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/1SaHpatXvIGfmo0t9LOHtJRmLuz6vjOX-#scrollTo=SwyDuiuUqIDv

```
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[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)
        print(len(train_loader))
        print(len(val_loader))
        print(len(test_loader))
        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
```

```
Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to ./data/cifar-10-python.tar.gz
100%                                      170498071/170498071 [00:10<00:00, 16980872.97it/s]
Extracting ./data/cifar-10-python.tar.gz to ./data
Files already downloaded and verified
8000
2000
2000
```

Part (a) -- 1 pt

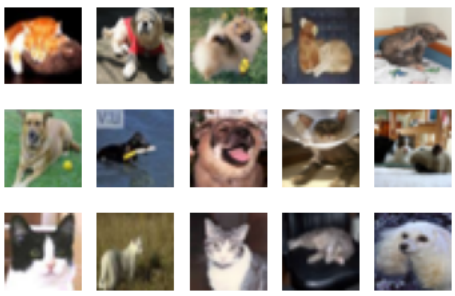
Visualize some of the data by running the code below. Include the visualization in your writeup.

(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?

ANSWER: There are 8000 training examples, 2000 validation examples and 2000 test examples. I got these values by printing their values from the get_data_loader function.

Part (c) -- 3pt

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

ANSWER: Using the performance from our training models to judge the performance of the model results in overfitting the model. Essentially, the model learns the data that we provided and becomes more accurate for the training set only. This is an unrealistic situation because it does not reflect real life events and the model will not perform as well when we provide it with a completely new data set. This is why we use a validations set. The validation set is different from the data we use to train, so it ensures that our model is being exposed to a differnt set of data that from what it was trained on. This is the reason why we use the validation loss/error instead of the training loss/error

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?)

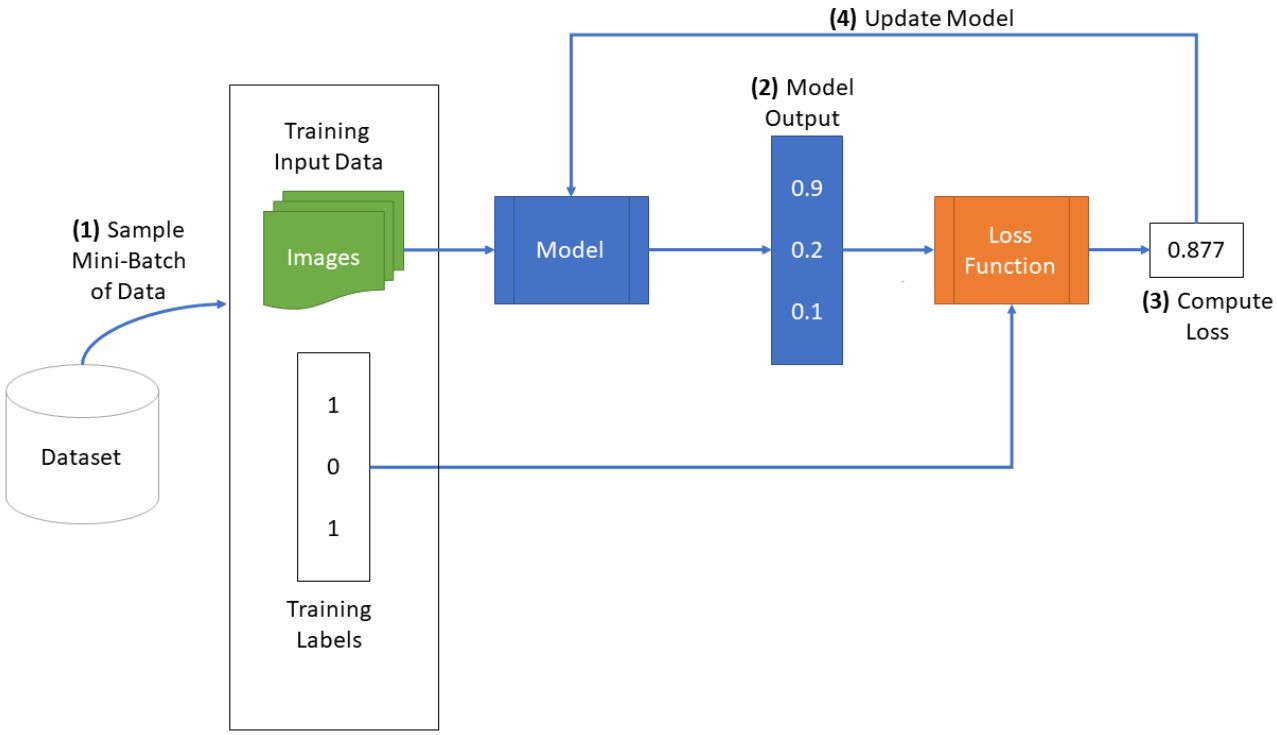
```
for param in small_net.parameters():
    print(param.shape)
print("-----------")
for param in large_net.parameters():
    print(param.shape)
```

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

There are 385 parameters in small_net and 9711 in large_net

The function train_net

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




```
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)

    return [[train_err], [train_loss], [val_err], [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 the parameters are:
batch_size: 64
learning_rate: 0.01
num_epochs: 30

Part (c) -- 3 pt

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

