
```

```
    Epoch 10: Train err: 0.288625, Train loss: 0.5591275158834981 |Validation err: 0.3095
    Epoch 11: Train err: 0.288, Train loss: 0.5594752118587494 |Validation err: 0.3125, V
    Epoch 12: Train err: 0.281, Train loss: 0.553269855260849 |Validation err: 0.311, Val
    Epoch 13: Train err: 0.28675, Train loss: 0.5568746740818024 |Validation err: 0.309,
    Epoch 14: Train err: 0.276875, Train loss: 0.5496627376079559 |Validation err: 0.3085
    Epoch 15: Train err: 0.27625, Train loss: 0.5496549184322357 |Validation err: 0.306,
    Epoch 16: Train err: 0.281625, Train loss: 0.5551487491130829 |Validation err: 0.3125
    Epoch 17: Train err: 0.281625, Train loss: 0.5518249912261963 |Validation err: 0.3085
    Epoch 18: Train err: 0.28075, Train loss: 0.5481037192344665 |Validation err: 0.3045,
    Epoch 19: Train err: 0.276375, Train loss: 0.5453145859241486 |Validation err: 0.3125
    Epoch 20: Train err: 0.273625, Train loss: 0.5442279896736145 |Validation err: 0.3175
```

```
Epoch 21: Train err: 0.283375, Train loss: 0.5486180992126465 |Validation err: 0.3095
Epoch 22: Train err: 0.280375, Train loss: 0.5450750262737274 |Validation err: 0.3055
Epoch 23: Train err: 0.27775, Train loss: 0.5452287113666534 |Validation err: 0.31, V
Epoch 24: Train err: 0.276, Train loss: 0.542751472234726 |Validation err: 0.3055, Va
Epoch 25: Train err: 0.272625, Train loss: 0.540872226715088 |Validation err: 0.306,
Epoch 26: Train err: 0.274875, Train loss: 0.5414948894977569 |Validation err: 0.2985
Epoch 27: Train err: 0.276375, Train loss: 0.5410179054737091 |Validation err: 0.313,
Epoch 28: Train err: 0.279, Train loss: 0.5419466536045074 |Validation err: 0.313, Va
Epoch 29: Train err: 0.27725, Train loss: 0.5446690742969513 |Validation err: 0.3085,
Epoch 30: Train err: 0.271, Train loss: 0.5431376056671142 |Validation err: 0.3115, V
Finished Training
Total time elapsed: 145.43 seconds
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.473875, Train loss: 0.6913936295509339 |Validation err: 0.446,
Epoch 2: Train err: 0.444625, Train loss: 0.6843735795021058 |Validation err: 0.4265,
Epoch 3: Train err: 0.416875, Train loss: 0.673103590965271 |Validation err: 0.3795,
Epoch 4: Train err: 0.378625, Train loss: 0.65466805934906 |Validation err: 0.381, Va
Epoch 5: Train err: 0.3575, Train loss: 0.6382192006111145 |Validation err: 0.3615, V
Epoch 6: Train err: 0.344, Train loss: 0.6186007556915283 |Validation err: 0.3515, Va
Epoch 7: Train err: 0.335625, Train loss: 0.607559045791626 |Validation err: 0.3445,
Epoch 8: Train err: 0.31875, Train loss: 0.5870774610042572 |Validation err: 0.338, V
Epoch 9: Train err: 0.306125, Train loss: 0.5835310232639312 |Validation err: 0.318,
Epoch 10: Train err: 0.29475, Train loss: 0.5652833173274994 |Validation err: 0.291,
Epoch 11: Train err: 0.281375, Train loss: 0.5526398799419403 |Validation err: 0.293,
Epoch 12: Train err: 0.2735, Train loss: 0.5377612402439117 |Validation err: 0.299, V
Epoch 13: Train err: 0.267125, Train loss: 0.5273976798057556 |Validation err: 0.288,
Epoch 14: Train err: 0.2575, Train loss: 0.5140651280879974 |Validation err: 0.2975,
Epoch 15: Train err: 0.245, Train loss: 0.5030667924880982 |Validation err: 0.2855, V
Epoch 16: Train err: 0.246625, Train loss: 0.49436042857170104 |Validation err: 0.290
Epoch 17: Train err: 0.235125, Train loss: 0.48851676869392396 |Validation err: 0.29,
Epoch 18: Train err: 0.23125, Train loss: 0.46840514326095584 |Validation err: 0.2875
Epoch 19: Train err: 0.22075, Train loss: 0.4617292220592499 |Validation err: 0.288,
Epoch 20: Train err: 0.218, Train loss: 0.4509369087219238 |Validation err: 0.2925, V
Epoch 21: Train err: 0.208125, Train loss: 0.4383981740474701 |Validation err: 0.2985
Epoch 22: Train err: 0.202, Train loss: 0.4233166215419769 |Validation err: 0.289, Va
Epoch 23: Train err: 0.19, Train loss: 0.40923917067050936 |Validation err: 0.2935, V
Epoch 24: Train err: 0.17925, Train loss: 0.3979230933189392 |Validation err: 0.298,
Epoch 25: Train err: 0.170375, Train loss: 0.37815937650203707 |Validation err: 0.306
Epoch 26: Train err: 0.16125, Train loss: 0.36076728463172913 |Validation err: 0.3055
Epoch 27: Train err: 0.15325, Train loss: 0.3435948469638824 |Validation err: 0.298,
Epoch 28: Train err: 0.151, Train loss: 0.3367327309846878 |Validation err: 0.297, Va
Epoch 29: Train err: 0.141625, Train loss: 0.32155103266239166 |Validation err: 0.311
Epoch 30: Train err: 0.13875, Train loss: 0.3114397028684616 |Validation err: 0.302,
Finished Training
Total time elapsed: 154.19 seconds
```

The small net training took 145.43 seconds and the large net training took 154.19 seconds. The large net training took longer because it has an additonal layer and more paramaters which need to be updated during training.

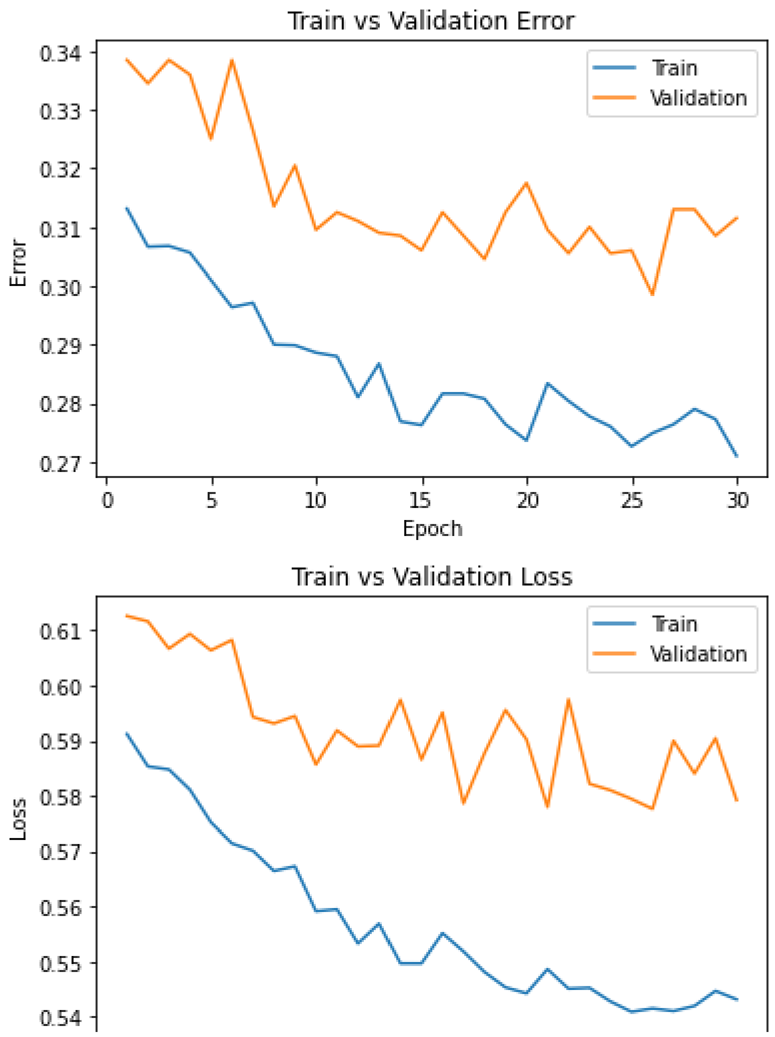
Part (e) - 2pt

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.