```
train_net(small_net, num_epochs=5)
```

```
Files already downloaded and verified
Files already downloaded and verified
125
32
32
Epoch 1: Train err: 0.42425, Train loss: 0.6727468619346618 |Validation err: 0.374, Validation loss: 0.6540381293743849
Epoch 2: Train err: 0.372, Train loss: 0.646767008304596 |Validation err: 0.3845, Validation loss: 0.6610243525356054
Epoch 3: Train err: 0.35825, Train loss: 0.6394234933853149 |Validation err: 0.3505, Validation loss: 0.6301982179284096
Epoch 4: Train err: 0.351125, Train loss: 0.6309008574485779 |Validation err: 0.3565, Validation loss: 0.6277660019695759
Epoch 5: Train err: 0.3475, Train loss: 0.6269908666610717 |Validation err: 0.347, Validation loss: 0.6286261063069105
Finished Training
Total time elapsed: 22.40 seconds
[[array([0.42425 , 0.372   , 0.35825 , 0.351125, 0.3475  ])],
 [array([0.67274686, 0.64676701, 0.63942349, 0.63090086, 0.62699087])],
 [array([0.374 , 0.3845, 0.3505, 0.3565, 0.347 ])],
 [array([0.65403813, 0.66102435, 0.63019822, 0.627766  , 0.62862611])]]
```

There were 4 files that were written to the disk.

1. A csv file with the training error
2. A csv file with the validation error
3. A csv file with the training loss
4. A csv file with the validation loss

Part (d) -- 2pt

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

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

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

```
# Since the function writes files to disk, you will need to mount
# your Google Drive. If you are working on the lab locally, you
# can comment out this code.

from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

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

```
Files already downloaded and verified
Files already downloaded and verified
125
32
32
Epoch 1: Train err: 0.343875, Train loss: 0.6221144685745239 |Validation err: 0.3495, Validation loss: 0.6283891219645739
Epoch 2: Train err: 0.3445, Train loss: 0.6177322926521301 |Validation err: 0.357, Validation loss: 0.6463163699954748
Epoch 3: Train err: 0.3285, Train loss: 0.6127520666122437 |Validation err: 0.328, Validation loss: 0.609689312055707
Epoch 4: Train err: 0.32325, Train loss: 0.603246077299118 |Validation err: 0.328, Validation loss: 0.6068546213209629
Epoch 5: Train err: 0.316875, Train loss: 0.5968789088726044 |Validation err: 0.326, Validation loss: 0.6049604173749685
Epoch 6: Train err: 0.30475, Train loss: 0.5872797875404357 |Validation err: 0.3195, Validation loss: 0.6004255916923285
Epoch 7: Train err: 0.305375, Train loss: 0.5825518536567688 |Validation err: 0.3285, Validation loss: 0.6007908657193184
Epoch 8: Train err: 0.305375, Train loss: 0.5764223735332489 |Validation err: 0.319, Validation loss: 0.5961459185928106
Epoch 9: Train err: 0.294625, Train loss: 0.5713509321212769 |Validation err: 0.3075, Validation loss: 0.5884125046432018
Epoch 10: Train err: 0.28825, Train loss: 0.564750387430191 |Validation err: 0.3055, Validation loss: 0.5864118542522192
Epoch 11: Train err: 0.290875, Train loss: 0.5626361379623414 |Validation err: 0.3055, Validation loss: 0.5878990478813648
Epoch 12: Train err: 0.28425, Train loss: 0.5562916657924653 |Validation err: 0.3205, Validation loss: 0.5891999276354909
Epoch 13: Train err: 0.285875, Train loss: 0.5568545010089875 |Validation err: 0.3085, Validation loss: 0.5873084599152207
Epoch 14: Train err: 0.2815, Train loss: 0.5515758106708527 |Validation err: 0.3115, Validation loss: 0.5877565033733845
Epoch 15: Train err: 0.282, Train loss: 0.5506065418720245 |Validation err: 0.302, Validation loss: 0.5879388665780425
```

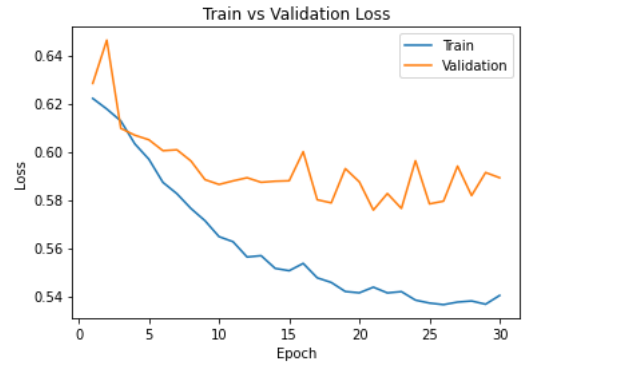
```
Epoch 16: Train err: 0.282, Train loss: 0.5536425430774689 |Validation err: 0.3135, Validation loss: 0.600009385496378
Epoch 17: Train err: 0.278125, Train loss: 0.5476547846794129 |Validation err: 0.305, Validation loss: 0.5800600228831172
Epoch 18: Train err: 0.274375, Train loss: 0.545735451221466 |Validation err: 0.3025, Validation loss: 0.5787737760692835
Epoch 19: Train err: 0.26925, Train loss: 0.5419756600856781 |Validation err: 0.3125, Validation loss: 0.5929877841845155
Epoch 20: Train err: 0.274, Train loss: 0.541430317401886 |Validation err: 0.295, Validation loss: 0.587486170232296
Epoch 21: Train err: 0.275875, Train loss: 0.5437810168266296 |Validation err: 0.303, Validation loss: 0.5757661610841751
Epoch 22: Train err: 0.27375, Train loss: 0.541393982887268 |Validation err: 0.298, Validation loss: 0.582647955045104
Epoch 23: Train err: 0.2715, Train loss: 0.541929416179657 |Validation err: 0.299, Validation loss: 0.5764499651268125
Epoch 24: Train err: 0.267875, Train loss: 0.5384023101329803 |Validation err: 0.302, Validation loss: 0.5962837059050798
Epoch 25: Train err: 0.26825, Train loss: 0.5371822099685669 |Validation err: 0.307, Validation loss: 0.5783445229753852
Epoch 26: Train err: 0.269625, Train loss: 0.5365174922943116 |Validation err: 0.296, Validation loss: 0.5794801386073232
Epoch 27: Train err: 0.271625, Train loss: 0.5375908727645874 |Validation err: 0.305, Validation loss: 0.5940241413190961
Epoch 28: Train err: 0.2715, Train loss: 0.5380516080856323 |Validation err: 0.3015, Validation loss: 0.5817986587062478
Epoch 29: Train err: 0.26875, Train loss: 0.5366939411163331 |Validation err: 0.3025, Validation loss: 0.5913678156211972
Epoch 30: Train err: 0.271875, Train loss: 0.5403403687477112 |Validation err: 0.312, Validation loss: 0.5891934959217906
Finished Training
Total time elapsed: 139.40 seconds
Files already downloaded and verified
Files already downloaded and verified
125
32
32
Epoch 1: Train err: 0.456875, Train loss: 0.6876596741676331 |Validation err: 0.423, Validation loss: 0.6740244068205357
Epoch 2: Train err: 0.4205, Train loss: 0.6732165403366089 |Validation err: 0.4205, Validation loss: 0.6773502249270678
Epoch 3: Train err: 0.393, Train loss: 0.6580913758277893 |Validation err: 0.361, Validation loss: 0.6387437768280506
Epoch 4: Train err: 0.37275, Train loss: 0.6413484840393067 |Validation err: 0.373, Validation loss: 0.6473743487149477
Epoch 5: Train err: 0.354375, Train loss: 0.6297103700637817 |Validation err: 0.341, Validation loss: 0.624759977683425
Epoch 6: Train err: 0.34, Train loss: 0.6173185524940491 |Validation err: 0.3365, Validation loss: 0.6158000845462084
Epoch 7: Train err: 0.328625, Train loss: 0.6095173954963684 |Validation err: 0.3265, Validation loss: 0.6091277338564396
Epoch 8: Train err: 0.3195, Train loss: 0.5935412802696228 |Validation err: 0.327, Validation loss: 0.6041819397360086
Epoch 9: Train err: 0.312625, Train loss: 0.5870700495243073 |Validation err: 0.3175, Validation loss: 0.59456991776824
Epoch 10: Train err: 0.296, Train loss: 0.5752122101783752 |Validation err: 0.3035, Validation loss: 0.5810930477455258
Epoch 11: Train err: 0.292875, Train loss: 0.5616562416553498 |Validation err: 0.3145, Validation loss: 0.594693822786212
Epoch 12: Train err: 0.281125, Train loss: 0.5505828301906586 |Validation err: 0.2975, Validation loss: 0.5762101598083973
Epoch 13: Train err: 0.2745, Train loss: 0.5387271451950073 |Validation err: 0.295, Validation loss: 0.5752458581700921
Epoch 14: Train err: 0.26575, Train loss: 0.5293587062358857 |Validation err: 0.293, Validation loss: 0.5743588227778673
Epoch 15: Train err: 0.263125, Train loss: 0.52365722823143 |Validation err: 0.307, Validation loss: 0.5787387797608972
Epoch 16: Train err: 0.25875, Train loss: 0.5166817557811737 |Validation err: 0.31, Validation loss: 0.6024110801517963
```

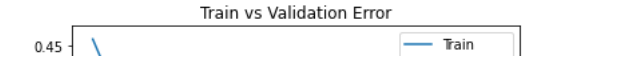
Part (e) - 2pt

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.

```
model_path = get_model_name("small", batch_size=64, learning_rate=0.01, epoch=29)
plot_training_curve(model_path)
print("_____\n")
model_path = get_model_name("large", batch_size=64, learning_rate=0.01, epoch=29)
plot_training_curve(model_path)
```





Part (f) - 5pt

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

In the curves for the small net, we see the validation curves have higher error and loss values than the training curves. It generally appears that as the number of epochs increases, the validation error and loss have a decreasing trend but there are fluctuations in their values at certain intervals. They appear to be slightly overfit. The training curves for the small net have a decreasing trend for error and loss as the number of epochs decreases. They also seem to flunctuate less and seem to be fitted properly.

In the curves for the large net, we see the validation curves have higher error and loss values than the training curves. There validation curves seem to have overfitting. The training curves also decrease as the number of epochs increases but these seem to have less fluctuaion in error and loss values. The training curves also seem to be underfit because we never see the concave minima form.

Part 3. Optimization Parameters [12 pt]

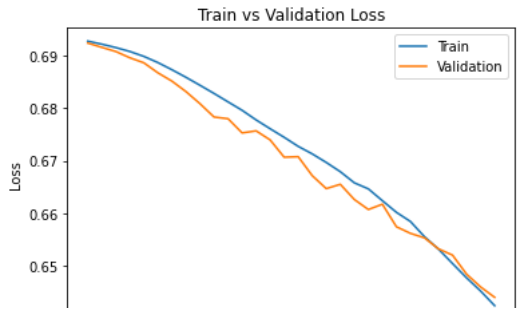
For this section, we will work with `large_net` only.