```
small_model_path = get_model_name("small", batch_size=64, learning_rate=0.01, epoch=29)
print("Small Network: ")
plot_training_curve(small_model_path)

large_model_path = get_model_name("large", batch_size=64, learning_rate=0.01, epoch=29)
print("Large Network: ")
plot_training_curve(large_model_path)
```

```
Small Network:
```



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.

One consistency I see with the training curve for errors and losses for both the small_net and large_net is that it is consistently decreasing as we increase the number of epochs. This is an indication that the parameters are being fine tuned in order to give us more accurate results. Although, it is notable that for the validation set in the large_net, it begins to plateau after 5 epochs. For the large_net, after 20 epochs, the validation set actually begins to increase; this is an indication of overfitting - the model memorizes the training set data too well and doesn't find patterns, so when it's tasked with classifying new data, it doesn't give accurate results as expected.

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.

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

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.47625, Train loss: 0.6928360013961792 |Validation err: 0.467, Vali
Epoch 2: Train err: 0.448625, Train loss: 0.6922589712142945 |Validation err: 0.4305, Va
Epoch 3: Train err: 0.43575, Train loss: 0.6916067280769348 |Validation err: 0.4285, Val
Epoch 4: Train err: 0.43, Train loss: 0.690861343383789 |Validation err: 0.424, Validati
Epoch 5: Train err: 0.434125, Train loss: 0.6899195008277893 |Validation err: 0.4195, Va
Epoch 6: Train err: 0.43575, Train loss: 0.6887411961555481 |Validation err: 0.4195, Val
Epoch 7: Train err: 0.437125, Train loss: 0.6873774147033691 |Validation err: 0.4185, Va
Epoch 8: Train err: 0.4375, Train loss: 0.6859278454780579 |Validation err: 0.412, Valic
Epoch 9: Train err: 0.424375, Train loss: 0.6844058036804199 |Validation err: 0.411, Val
Epoch 10: Train err: 0.424, Train loss: 0.6828502931594849 |Validation err: 0.408, Valic
Epoch 11: Train err: 0.425375, Train loss: 0.6812348766326904 |Validation err: 0.4125, \
Epoch 12: Train err: 0.42, Train loss: 0.6796319708824158 |Validation err: 0.4125, Valic
Epoch 13: Train err: 0.414875, Train loss: 0.6777918744087219 |Validation err: 0.415, Va
Epoch 14: Train err: 0.412375, Train loss: 0.6761112003326416 |Validation err: 0.412, Va
Epoch 15: Train err: 0.40925, Train loss: 0.674472680568695 |Validation err: 0.415, Vali
Epoch 16: Train err: 0.406375, Train loss: 0.6727448840141297 |Validation err: 0.4105, \
Epoch 17: Train err: 0.4015, Train loss: 0.6713076601028443 |Validation err: 0.4045, Val
Epoch 18: Train err: 0.3995, Train loss: 0.6696742882728577 |Validation err: 0.4055, Val
Epoch 19: Train err: 0.40075, Train loss: 0.6679086356163025 |Validation err: 0.396, Val
Epoch 20: Train err: 0.392375, Train loss: 0.665787980556488 |Validation err: 0.405, Val
Epoch 21: Train err: 0.38975, Train loss: 0.6646300601959229 |Validation err: 0.394, Val
Epoch 22: Train err: 0.388875, Train loss: 0.662373058795929 |Validation err: 0.393, Val
Epoch 23: Train err: 0.38425, Train loss: 0.6601516346931458 |Validation err: 0.3975, Va
Epoch 24: Train err: 0.382375, Train loss: 0.6584009389877319 |Validation err: 0.386, Va
```

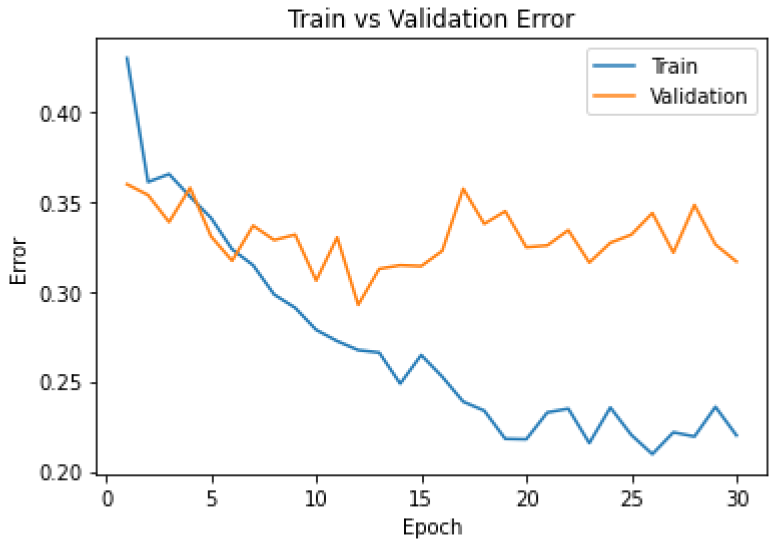
The large_net training with 0.01 learning rate took 154.19 seconds. The large_net training with 0.001 learning rate took 154.88 seconds. These training times are both similar. Some effects of lowering the learning rate are: the validation and training set curves for loss and error are almost identical if the learning rate is lower. The overfitting that we observed with the larger learning rate is avoided here. The loss and error values are overall higher with the lower learning late. As the validation curve is lower than the training curve for both loss and error with the lower learning rate, it might be slightly underfitted.

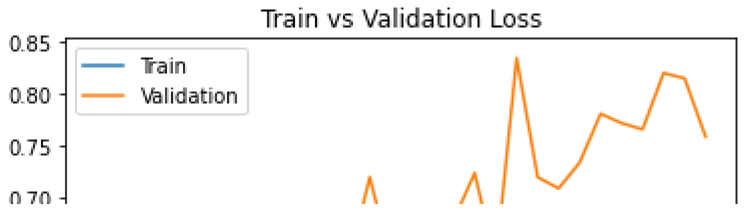
Part (b) - 3pt

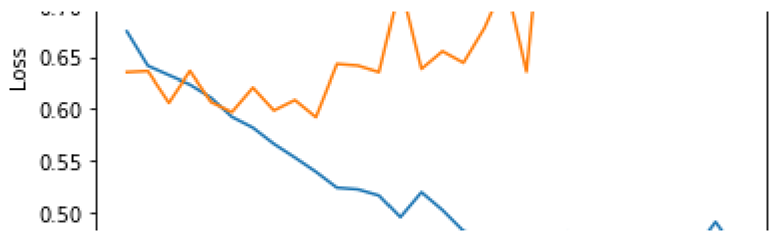
Train `large_net` with all default parameters, except set `learning_rate=0.1`. Does the model take longer/shorter to train? Plot the training curve. Describe the effect of *increasing* the learning rate.

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

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.4295, Train loss: 0.67437779712677 |Validation err: 0.3595, Valida
Epoch 2: Train err: 0.36075, Train loss: 0.6411805458068848 |Validation err: 0.3535, Va
Epoch 3: Train err: 0.365125, Train loss: 0.6321813461780548 |Validation err: 0.3385, Va
Epoch 4: Train err: 0.352625, Train loss: 0.6233456182479858 |Validation err: 0.3575, Va
Epoch 5: Train err: 0.34075, Train loss: 0.6108013873100281 |Validation err: 0.3305, Va
Epoch 6: Train err: 0.323375, Train loss: 0.5921835997104645 |Validation err: 0.317, Va
Epoch 7: Train err: 0.3145, Train loss: 0.5817317583560944 |Validation err: 0.3365, Val
Epoch 8: Train err: 0.29825, Train loss: 0.5660300073623658 |Validation err: 0.3285, Va
Epoch 9: Train err: 0.290875, Train loss: 0.552809501171112 |Validation err: 0.3315, Va
Epoch 10: Train err: 0.278625, Train loss: 0.539032607793808 |Validation err: 0.306, Va
Epoch 11: Train err: 0.272375, Train loss: 0.5236025826931 |Validation err: 0.33, Valida
Epoch 12: Train err: 0.267375, Train loss: 0.5220149435997009 |Validation err: 0.2925, \
Epoch 13: Train err: 0.266, Train loss: 0.5160510110855102 |Validation err: 0.3125, Val
Epoch 14: Train err: 0.24875, Train loss: 0.4951590054035187 |Validation err: 0.3145, Va
Epoch 15: Train err: 0.264625, Train loss: 0.519231944322586 |Validation err: 0.314, Va
Epoch 16: Train err: 0.252625, Train loss: 0.5020012385845184 |Validation err: 0.3225, \
Epoch 17: Train err: 0.23875, Train loss: 0.481714787364006 |Validation err: 0.357, Val
Epoch 18: Train err: 0.23375, Train loss: 0.47645506453514097 |Validation err: 0.3375, \
Epoch 19: Train err: 0.218125, Train loss: 0.45134368968009947 |Validation err: 0.3445,
Epoch 20: Train err: 0.217875, Train loss: 0.45516350817680357 |Validation err: 0.3245,
Epoch 21: Train err: 0.23275, Train loss: 0.47897080445289614 |Validation err: 0.3255, \
Epoch 22: Train err: 0.234875, Train loss: 0.4808810565471649 |Validation err: 0.334, Va
Epoch 23: Train err: 0.21575, Train loss: 0.4563647754192352 |Validation err: 0.316, Va
Epoch 24: Train err: 0.2355, Train loss: 0.47718250966072084 |Validation err: 0.327, Va
Epoch 25: Train err: 0.22025, Train loss: 0.4583414270877838 |Validation err: 0.3315, Va
Epoch 26: Train err: 0.209625, Train loss: 0.4519626965522766 |Validation err: 0.3435, \
Epoch 27: Train err: 0.22175, Train loss: 0.4636160457134247 |Validation err: 0.3215, Va
Epoch 28: Train err: 0.219375, Train loss: 0.46314777398109436 |Validation err: 0.348, \
Epoch 29: Train err: 0.235875, Train loss: 0.49053542733192446 |Validation err: 0.326, \
Epoch 30: Train err: 0.22, Train loss: 0.4623157248497009 |Validation err: 0.3165, Valic
Finished Training
Total time elapsed: 150.96 seconds
```







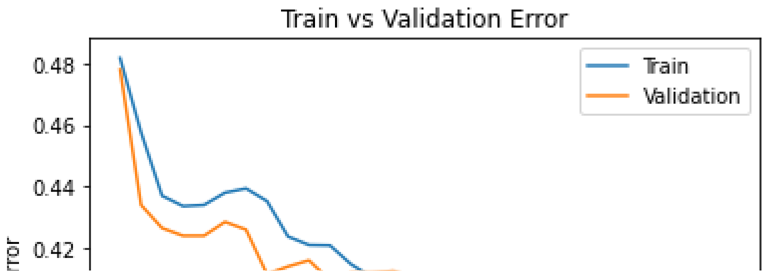
The large net with a lower learning rate took 150.96 seconds and the large net with default paramaters took 154.19 seconds. The times for both are similar. Some effects of increasing the learning rate are: fluctuation in both loss and error curves for train and validation as compared to a lower learning rate, an overfitted model with the number of epochs increasing, an exponential decrease in training error and loss, and an increase in validation loss as number of epochs are increasing.

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.

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

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.48175, Train loss: 0.6929379552602768 |Validation err: 0.478, Vali
Epoch 2: Train err: 0.457625, Train loss: 0.6924104019999504 |Validation err: 0.434, Va
Epoch 3: Train err: 0.437, Train loss: 0.6916500590741634 |Validation err: 0.4265, Valic
Epoch 4: Train err: 0.433625, Train loss: 0.6908449940383434 |Validation err: 0.424, Va
Epoch 5: Train err: 0.434, Train loss: 0.6896935552358627 |Validation err: 0.424, Valida
Epoch 6: Train err: 0.438, Train loss: 0.688353206962347 |Validation err: 0.4285, Valida
Epoch 7: Train err: 0.439375, Train loss: 0.6866871677339077 |Validation err: 0.426, Va
Epoch 8: Train err: 0.43525, Train loss: 0.6849770769476891 |Validation err: 0.4115, Va
Epoch 9: Train err: 0.42375, Train loss: 0.6832009293138981 |Validation err: 0.414, Vali
Epoch 10: Train err: 0.421, Train loss: 0.6811089366674423 |Validation err: 0.416, Valic
Epoch 11: Train err: 0.420875, Train loss: 0.6794026419520378 |Validation err: 0.4095, \
Epoch 12: Train err: 0.41475, Train loss: 0.6768048219382763 |Validation err: 0.412, Va
Epoch 13: Train err: 0.4105, Train loss: 0.6749702803790569 |Validation err: 0.412, Vali
Epoch 14: Train err: 0.407125, Train loss: 0.6730880849063396 |Validation err: 0.4125, \
Epoch 15: Train err: 0.4005, Train loss: 0.6706806942820549 |Validation err: 0.4105, Va
Epoch 16: Train err: 0.397625, Train loss: 0.6691771410405636 |Validation err: 0.405, Va
Epoch 17: Train err: 0.393875, Train loss: 0.6675694733858109 |Validation err: 0.401, Va
Epoch 18: Train err: 0.393, Train loss: 0.6648042872548103 |Validation err: 0.3945, Vali
Epoch 19: Train err: 0.38625, Train loss: 0.662746611982584 |Validation err: 0.388, Vali
Epoch 20: Train err: 0.38175, Train loss: 0.6596181839704514 |Validation err: 0.4005, Va
Epoch 21: Train err: 0.38575, Train loss: 0.6584899798035622 |Validation err: 0.3885, Va
Epoch 22: Train err: 0.378125, Train loss: 0.655123382806778 |Validation err: 0.3855, Va
Epoch 23: Train err: 0.372125, Train loss: 0.6508794128894806 |Validation err: 0.3835, \
Epoch 24: Train err: 0.37675, Train loss: 0.6488028429448605 |Validation err: 0.385, Va
Epoch 25: Train err: 0.368625, Train loss: 0.6445869170129299 |Validation err: 0.382, Va
Epoch 26: Train err: 0.372625, Train loss: 0.6428566053509712 |Validation err: 0.3745, \
Epoch 27: Train err: 0.359375, Train loss: 0.6372117549180984 |Validation err: 0.379, Va
Epoch 28: Train err: 0.35425, Train loss: 0.6337667480111122 |Validation err: 0.3695, Va
Epoch 29: Train err: 0.3535, Train loss: 0.6311353109776974 |Validation err: 0.366, Vali
Epoch 30: Train err: 0.353, Train loss: 0.6283832415938377 |Validation err: 0.3675, Vali
Finished Training
Total time elapsed: 137.11 seconds
```



The large net with a batch size of 512 took 137.11 seconds and the large net with default paramaters took 154.19 seconds. The higher batch size training time was quicker. This is because there is less iteration that takes place per epoch. The effects of increasing the bath size are: smoother training and validation curves, more time to decrease loss curves for training and validation data, steady decrease for training and validation curves as the number of epochs increases.