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) #adjust the learning rate
plot_training_curve(get_model_name('large', 64, 0.001, 29)) #plotting the training curves
```

```
Files already downloaded and verified
Files already downloaded and verified
125
32
32
Epoch 1: Train err: 0.47625, Train loss: 0.6928360013961792 |Validation err: 0.467, Validation loss: 0.6924686580896378
Epoch 2: Train err: 0.448625, Train loss: 0.6922589712142945 |Validation err: 0.4305, Validation loss: 0.691649341955781
Epoch 3: Train err: 0.43575, Train loss: 0.6916067280769348 |Validation err: 0.4285, Validation loss: 0.6908544246610734
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 err: 0.4195, Validation loss: 0.6886935643851757
Epoch 6: Train err: 0.43575, Train loss: 0.6887411961555481 |Validation err: 0.4195, Validation loss: 0.6867824867367744
Epoch 7: Train err: 0.437125, Train loss: 0.6873774147033691 |Validation err: 0.4185, Validation loss: 0.6851982977241278
Epoch 8: Train err: 0.4375, Train loss: 0.6859278454780579 |Validation err: 0.412, Validation loss: 0.683199780061841
Epoch 9: Train err: 0.424375, Train loss: 0.6844058036804199 |Validation err: 0.411, Validation loss: 0.6808880660682917
Epoch 10: Train err: 0.424, Train loss: 0.6828502931594849 |Validation err: 0.408, Validation loss: 0.6783502567559481
Epoch 11: Train err: 0.425375, Train loss: 0.6812348766326904 |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 err: 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 err: 0.4045, Validation loss: 0.6671545393764973
Epoch 18: Train err: 0.3995, Train loss: 0.6696742882728577 |Validation err: 0.4055, Validation loss: 0.6646782550960779
Epoch 19: Train err: 0.40075, Train loss: 0.6679086356163025 |Validation err: 0.396, Validation loss: 0.6655019577592611
Epoch 20: Train err: 0.392375, Train loss: 0.665787980556488 |Validation err: 0.405, Validation loss: 0.6626011095941067
Epoch 21: Train err: 0.38975, Train loss: 0.6646300601959229 |Validation err: 0.394, Validation loss: 0.660687854513526
Epoch 22: Train err: 0.388875, Train loss: 0.662373058795929 |Validation err: 0.393, Validation loss: 0.6616998575627804
Epoch 23: Train err: 0.38425, Train loss: 0.6601516346931458 |Validation err: 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 err: 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 err: 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: 157.52 seconds
```



The result of using a learning rate of 0.001 is a slightly slower model. The previous model had a running time of 155 seconds whereas this model has a run time of 157. The trend we see suggests that if we were to increase the number of epochs that we would see a lower error and loss for both training and validation curves. This represents underfitting though. In both cases, the curves are underfit. This is because the model takes smaller steps when adjusting parameters.

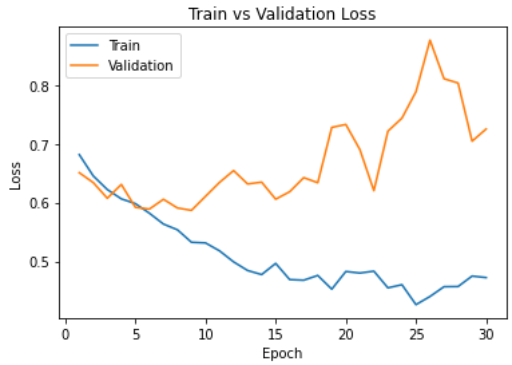
Part (b) - 3pt

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) #adjust the learning rate
```

```
plot_training_curve(get_model_name('large', 64, 0.1, 29)) #plotting the training curves
```

```
Files already downloaded and verified
Files already downloaded and verified
125
32
32
Epoch 1: Train err: 0.449375, Train loss: 0.6816645660400391 |Validation err: 0.389, Validation loss: 0.6508823484182358
Epoch 2: Train err: 0.37425, Train loss: 0.6453690614700317 |Validation err: 0.365, Validation loss: 0.6340354755520821
Epoch 3: Train err: 0.349875, Train loss: 0.6222052867412567 |Validation err: 0.332, Validation loss: 0.6073866505175829
Epoch 4: Train err: 0.3325, Train loss: 0.6063648879528045 |Validation err: 0.351, Validation loss: 0.6309116445481777
Epoch 5: Train err: 0.335, Train loss: 0.5982533438205719 |Validation err: 0.329, Validation loss: 0.5919010397046804
Epoch 6: Train err: 0.314, Train loss: 0.5819428436756134 |Validation err: 0.3115, Validation loss: 0.5889984797686338
Epoch 7: Train err: 0.301875, Train loss: 0.5635049724578858 |Validation err: 0.3345, Validation loss: 0.6056453324854374
Epoch 8: Train err: 0.2945, Train loss: 0.5536779646873474 |Validation err: 0.3195, Validation loss: 0.5908529022708535
Epoch 9: Train err: 0.277375, Train loss: 0.5324801764488221 |Validation err: 0.3, Validation loss: 0.5868263449519873
Epoch 10: Train err: 0.267875, Train loss: 0.5313973410129547 |Validation err: 0.3265, Validation loss: 0.6106775160878897
Epoch 11: Train err: 0.263875, Train loss: 0.5178652312755585 |Validation err: 0.302, Validation loss: 0.6344046164304018
Epoch 12: Train err: 0.24675, Train loss: 0.49915687370300293 |Validation err: 0.3015, Validation loss: 0.6545089883729815
Epoch 13: Train err: 0.244625, Train loss: 0.4844502520561218 |Validation err: 0.298, Validation loss: 0.6317126704379916
Epoch 14: Train err: 0.236625, Train loss: 0.47751300716400147 |Validation err: 0.315, Validation loss: 0.6348515311256051
Epoch 15: Train err: 0.240875, Train loss: 0.4965356433391571 |Validation err: 0.2955, Validation loss: 0.6057963753119111
Epoch 16: Train err: 0.225625, Train loss: 0.4689810929298401 |Validation err: 0.2965, Validation loss: 0.618559280410409
Epoch 17: Train err: 0.226875, Train loss: 0.467936817407608 |Validation err: 0.315, Validation loss: 0.6425053793936968
Epoch 18: Train err: 0.230375, Train loss: 0.47604057478904727 |Validation err: 0.3105, Validation loss: 0.6338433474302292
Epoch 19: Train err: 0.216125, Train loss: 0.4526802043914795 |Validation err: 0.32, Validation loss: 0.7280397713184357
Epoch 20: Train err: 0.233125, Train loss: 0.48278153681755065 |Validation err: 0.2995, Validation loss: 0.7329816520214081
Epoch 21: Train err: 0.2325, Train loss: 0.4801329627037048 |Validation err: 0.306, Validation loss: 0.6903357300907373
Epoch 22: Train err: 0.228875, Train loss: 0.4833245195150375 |Validation err: 0.348, Validation loss: 0.6202784758061171
Epoch 23: Train err: 0.217, Train loss: 0.4549141799211502 |Validation err: 0.3335, Validation loss: 0.7215435951948166
Epoch 24: Train err: 0.211375, Train loss: 0.460289080619812 |Validation err: 0.33, Validation loss: 0.7437789365649223
Epoch 25: Train err: 0.19775, Train loss: 0.4263128218650818 |Validation err: 0.312, Validation loss: 0.7887000488117337
Epoch 26: Train err: 0.201125, Train loss: 0.4402311108112335 |Validation err: 0.335, Validation loss: 0.8765093488618731
Epoch 27: Train err: 0.2065, Train loss: 0.45688191390037536 |Validation err: 0.3375, Validation loss: 0.8108030054718256
Epoch 28: Train err: 0.212, Train loss: 0.457144247174263 |Validation err: 0.3105, Validation loss: 0.8035089243203402
Epoch 29: Train err: 0.2225, Train loss: 0.474978942155838 |Validation err: 0.318, Validation loss: 0.7044840585440397
Epoch 30: Train err: 0.223125, Train loss: 0.47241463100910186 |Validation err: 0.367, Validation loss: 0.7252587173134089
Finished Training
Total time elapsed: 157.54 seconds
```