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.

```
large_net = LargeNet()
train_net(large_net, batch_size=16)
model_path_large = get_model_name("large", batch_size=16, learning_rate=0.01, epoch=29)
plot_training_curve(model_path_large)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.43175, Train loss: 0.6774994022846222 |Validation err: 0.382, Vali
Epoch 2: Train err: 0.369, Train loss: 0.639639899969101 |Validation err: 0.3465, Valida
Epoch 3: Train err: 0.34375, Train loss: 0.6098222947120666 |Validation err: 0.3325, Val
Epoch 4: Train err: 0.314375, Train loss: 0.5849691489338875 |Validation err: 0.34, Vali
Epoch 5: Train err: 0.301125, Train loss: 0.5689119303822517 |Validation err: 0.3125, Va
Epoch 6: Train err: 0.281, Train loss: 0.5452213581204415 |Validation err: 0.308, Valida
Epoch 7: Train err: 0.270875, Train loss: 0.5272981298565864 |Validation err: 0.307, Val
Epoch 8: Train err: 0.259375, Train loss: 0.5070905526578426 |Validation err: 0.313, Val
Epoch 9: Train err: 0.242375, Train loss: 0.4968344421982765 |Validation err: 0.313, Val
Epoch 10: Train err: 0.236375, Train loss: 0.4756101597249508 |Validation err: 0.297, Va
Epoch 11: Train err: 0.222125, Train loss: 0.4599769461452961 |Validation err: 0.2975, V
Epoch 12: Train err: 0.211, Train loss: 0.4454492371380329 |Validation err: 0.2995, Vali
Epoch 13: Train err: 0.19875, Train loss: 0.4245421719551086 |Validation err: 0.3075, Va
Epoch 14: Train err: 0.18675, Train loss: 0.4007472907453775 |Validation err: 0.3085, Va
Epoch 15: Train err: 0.1645, Train loss: 0.3759974058121443 |Validation err: 0.3105, Val
Epoch 16: Train err: 0.16125, Train loss: 0.3591455406397581 |Validation err: 0.3005, Va
Epoch 17: Train err: 0.15775, Train loss: 0.3463234790861607 |Validation err: 0.307, Val
Epoch 18: Train err: 0.141625, Train loss: 0.32175366275012496 |Validation err: 0.3195, 
Epoch 19: Train err: 0.13375, Train loss: 0.30618105667084455 |Validation err: 0.335, Va
Epoch 20: Train err: 0.126625, Train loss: 0.3029071792438626 |Validation err: 0.32, Val
Epoch 21: Train err: 0.12025, Train loss: 0.28682796490937473 |Validation err: 0.3205, V
```

The large net with a batch size of 16 took 215.88 seconds. The large net with default paramaters took 154.19 seconds. The large net with a smaller batch size took longer to train. This is because there are more iterations that take place per epoch. The effects of lowering the batch size are: low training loss and error, overfitting for validation curve for both loss and error - for error, the validation plateaurs whereas it consistently increases for loss as we increase the number of epochs. The validation curves in general are very poor in comparison with the training curves.

```
Total time elapsed: 215.88 seconds
```

Part 4. Hyperparameter Search [6 pt]

Part (a) - 2pt

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

I would choose the following hyperparameters:

1. Large Network
    - Validation and training curves for loss and error fluctuate less than the small net
2. Batch size = 512
    - There was no overfitting observerd

- The loss and error curves for both validation and training decrease steadily as we increase the number of epochs

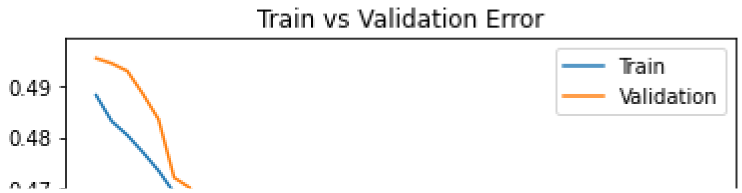
3. Epochs = 40
    - There was no overfitting observerd
4. Learning rate = 0.001
    - There was no overfitting observerd
    - The loss and error curves for both validation and training decrease steadily as we increase the number of epochs

Part (b) - 1pt

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

```
large_net = LargeNet()
train_net(large_net, batch_size= 512, learning_rate= 0.001, num_epochs=40)
large_model_path = get_model_name("large", batch_size= 512, learning_rate= 0.001, epoch=39)
plot_training_curve(large_model_path)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.48825, Train loss: 0.6930677480995655 |Validation err: 0.4955, Va
Epoch 2: Train err: 0.483125, Train loss: 0.692995510995388 |Validation err: 0.4945, Va
Epoch 3: Train err: 0.480375, Train loss: 0.6929280497133732 |Validation err: 0.493, Va
Epoch 4: Train err: 0.477, Train loss: 0.6928808391094208 |Validation err: 0.4885, Vali
Epoch 5: Train err: 0.473375, Train loss: 0.692774411290884 |Validation err: 0.4835, Va
Epoch 6: Train err: 0.469, Train loss: 0.6926896274089813 |Validation err: 0.472, Valid
Epoch 7: Train err: 0.46325, Train loss: 0.692620363086462 |Validation err: 0.47, Valid
Epoch 8: Train err: 0.46225, Train loss: 0.6925435550510883 |Validation err: 0.463, Val
Epoch 9: Train err: 0.459625, Train loss: 0.6924680322408676 |Validation err: 0.457, Va
Epoch 10: Train err: 0.458, Train loss: 0.6923965662717819 |Validation err: 0.4555, Val
Epoch 11: Train err: 0.454875, Train loss: 0.6923230737447739 |Validation err: 0.4505, V
Epoch 12: Train err: 0.4535, Train loss: 0.6922412514686584 |Validation err: 0.441, Val
Epoch 13: Train err: 0.450375, Train loss: 0.6921614557504654 |Validation err: 0.437, Va
Epoch 14: Train err: 0.44725, Train loss: 0.6921032443642616 |Validation err: 0.433, Va
Epoch 15: Train err: 0.449375, Train loss: 0.6920064650475979 |Validation err: 0.432, Va
Epoch 16: Train err: 0.44425, Train loss: 0.6919283680617809 |Validation err: 0.432, Va
Epoch 17: Train err: 0.441375, Train loss: 0.6918644718825817 |Validation err: 0.431, Va
Epoch 18: Train err: 0.438125, Train loss: 0.6917712315917015 |Validation err: 0.4295, V
Epoch 19: Train err: 0.436375, Train loss: 0.6917018257081509 |Validation err: 0.428, Va
Epoch 20: Train err: 0.436375, Train loss: 0.6915871091187 |Validation err: 0.4275, Val
Epoch 21: Train err: 0.437, Train loss: 0.6915052235126495 |Validation err: 0.4285, Val
Epoch 22: Train err: 0.438625, Train loss: 0.6914149634540081 |Validation err: 0.428, Va
Epoch 23: Train err: 0.436875, Train loss: 0.6912974379956722 |Validation err: 0.428, Va
Epoch 24: Train err: 0.436875, Train loss: 0.6912120543420315 |Validation err: 0.425, Va
Epoch 25: Train err: 0.435125, Train loss: 0.6910865269601345 |Validation err: 0.4255, V
Epoch 26: Train err: 0.434625, Train loss: 0.6910119205713272 |Validation err: 0.4245, V
Epoch 27: Train err: 0.43675, Train loss: 0.6909283325076103 |Validation err: 0.4265, Va
Epoch 28: Train err: 0.43575, Train loss: 0.6908275187015533 |Validation err: 0.4265, Va
Epoch 29: Train err: 0.436375, Train loss: 0.6906765103340149 |Validation err: 0.423, Va
Epoch 30: Train err: 0.435625, Train loss: 0.6905755028128624 |Validation err: 0.4235, V
Epoch 31: Train err: 0.435625, Train loss: 0.6905272789299488 |Validation err: 0.424, Va
Epoch 32: Train err: 0.435375, Train loss: 0.6903932206332684 |Validation err: 0.4245, V
Epoch 33: Train err: 0.434375, Train loss: 0.690232016146183 |Validation err: 0.425, Va
Epoch 34: Train err: 0.434, Train loss: 0.6901473812758923 |Validation err: 0.423, Vali
Epoch 35: Train err: 0.434, Train loss: 0.6899915188550949 |Validation err: 0.422, Vali
Epoch 36: Train err: 0.433125, Train loss: 0.6898530460894108 |Validation err: 0.4205, V
Epoch 37: Train err: 0.435875, Train loss: 0.689723439514637 |Validation err: 0.42, Val
Epoch 38: Train err: 0.435125, Train loss: 0.6896037347614765 |Validation err: 0.421, Va
Epoch 39: Train err: 0.436625, Train loss: 0.6894075199961662 |Validation err: 0.4175, V
Epoch 40: Train err: 0.435375, Train loss: 0.6893158182501793 |Validation err: 0.4195, V
Finished Training
Total time elapsed: 184.24 seconds
```



Part (c) - 2pt

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

Based on my result from Part(a), I notice that my training and validation error curves are both plateau until 20 epochs and then they rapidly decrease. My hyperparameters chosen generally overfit the data as the loss and error are higher than I'd want them to be. I want to try increasing the number of epochs as that would give us more iterations, keep the large network, and increase the learning rate while decreasing the batch size.

Hyperparameters chosen: Batch size = 256, learning rate = 0.005, epochs = 75, large network.

Part (d) - 1pt

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

```
large_net = LargeNet()
train_net(large_net, batch_size= 256, learning_rate= 0.005, num_epochs=75)
large_model_path = get_model_name("large", batch_size= 256, learning_rate= 0.005, epoch=74)
plot_training_curve(large_model_path)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.50225, Train loss: 0.6929918043315411 |Validation err: 0.4895, Va
Epoch 2: Train err: 0.48725, Train loss: 0.6922785621136427 |Validation err: 0.4675, Va
Epoch 3: Train err: 0.453125, Train loss: 0.6915246844291687 |Validation err: 0.434, Va
Epoch 4: Train err: 0.45225, Train loss: 0.690559322014451 |Validation err: 0.432, Vali
Epoch 5: Train err: 0.448375, Train loss: 0.6895249728113413 |Validation err: 0.432, Va
Epoch 6: Train err: 0.448125, Train loss: 0.6886843405663967 |Validation err: 0.428, Va
Epoch 7: Train err: 0.44375, Train loss: 0.687343480065465 |Validation err: 0.431, Vali
Epoch 8: Train err: 0.443125, Train loss: 0.6859446428716183 |Validation err: 0.426, Va
Epoch 9: Train err: 0.443625, Train loss: 0.6844949517399073 |Validation err: 0.4235, V
Epoch 10: Train err: 0.439375, Train loss: 0.6831344421952963 |Validation err: 0.4245, 
Epoch 11: Train err: 0.432125, Train loss: 0.6806647386401892 |Validation err: 0.4225, 
Epoch 12: Train err: 0.425875, Train loss: 0.6779768317937851 |Validation err: 0.417, V
Epoch 13: Train err: 0.42, Train loss: 0.6764416564255953 |Validation err: 0.409, Valid
Epoch 14: Train err: 0.41125, Train loss: 0.6726823877543211 |Validation err: 0.408, Va
Epoch 15: Train err: 0.40075, Train loss: 0.6688256449997425 |Validation err: 0.3985, V
Epoch 16: Train err: 0.38725, Train loss: 0.6637918706983328 |Validation err: 0.3955, V
Epoch 17: Train err: 0.3825, Train loss: 0.6603934373706579 |Validation err: 0.3945, Va
Epoch 18: Train err: 0.37325, Train loss: 0.6561635211110115 |Validation err: 0.379, Va
Epoch 19: Train err: 0.368875, Train loss: 0.648387698456645 |Validation err: 0.3865, V
Epoch 20: Train err: 0.357875, Train loss: 0.6418450940400362 |Validation err: 0.378, V
Epoch 21: Train err: 0.35525, Train loss: 0.6384237036108971 |Validation err: 0.369, Va
Epoch 22: Train err: 0.35325, Train loss: 0.6325437482446432 |Validation err: 0.3685, V
Epoch 23: Train err: 0.349375, Train loss: 0.6300209816545248 |Validation err: 0.358, V
Epoch 24: Train err: 0.35125, Train loss: 0.6276268921792507 |Validation err: 0.3565, V
Epoch 25: Train err: 0.34675, Train loss: 0.6194862555712461 |Validation err: 0.355, Va
Epoch 26: Train err: 0.343875, Train loss: 0.6183481328189373 |Validation err: 0.3485, 
Epoch 27: Train err: 0.34025, Train loss: 0.6146794985979795 |Validation err: 0.353, Va
Epoch 28: Train err: 0.33825, Train loss: 0.6137466821819544 |Validation err: 0.37, Val
Epoch 29: Train err: 0.33925, Train loss: 0.611265741288662 |Validation err: 0.352, Val
Epoch 30: Train err: 0.331625, Train loss: 0.6076459847390652 |Validation err: 0.351, V
Epoch 31: Train err: 0.328375, Train loss: 0.6045604012906551 |Validation err: 0.347, V
Epoch 32: Train err: 0.32875, Train loss: 0.6016061473637819 |Validation err: 0.339, Va
Epoch 33: Train err: 0.32425, Train loss: 0.6007344089448452 |Validation err: 0.3395, V
Epoch 34: Train err: 0.320375, Train loss: 0.5989894699305296 |Validation err: 0.35, Va
Epoch 35: Train err: 0.31575, Train loss: 0.5929280035197735 |Validation err: 0.3445, V
Epoch 36: Train err: 0.31925, Train loss: 0.5902958251535892 |Validation err: 0.3365, V
Epoch 37: Train err: 0.319875, Train loss: 0.5925506986677647 |Validation err: 0.3345, 
Epoch 38: Train err: 0.316375, Train loss: 0.5879574175924063 |Validation err: 0.331, V
Epoch 39: Train err: 0.311, Train loss: 0.5855193845927715 |Validation err: 0.331, Vali
Epoch 40: Train err: 0.308375, Train loss: 0.5795379020273685 |Validation err: 0.3375, 
Epoch 41: Train err: 0.30875, Train loss: 0.5827178079634905 |Validation err: 0.3275, V
Epoch 42: Train err: 0.30725, Train loss: 0.5774840898811817 |Validation err: 0.331, Va
Epoch 43: Train err: 0.301625, Train loss: 0.575823824852705 |Validation err: 0.322, Va
Epoch 44: Train err: 0.3015, Train loss: 0.5732897259294987 |Validation err: 0.332, Val
Epoch 45: Train err: 0.299, Train loss: 0.5690470151603222 |Validation err: 0.331, Vali
Epoch 46: Train err: 0.2995, Train loss: 0.5708411745727062 |Validation err: 0.316, Val
Epoch 47: Train err: 0.293125, Train loss: 0.5654510362073779 |Validation err: 0.3185, 
Epoch 48: Train err: 0.29425, Train loss: 0.5594953894615173 |Validation err: 0.3205, V
Epoch 49: Train err: 0.29075, Train loss: 0.5586610324680805 |Validation err: 0.3165, V
Epoch 50: Train err: 0.2895, Train loss: 0.5552052650600672 |Validation err: 0.3145, Va
Epoch 51: Train err: 0.28925, Train loss: 0.5549077354371548 |Validation err: 0.3295, V
Epoch 52: Train err: 0.287625, Train loss: 0.5605413280427456 |Validation err: 0.317, V
Epoch 53: Train err: 0.282375, Train loss: 0.553272657096386 |Validation err: 0.323, Va
Epoch 54: Train err: 0.281125, Train loss: 0.5468429587781429 |Validation err: 0.3075, 
Epoch 55: Train err: 0.276375, Train loss: 0.5459079453721642 |Validation err: 0.312, Va
Epoch 56: Train err: 0.2715, Train loss: 0.5411496963351965 |Validation err: 0.322, Vali
Epoch 57: Train err: 0.274625, Train loss: 0.5408895323053002 |Validation err: 0.3075, \
Epoch 58: Train err: 0.2775, Train loss: 0.5385762108489871 |Validation err: 0.307, Vali
Epoch 59: Train err: 0.275625, Train loss: 0.5378871727734804 |Validation err: 0.3055, \
Epoch 60: Train err: 0.275625, Train loss: 0.5361381564289331 |Validation err: 0.308, Va
Epoch 61: Train err: 0.26425, Train loss: 0.5286107268184423 |Validation err: 0.3065, Va
Epoch 62: Train err: 0.269625, Train loss: 0.5307895923033357 |Validation err: 0.307, Va
Epoch 63: Train err: 0.269375, Train loss: 0.5279002841562033 |Validation err: 0.3045, \
Epoch 64: Train err: 0.265375, Train loss: 0.5220356099307537 |Validation err: 0.304, Va
Epoch 65: Train err: 0.259875, Train loss: 0.518950967118144 |Validation err: 0.308, Val
Epoch 66: Train err: 0.260125, Train loss: 0.5146753611043096 |Validation err: 0.298, Va
Epoch 67: Train err: 0.25325, Train loss: 0.5101203173398972 |Validation err: 0.3115, Va
Epoch 68: Train err: 0.259625, Train loss: 0.5151218203827739 |Validation err: 0.3145, \
Epoch 69: Train err: 0.25275, Train loss: 0.512162764556706 |Validation err: 0.3025, Val
Epoch 70: Train err: 0.24675, Train loss: 0.5013926783576608 |Validation err: 0.303, Val
Epoch 71: Train err: 0.247625, Train loss: 0.5000795731320977 |Validation err: 0.314, Va
Epoch 72: Train err: 0.24825, Train loss: 0.502711039967835 |Validation err: 0.308, Vali
Epoch 73: Train err: 0.24375, Train loss: 0.4973066998645663 |Validation err: 0.3035, Va
Epoch 74: Train err: 0.246375, Train loss: 0.5002325568348169 |Validation err: 0.316, Va
Epoch 75: Train err: 0.24575, Train loss: 0.4995848061516881 |Validation err: 0.302, Val
Finished Training
Total time elapsed: 361.62 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`.