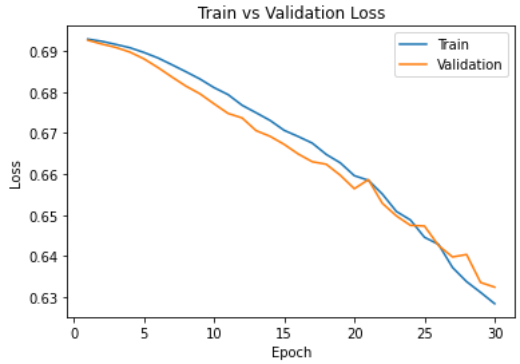
Increasing the learnign rate to 0.1 results in a similar run time as the previous model but the difference is that the model is overfit in this instance. Increasing the learning rate results in an overfit training and validation model error and loss

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, learning_rate=0.01,) #adjust the learning rate
plot_training_curve(get_model_name('large', 512, 0.01, 29)) #plotting the training curves
```

```
Files already downloaded and verified
Files already downloaded and verified
16
4
4
Epoch 1: Train err: 0.48175, Train loss: 0.6929379552602768 |Validation err: 0.478, Validation loss: 0.6926824003458023
Epoch 2: Train err: 0.457625, Train loss: 0.6924104019999504 |Validation err: 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 err: 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 err: 0.426, Validation loss: 0.6836968809366226
Epoch 8: Train err: 0.43525, Train loss: 0.6849770769476891 |Validation err: 0.4115, Validation loss: 0.6814671903848648
Epoch 9: Train err: 0.42375, Train loss: 0.6832009293138981 |Validation err: 0.414, Validation loss: 0.679591491818428
Epoch 10: Train err: 0.421, Train loss: 0.6811089366674423 |Validation err: 0.416, Validation loss: 0.6771548539400101
Epoch 11: Train err: 0.420875, Train loss: 0.6794026419520378 |Validation err: 0.4095, Validation loss: 0.6748111099004745
Epoch 12: Train err: 0.41475, Train loss: 0.6768048219382763 |Validation err: 0.412, Validation loss: 0.6737060546875
Epoch 13: Train err: 0.4105, Train loss: 0.6749702803790569 |Validation err: 0.412, Validation loss: 0.6706101596355438
Epoch 14: Train err: 0.407125, Train loss: 0.6730880849063396 |Validation err: 0.4125, Validation loss: 0.6692148000001907
Epoch 15: Train err: 0.4005, Train loss: 0.6706806942820549 |Validation err: 0.4105, Validation loss: 0.667252704501152
Epoch 16: Train err: 0.397625, Train loss: 0.6691771410405636 |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 err: 0.3945, Validation loss: 0.6624014377593994
Epoch 19: Train err: 0.38625, Train loss: 0.662746611982584 |Validation err: 0.388, Validation loss: 0.6597220152616501
Epoch 20: Train err: 0.38175, Train loss: 0.6596181839704514 |Validation err: 0.4005, Validation loss: 0.6564337313175201
Epoch 21: Train err: 0.38575, Train loss: 0.6584899798035622 |Validation err: 0.3885, Validation loss: 0.6586423963308334
Epoch 22: Train err: 0.378125, Train loss: 0.655123382806778 |Validation err: 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 err: 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 err: 0.3695, Validation loss: 0.6403783112764359
Epoch 29: Train err: 0.3535, Train loss: 0.6311353109776974 |Validation err: 0.366, Validation loss: 0.6335585117340088
Epoch 30: Train err: 0.353, Train loss: 0.6283832415938377 |Validation err: 0.3675, Validation loss: 0.6324127316474915
Finished Training
Total time elapsed: 136.79 seconds
```



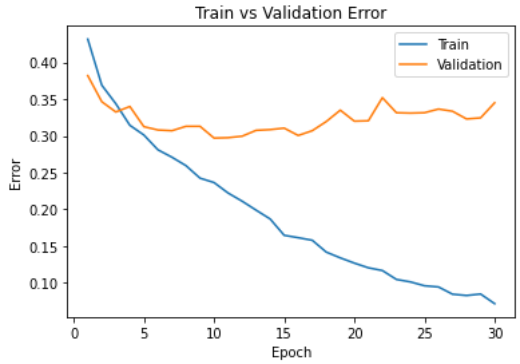
Changing the learning rate to 0.01 and batch size to 512 results in an appropriately fit training curve for the training and validation error but a slightly underfit training curve for the training and validation loss. The model is also faster than the others with a run time of 136 seconds.

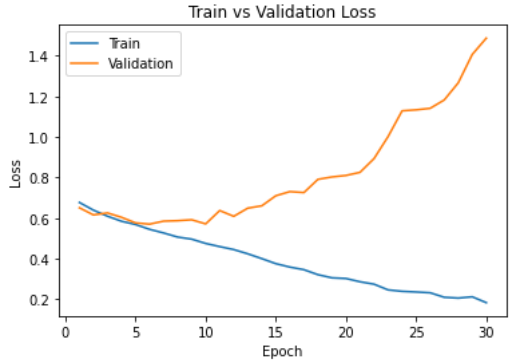
Part (d) - 3pt

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, learning_rate=0.01,) #adjust the learning rate
plot_training_curve(get_model_name('large', 16, 0.01, 29)) #plotting the training curves
```

```
Files already downloaded and verified
Files already downloaded and verified
500
125
125
Epoch 1: Train err: 0.43175, Train loss: 0.6774994022846222 |Validation err: 0.382, Validation loss: 0.6513170118331909
Epoch 2: Train err: 0.369, Train loss: 0.639639899969101 |Validation err: 0.3465, Validation loss: 0.6161113576889038
Epoch 3: Train err: 0.34375, Train loss: 0.6098222947120666 |Validation err: 0.3325, Validation loss: 0.6260210764408112
Epoch 4: Train err: 0.314375, Train loss: 0.5849691489338875 |Validation err: 0.34, Validation loss: 0.6044013917446136
Epoch 5: Train err: 0.301125, Train loss: 0.5689119303822517 |Validation err: 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 err: 0.307, Validation loss: 0.5854293291568756
Epoch 8: Train err: 0.259375, Train loss: 0.5070905526578426 |Validation err: 0.313, Validation loss: 0.5877130818367005
Epoch 9: Train err: 0.242375, Train loss: 0.4968344421982765 |Validation err: 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 err: 0.2995, Validation loss: 0.609202565908432
Epoch 13: Train err: 0.19875, Train loss: 0.4245421719551086 |Validation err: 0.3075, Validation loss: 0.6494987765550614
Epoch 14: Train err: 0.18675, Train loss: 0.4007472907453775 |Validation err: 0.3085, Validation loss: 0.6610016552209854
Epoch 15: Train err: 0.1645, Train loss: 0.3759974058121443 |Validation err: 0.3105, Validation loss: 0.7106090537309646
Epoch 16: Train err: 0.16125, Train loss: 0.3591455406397581 |Validation err: 0.3005, Validation loss: 0.7310364942550659
Epoch 17: Train err: 0.15775, Train loss: 0.3463234790861607 |Validation err: 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 err: 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 err: 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 err: 0.323, Validation loss: 1.266836181640625
Epoch 29: Train err: 0.0845, Train loss: 0.21273409337876364 |Validation err: 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: 207.78 seconds
```





This model results in the validation error and loss training curves being overfit and the training error and loss training curves being underfit.

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.

Based on the plots above, I would pick a batch size of 550 and a learning rate of 0.01. I would apply these to the large net that was used above. I picked these values because based on the plotting results, providing additional samples to the batch should improve the results. The code can be seen below:

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=550, learning_rate=0.01,) #adjust the learning rate
plot_training_curve(get_model_name('large', 550, 0.01, 29)) #plotting the training curves
```