```
net = LargeNet()
train_net(net, batch_size= 256, learning_rate= 0.005, num_epochs=75)
large_model_path = get_model_name("large", batch_size= 256, learning_rate= 0.005, epoch=74)
state = torch.load(large_model_path)
net.load_state_dict(state)
```

```
[illegible]
Epoch 23: Train err: 0.33425, Train loss: 0.6149402912706137 |Validation err: 0.3455,
Epoch 24: Train err: 0.337875, Train loss: 0.6155515518039465 |Validation err: 0.3375
Epoch 25: Train err: 0.332625, Train loss: 0.6059867162257433 |Validation err: 0.3295
Epoch 26: Train err: 0.327375, Train loss: 0.6064288131892681 |Validation err: 0.34,
Epoch 27: Train err: 0.322625, Train loss: 0.6010923199355602 |Validation err: 0.3325
Epoch 28: Train err: 0.322125, Train loss: 0.60178660787642 |Validation err: 0.3325,
Epoch 29: Train err: 0.319375, Train loss: 0.598471075296402 |Validation err: 0.325,
Epoch 30: Train err: 0.319375, Train loss: 0.5964281503111124 |Validation err: 0.3365
Epoch 31: Train err: 0.312625, Train loss: 0.5925806201994419 |Validation err: 0.338,
Epoch 32: Train err: 0.313125, Train loss: 0.5878430474549532 |Validation err: 0.333,
Epoch 33: Train err: 0.309125, Train loss: 0.586502570658922 |Validation err: 0.319,
Epoch 34: Train err: 0.307125, Train loss: 0.5855137836188078 |Validation err: 0.3255
Epoch 35: Train err: 0.303375, Train loss: 0.5804027076810598 |Validation err: 0.3175
Epoch 36: Train err: 0.30175, Train loss: 0.575092613697052 |Validation err: 0.3125,
Epoch 37: Train err: 0.304875, Train loss: 0.577560456469655 |Validation err: 0.3255,
Epoch 38: Train err: 0.302625, Train loss: 0.5743005909025669 |Validation err: 0.309,
Epoch 39: Train err: 0.296125, Train loss: 0.5701638367027044 |Validation err: 0.3135
Epoch 40: Train err: 0.2945, Train loss: 0.5656838286668062 |Validation err: 0.3225,
Epoch 41: Train err: 0.296375, Train loss: 0.567337404936552 |Validation err: 0.3055,
Epoch 42: Train err: 0.288875, Train loss: 0.5602938402444124 |Validation err: 0.317,
Epoch 43: Train err: 0.2885, Train loss: 0.5593230184167624 |Validation err: 0.3035,
Epoch 44: Train err: 0.284, Train loss: 0.5575823578983545 |Validation err: 0.316, Va
Epoch 45: Train err: 0.28525, Train loss: 0.5548238847404718 |Validation err: 0.3, Va
Epoch 46: Train err: 0.28725, Train loss: 0.553896252065897 |Validation err: 0.3065,
Epoch 47: Train err: 0.28575, Train loss: 0.5554561326280236 |Validation err: 0.3115,
Epoch 48: Train err: 0.27725, Train loss: 0.5463012456893921 |Validation err: 0.304,
Epoch 49: Train err: 0.283, Train loss: 0.5417572166770697 |Validation err: 0.297, Va
Epoch 50: Train err: 0.2795, Train loss: 0.5386545537039638 |Validation err: 0.303, V
Epoch 51: Train err: 0.27675, Train loss: 0.5411438327282667 |Validation err: 0.3055,
Epoch 52: Train err: 0.275625, Train loss: 0.5423587914556265 |Validation err: 0.294,
Epoch 53: Train err: 0.274875, Train loss: 0.5417576134204865 |Validation err: 0.302,
Epoch 54: Train err: 0.270875, Train loss: 0.5325044523924589 |Validation err: 0.295,
Epoch 55: Train err: 0.266375, Train loss: 0.5294145783409476 |Validation err: 0.298,
Epoch 56: Train err: 0.268, Train loss: 0.5301100201904774 |Validation err: 0.3175, V
Epoch 57: Train err: 0.2695, Train loss: 0.5315655674785376 |Validation err: 0.294, V
Epoch 58: Train err: 0.268, Train loss: 0.5232609640806913 |Validation err: 0.311, Va
Epoch 59: Train err: 0.26475, Train loss: 0.5214256271719933 |Validation err: 0.2945,
Epoch 60: Train err: 0.26525, Train loss: 0.5202046353369951 |Validation err: 0.299,
Epoch 61: Train err: 0.263125, Train loss: 0.5184290977194905 |Validation err: 0.302,
Epoch 62: Train err: 0.261875, Train loss: 0.5161194941028953 |Validation err: 0.313,
Epoch 63: Train err: 0.2625, Train loss: 0.5174261517822742 |Validation err: 0.3015,
Epoch 64: Train err: 0.25525, Train loss: 0.5084416605532169 |Validation err: 0.296,
Epoch 65: Train err: 0.2585, Train loss: 0.507849670946598 |Validation err: 0.294, Va
Epoch 66: Train err: 0.253875, Train loss: 0.5073753613978624 |Validation err: 0.3055
Epoch 67: Train err: 0.253125, Train loss: 0.5041005052626133 |Validation err: 0.309,
Epoch 68: Train err: 0.25475, Train loss: 0.5037137232720852 |Validation err: 0.305,
Epoch 69: Train err: 0.249875, Train loss: 0.5008052075281739 |Validation err: 0.2885
Epoch 70: Train err: 0.24075, Train loss: 0.49319271743297577 |Validation err: 0.3225
Epoch 71: Train err: 0.25175, Train loss: 0.4986856309697032 |Validation err: 0.2985,
Epoch 72: Train err: 0.241, Train loss: 0.4962032027542591 |Validation err: 0.2905, V
Epoch 73: Train err: 0.241875, Train loss: 0.49202906247228384 |Validation err: 0.289
Epoch 74: Train err: 0.24775, Train loss: 0.49573428742587566 |Validation err: 0.3015
Epoch 75: Train err: 0.240875, Train loss: 0.4880987871438265 |Validation err: 0.2915
Finished Training
Total time elapsed: 348.33 seconds
<All keys matched successfully>
```

Part (b) - 2pt

Justify your choice of model from part (a).

These are the reasons why the following hyperparameters performed the best:

1. Large Network
    - Validation and training curves for loss and error fluctuate less than the small net
2. Batch size = 256
    - The loss and error curves for both validation and training decrease steadily as we increase the number of epochs
    - A large batch size doesn't predict validation data well (generalizes the training data) but a low one doesn't give us the best validation accuracy.
3. Epochs = 75
    - There was no overfitting observerd
    - There was converging for the validation data at the loss curve. The slope was about to plateau for a high number of epochs, which indicates neither overfitting or underfitting
4. Learning rate = 0.005
    - There was no overfitting observerd
    - The loss and error curves for both validation and training decrease steadily as we increase the number of epochs

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.