```
Files already downloaded and verified
Files already downloaded and verified
15
4
4
Epoch 1: Train err: 0.49775, Train loss: 0.6958843509356181 |Validation err: 0.509, Validation loss: 0.6966084986925125
Epoch 2: Train err: 0.49775, Train loss: 0.6944016655286153 |Validation err: 0.509, Validation loss: 0.6944023668766022
Epoch 3: Train err: 0.49775, Train loss: 0.6934534947077433 |Validation err: 0.509, Validation loss: 0.693692073225975
Epoch 4: Train err: 0.49775, Train loss: 0.6930482228597005 |Validation err: 0.509, Validation loss: 0.6934096217155457
Epoch 5: Train err: 0.497625, Train loss: 0.6927260796229044 |Validation err: 0.509, Validation loss: 0.6928576678037643
Epoch 6: Train err: 0.4955, Train loss: 0.6925282677014669 |Validation err: 0.496, Validation loss: 0.6925966739654541
Epoch 7: Train err: 0.475875, Train loss: 0.6922845164934794 |Validation err: 0.467, Validation loss: 0.6923087537288666
Epoch 8: Train err: 0.44525, Train loss: 0.6919761459032695 |Validation err: 0.441, Validation loss: 0.6919292956590652
Epoch 9: Train err: 0.451125, Train loss: 0.6916011253992717 |Validation err: 0.458, Validation loss: 0.6914522796869278
Epoch 10: Train err: 0.460125, Train loss: 0.6910890658696492 |Validation err: 0.4565, Validation loss: 0.69090735912323
```

This resulted in a run time of 139 seconds. The results of the training curves were slightly better, increasing the number of epochs is something to be considered.

```
Epoch 15: Train err: 0.440875, Train loss: 0.6861750324567158 |Validation err: 0.4235, Validation loss: 0.686015322804451
```

Part (c) - 2pt

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

```
Epoch 20: Train err: 0.427875, Train loss: 0.6769704205014726 |Validation err: 0.415, Validation loss: 0.6777022997700775
```

For this iteration, I suggest increasing the batch size to 600 and increasing the numebr of epochs to 60.

```
Epoch 24: Train err: 0.401375, Train loss: 0.6707037051518758 |Validation err: 0.407, Validation loss: 0.6687173992395401
```

```
large_net = LargeNet()
train_net(large_net, batch_size=600, learning_rate=0.01, num_epochs=60) #adjust the learning rate
plot_training_curve(get_model_name('large', 600, 0.01, 59)) #plotting the training curves
```

```
Files already downloaded and verified
Files already downloaded and verified
14
4
4
Epoch 1: Train err: 0.485125, Train loss: 0.6929567583969661 |Validation err: 0.484, Validation loss: 0.692743107676506
Epoch 2: Train err: 0.46775, Train loss: 0.6925280860492161 |Validation err: 0.4375, Validation loss: 0.6919053345918655
Epoch 3: Train err: 0.44575, Train loss: 0.6919443351881844 |Validation err: 0.427, Validation loss: 0.6913284361362457
Epoch 4: Train err: 0.430375, Train loss: 0.691216515643256 |Validation err: 0.427, Validation loss: 0.6904537826776505
Epoch 5: Train err: 0.43625, Train loss: 0.6903459472315652 |Validation err: 0.429, Validation loss: 0.6892859935760498
Epoch 6: Train err: 0.439875, Train loss: 0.6894583659512656 |Validation err: 0.431, Validation loss: 0.6884101033210754
Epoch 7: Train err: 0.442875, Train loss: 0.687961995601654 |Validation err: 0.427, Validation loss: 0.6856502145528793
Epoch 8: Train err: 0.444375, Train loss: 0.6867644957133702 |Validation err: 0.427, Validation loss: 0.6845304816961288
Epoch 9: Train err: 0.4345, Train loss: 0.6853091674191611 |Validation err: 0.4125, Validation loss: 0.6809205114841461
Epoch 10: Train err: 0.427125, Train loss: 0.6836487863745008 |Validation err: 0.4125, Validation loss: 0.6798511743545532
Epoch 11: Train err: 0.427, Train loss: 0.6820007732936314 |Validation err: 0.4135, Validation loss: 0.6774551272392273
Epoch 12: Train err: 0.421625, Train loss: 0.6793753504753113 |Validation err: 0.4135, Validation loss: 0.6750219315290451
Epoch 13: Train err: 0.417375, Train loss: 0.678205281496048 |Validation err: 0.4095, Validation loss: 0.673250675201416
Epoch 14: Train err: 0.4125, Train loss: 0.6760282303605761 |Validation err: 0.4145, Validation loss: 0.6725631803274155
Epoch 15: Train err: 0.408625, Train loss: 0.6740108600684575 |Validation err: 0.413, Validation loss: 0.670560285449028
Epoch 16: Train err: 0.405, Train loss: 0.6732917598315648 |Validation err: 0.4075, Validation loss: 0.6672533601522446
Epoch 17: Train err: 0.40225, Train loss: 0.6712638224874224 |Validation err: 0.4095, Validation loss: 0.6681034415960312
Epoch 18: Train err: 0.40075, Train loss: 0.6695902603013175 |Validation err: 0.4045, Validation loss: 0.6659357696771622
Epoch 19: Train err: 0.397125, Train loss: 0.6693094117300851 |Validation err: 0.396, Validation loss: 0.6629659533500671
Epoch 20: Train err: 0.390125, Train loss: 0.6661999140466962 |Validation err: 0.3995, Validation loss: 0.6611586064100266
Epoch 21: Train err: 0.391, Train loss: 0.6645644647734505 |Validation err: 0.3945, Validation loss: 0.662399098277092
Epoch 22: Train err: 0.38625, Train loss: 0.6625989122050149 |Validation err: 0.393, Validation loss: 0.6538701355457306
Epoch 23: Train err: 0.38325, Train loss: 0.6596324103219169 |Validation err: 0.388, Validation loss: 0.6577028036117554
Epoch 24: Train err: 0.38125, Train loss: 0.6570705814020974 |Validation err: 0.3865, Validation loss: 0.6576744318008423
Epoch 25: Train err: 0.380625, Train loss: 0.6545076199940273 |Validation err: 0.3835, Validation loss: 0.6500526070594788
Epoch 26: Train err: 0.3755, Train loss: 0.6513740335191999 |Validation err: 0.389, Validation loss: 0.6524895429611206
Epoch 27: Train err: 0.377375, Train loss: 0.6487023276942117 |Validation err: 0.381, Validation loss: 0.6525014042854309
Epoch 28: Train err: 0.36975, Train loss: 0.6440639666148594 |Validation err: 0.384, Validation loss: 0.6469500809907913
Epoch 29: Train err: 0.364375, Train loss: 0.6403735024588448 |Validation err: 0.3785, Validation loss: 0.6388891786336899
Epoch 30: Train err: 0.36125, Train loss: 0.6376060843467712 |Validation err: 0.3775, Validation loss: 0.6353624314069748
Epoch 31: Train err: 0.359125, Train loss: 0.6378880739212036 |Validation err: 0.374, Validation loss: 0.6378582268953323
Epoch 32: Train err: 0.3545, Train loss: 0.6335748902388981 |Validation err: 0.369, Validation loss: 0.6334873735904694
Epoch 33: Train err: 0.34925, Train loss: 0.6304481838430677 |Validation err: 0.3645, Validation loss: 0.6394166946411133
Epoch 34: Train err: 0.352, Train loss: 0.6296480042593819 |Validation err: 0.3665, Validation loss: 0.6412886530160904
Epoch 35: Train err: 0.348375, Train loss: 0.6266420909336635 |Validation err: 0.3635, Validation loss: 0.627095565199852
Epoch 36: Train err: 0.349375, Train loss: 0.627070495060512 |Validation err: 0.3655, Validation loss: 0.6386073976755142
Epoch 37: Train err: 0.3485, Train loss: 0.6265645367758614 |Validation err: 0.356, Validation loss: 0.6261403411626816
Epoch 38: Train err: 0.340125, Train loss: 0.6172546361173902 |Validation err: 0.347, Validation loss: 0.6337957978248596
Epoch 39: Train err: 0.344875, Train loss: 0.6165627198559898 |Validation err: 0.356, Validation loss: 0.6289205104112625
Epoch 40: Train err: 0.3415, Train loss: 0.6158347768442971 |Validation err: 0.35, Validation loss: 0.6218392252922058
Epoch 41: Train err: 0.340125, Train loss: 0.6120162010192871 |Validation err: 0.351, Validation loss: 0.6321983337402344
Epoch 42: Train err: 0.335625, Train loss: 0.611466109752655 |Validation err: 0.349, Validation loss: 0.6220906674861908
Epoch 43: Train err: 0.337, Train loss: 0.6115209460258484 |Validation err: 0.359, Validation loss: 0.637700766324997
Epoch 44: Train err: 0.33825, Train loss: 0.6123454230172294 |Validation err: 0.342, Validation loss: 0.6128052175045013
```

This output was better than the last, so I will be using the same batch size but change the number of epochs