```
# If you use the `evaluate` function provided in part 0, you will need to
# set batch_size > 1
train_loader, val_loader, test_loader, classes = get_data_loader(
    target_classes=["cat", "dog"],
    batch_size=512)

test_err, test_loss = evaluate(net, test_loader, nn.BCEWithLogitsLoss())
print("The error on the test set is: ", test_err)
print("The loss on the test set is: ", test_loss)
```

```
Files already downloaded and verified
Files already downloaded and verified
The error on the test set is:  0.279
The loss on the test set is:  0.5362545400857925
```

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 validation error is approximtely 29% and the test error is approximately 28%. We would usually expect the test error to be higher than the validation error because the model is not being tuned with hyperparameters during the validation set stage. However, test data is completely new data irrespective of hyperparamater tuning and model adjustments. Thus, test data accuracy is not being fine tuned when the model looks at it. This is the reason why we'd usually expect test data error to be greater than validation data error. However, in the case of my model, test error is actually slightly lower than validation error. This is unexpected but it could be because we have a large number of epochs, which makes the model robust. Furthermore, it could be because the images that are in the test dataset are similar to the images in the training dataset, and we obviously know we achieve optimal accuracy with training data.

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?

Test data is completey new to the fold. Training data is what our model uses to learn, validation data is what our model uses to fine tune hyperparamaters in order to become more accurate, and test data is the actual implementation of the model on data it supposedly never seen before. Theoretically, people can add more pictures of dogs and cats forever and add it to the test set. However, people can't add new pictures to the training or validation set without us having to compile the model again and fine tune it. The testing data is used at the end to avoid bias and memorization of test data in the training and validation stage.

Part (f) - 5pt

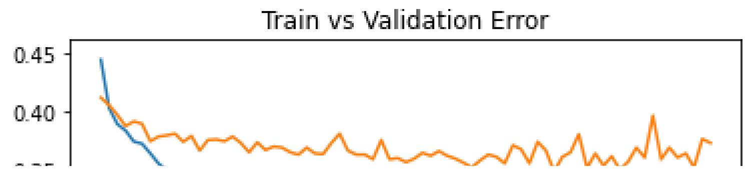
How does the your best CNN model compare with an 2-layer ANN model (no convolutional layers) on classifying cat and dog images. You can use a 2-layer ANN architecture similar to what you used in Lab 1. You should explore different hyperparameter settings to determine how well you can do on the validation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

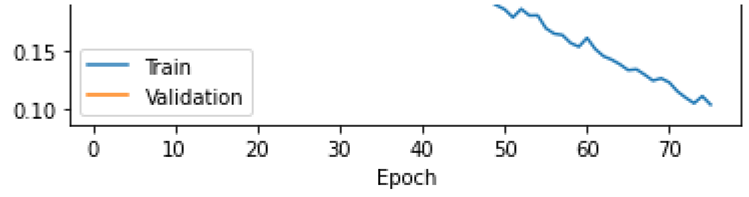
```
class ANN(nn.Module):
    def __init__(self):
        super(ANN, self).__init__()
        self.layer1 = nn.Linear(32 * 32 * 3, 30)
```

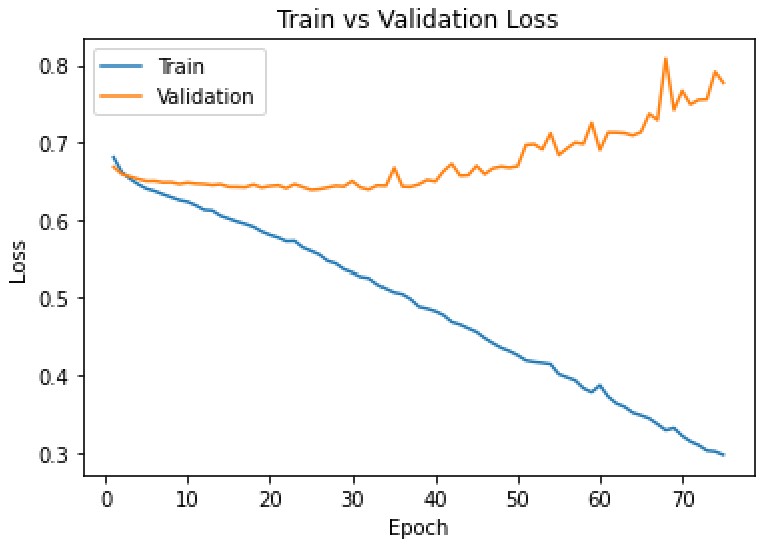
```
            self.layer2 = nn.Linear(30, 1)
            self.name = "ANN"
        def forward(self, img):
            flattened = img.view(-1, 32 * 32 * 3)
            activation1 = self.layer1(flattened)
            activation1 = F.relu(activation1)
            activation2 = self.layer2(activation1)
            return activation2.squeeze()

ANN = ANN()
train_net(ANN, 256, 0.005, 75)
model_path_ANN = get_model_name("ANN", batch_size=256, learning_rate=0.005, epoch=74)
plot_training_curve(model_path_ANN)

train_loader, val_loader, test_loader, classes = get_data_loader(
    target_classes=["cat", "dog"],
    batch_size = 256)

test_err, test_loss = evaluate(ANN, test_loader, nn.BCEWithLogitsLoss())
print("The error on the test set is: ", test_err)
print("The loss on the test set is: ", test_loss)
```

```
Files already downloaded and verified
Files already downloaded and verified
Epoch 1: Train err: 0.444625, Train loss: 0.6807940453290939 |Validation err: 0.412, Val
Epoch 2: Train err: 0.403, Train loss: 0.6615077760070562 |Validation err: 0.4055, Valid
Epoch 3: Train err: 0.38925, Train loss: 0.6529492177069187 |Validation err: 0.397, Vali
Epoch 4: Train err: 0.383625, Train loss: 0.6458388119935989 |Validation err: 0.3875, Va
Epoch 5: Train err: 0.37425, Train loss: 0.6401643473654985 |Validation err: 0.3915, Val
Epoch 6: Train err: 0.372375, Train loss: 0.6371180359274149 |Validation err: 0.3895, Va
Epoch 7: Train err: 0.364125, Train loss: 0.6329818572849035 |Validation err: 0.3745, Va
Epoch 8: Train err: 0.355375, Train loss: 0.6292251236736774 |Validation err: 0.3785, Va
Epoch 9: Train err: 0.34925, Train loss: 0.6254422683268785 |Validation err: 0.3795, Val
Epoch 10: Train err: 0.348875, Train loss: 0.6232133992016315 |Validation err: 0.381, Va
Epoch 11: Train err: 0.343375, Train loss: 0.6186827681958675 |Validation err: 0.374, Va
Epoch 12: Train err: 0.333625, Train loss: 0.6127997878938913 |Validation err: 0.379, Va
Epoch 13: Train err: 0.332875, Train loss: 0.6118991859257221 |Validation err: 0.3665, V
Epoch 14: Train err: 0.32725, Train loss: 0.6051324997097254 |Validation err: 0.3755, Va
Epoch 15: Train err: 0.3245, Train loss: 0.6013806909322739 |Validation err: 0.376, Vali
Epoch 16: Train err: 0.317125, Train loss: 0.5976146515458822 |Validation err: 0.3745, V
Epoch 17: Train err: 0.31525, Train loss: 0.594550209119916 |Validation err: 0.3785, Val
Epoch 18: Train err: 0.31325, Train loss: 0.5908138174563646 |Validation err: 0.373, Val
Epoch 19: Train err: 0.30775, Train loss: 0.5848279725760221 |Validation err: 0.365, Val
Epoch 20: Train err: 0.30775, Train loss: 0.580443924292922 |Validation err: 0.3735, Val
Epoch 21: Train err: 0.297625, Train loss: 0.5771284531801939 |Validation err: 0.367, Va
Epoch 22: Train err: 0.29425, Train loss: 0.5723215434700251 |Validation err: 0.37, Vali
Epoch 23: Train err: 0.296625, Train loss: 0.5727602075785398 |Validation err: 0.369, Va
Epoch 24: Train err: 0.2865, Train loss: 0.564289178699255 |Validation err: 0.365, Valid
Epoch 25: Train err: 0.286625, Train loss: 0.5599226765334606 |Validation err: 0.363, Va
Epoch 26: Train err: 0.28275, Train loss: 0.5552545860409737 |Validation err: 0.369, Val
Epoch 27: Train err: 0.2735, Train loss: 0.5471046101301908 |Validation err: 0.364, Vali
Epoch 28: Train err: 0.274625, Train loss: 0.5438736695796251 |Validation err: 0.3635, V
Epoch 29: Train err: 0.2665, Train loss: 0.5363734047859907 |Validation err: 0.373, Vali
Epoch 30: Train err: 0.26775, Train loss: 0.5323716048151255 |Validation err: 0.381, Val
Epoch 31: Train err: 0.255, Train loss: 0.5270347967743874 |Validation err: 0.3665, Vali
Epoch 32: Train err: 0.259125, Train loss: 0.5252703716978431 |Validation err: 0.363, Va
Epoch 33: Train err: 0.251875, Train loss: 0.5173808615654707 |Validation err: 0.363, Va
Epoch 34: Train err: 0.2455, Train loss: 0.5118708675727248 |Validation err: 0.359, Vali
Epoch 35: Train err: 0.249, Train loss: 0.5070429621264338 |Validation err: 0.3755, Vali
Epoch 36: Train err: 0.242875, Train loss: 0.5047477837651968 |Validation err: 0.359, Va
Epoch 37: Train err: 0.23925, Train loss: 0.4984351107850671 |Validation err: 0.36, Vali
Epoch 38: Train err: 0.226625, Train loss: 0.488625755533576 |Validation err: 0.3565, Va
Epoch 39: Train err: 0.2265, Train loss: 0.4861878603696823 |Validation err: 0.3595, Val
Epoch 40: Train err: 0.225375, Train loss: 0.48286199383437634 |Validation err: 0.3645,
Epoch 41: Train err: 0.220125, Train loss: 0.4778324896469712 |Validation err: 0.362, Va
Epoch 42: Train err: 0.218375, Train loss: 0.468983918428421 |Validation err: 0.366, Val
Epoch 43: Train err: 0.212875, Train loss: 0.46565637551248074 |Validation err: 0.362, V
Epoch 44: Train err: 0.2075, Train loss: 0.46078698709607124 |Validation err: 0.3595, Va
Epoch 45: Train err: 0.206125, Train loss: 0.45613582618534565 |Validation err: 0.356, V
Epoch 46: Train err: 0.20225, Train loss: 0.4480764139443636 |Validation err: 0.352, Val
Epoch 47: Train err: 0.197625, Train loss: 0.44148476887494326 |Validation err: 0.358, V
Epoch 48: Train err: 0.19325, Train loss: 0.43554615415632725 |Validation err: 0.363, Va
Epoch 49: Train err: 0.18925, Train loss: 0.4314077813178301 |Validation err: 0.361, Val
Epoch 50: Train err: 0.185625, Train loss: 0.4260500352829695 |Validation err: 0.3555, V
Epoch 51: Train err: 0.17875, Train loss: 0.41927433386445045 |Validation err: 0.371, Va
Epoch 52: Train err: 0.186, Train loss: 0.41735659074038267 |Validation err: 0.3675, Val
Epoch 53: Train err: 0.180375, Train loss: 0.41609777230769396 |Validation err: 0.3555,
Epoch 54: Train err: 0.1805, Train loss: 0.41456324327737093 |Validation err: 0.374, Val
Epoch 55: Train err: 0.169375, Train loss: 0.4014603700488806 |Validation err: 0.3665,
Epoch 56: Train err: 0.16475, Train loss: 0.3973986776545644 |Validation err: 0.3485, Va
Epoch 57: Train err: 0.163625, Train loss: 0.3934774240478873 |Validation err: 0.361, Va
Epoch 58: Train err: 0.15675, Train loss: 0.3832364147529006 |Validation err: 0.365, Val
Epoch 59: Train err: 0.1535, Train loss: 0.378123483620584 |Validation err: 0.3805, Vali
Epoch 60: Train err: 0.161125, Train loss: 0.3870730772614479 |Validation err: 0.3515,
Epoch 61: Train err: 0.1515, Train loss: 0.37242642790079117 |Validation err: 0.364, Val
Epoch 62: Train err: 0.145375, Train loss: 0.3636471377685666 |Validation err: 0.3535,
Epoch 63: Train err: 0.142375, Train loss: 0.3593013435602188 |Validation err: 0.3615,
Epoch 64: Train err: 0.138375, Train loss: 0.35157999116927385 |Validation err: 0.35, Va
Epoch 65: Train err: 0.133375, Train loss: 0.34811396803706884 |Validation err: 0.357,
Epoch 66: Train err: 0.134125, Train loss: 0.3441974092274904 |Validation err: 0.369, Va
Epoch 67: Train err: 0.1295, Train loss: 0.3369315229356289 |Validation err: 0.3605, Val
Epoch 68: Train err: 0.124375, Train loss: 0.3289829986169934 |Validation err: 0.3965,
Epoch 69: Train err: 0.126375, Train loss: 0.3318782355636358 |Validation err: 0.359, Va
Epoch 70: Train err: 0.12275, Train loss: 0.3216399606317282 |Validation err: 0.369, Val
Epoch 71: Train err: 0.115, Train loss: 0.31440728809684515 |Validation err: 0.3605, Val
Epoch 72: Train err: 0.109625, Train loss: 0.3097377885133028 |Validation err: 0.364, Va
Epoch 73: Train err: 0.104875, Train loss: 0.30267097149044275 |Validation err: 0.352,
Epoch 74: Train err: 0.111, Train loss: 0.30152484495192766 |Validation err: 0.3765, Val
Epoch 75: Train err: 0.10375, Train loss: 0.2970558968372643 |Validation err: 0.373, Val
Finished Training
Total time elapsed: 281.05 seconds
```



The CNN has an error of 27.9% on the test set and a loss of 53.6% on the test set. Comparatively, the ANN has an error of 37.3% on the test set and a loss of 76.2% on the test set. Thus, the CNN gives us better results.





```
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
The error on the test set is:  0.375
```

✓ 5m 22s completed at 2:43 PM