```
Epoch 47: Train err: 0.328875, Train loss: 0.5972407077059991 |Validation err: 0.331, Validation loss: 0.6100254654884338
```

Part (d) - 1pt

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

```
Epoch 53: Train err: 0.31625, Train loss: 0.5844197358403888 |Validation err: 0.3215, Validation loss: 0.6104704141616821
```

In this model, I will change the number of epochs to 100 and keep the batch size and learning rate at 600 and 0.01 respectively. This is done in hopes of the training curves reaching a slope of 0 so as to stop the underfitting that is recurring.

```
Epoch 58: Train err: 0.300125, Train loss: 0.5700425463063377 |Validation err: 0.32, Validation loss: 0.6014989614486694
```

```
large_net = LargeNet()
train_net(large_net, batch_size=600, learning_rate=0.01, num_epochs=100) #adjust the learning rate
plot_training_curve(get_model_name('large', 600, 0.01, 99)) #plotting the training curves
```

```
Files already downloaded and verified
Files already downloaded and verified
14
4
4
Epoch 1: Train err: 0.51075, Train loss: 0.6942584088870457 |Validation err: 0.505, Validation loss: 0.6936109215021133
Epoch 2: Train err: 0.502875, Train loss: 0.6933198656354632 |Validation err: 0.493, Validation loss: 0.6924757361412048
Epoch 3: Train err: 0.475875, Train loss: 0.6919615268707275 |Validation err: 0.464, Validation loss: 0.6912147402763367
Epoch 4: Train err: 0.451875, Train loss: 0.6908176583903176 |Validation err: 0.439, Validation loss: 0.6897194981575012
Epoch 5: Train err: 0.43975, Train loss: 0.6893220927034106 |Validation err: 0.435, Validation loss: 0.6881787478923798
Epoch 6: Train err: 0.434, Train loss: 0.6879397843565259 |Validation err: 0.4175, Validation loss: 0.6867521554231644
Epoch 7: Train err: 0.43, Train loss: 0.6854697040149144 |Validation err: 0.4105, Validation loss: 0.6834190487861633
Epoch 8: Train err: 0.428, Train loss: 0.6832955224173409 |Validation err: 0.4165, Validation loss: 0.6815724223852158
Epoch 9: Train err: 0.4245, Train loss: 0.6812518622194018 |Validation err: 0.414, Validation loss: 0.6772788316011429
Epoch 10: Train err: 0.419875, Train loss: 0.6784918308258057 |Validation err: 0.411, Validation loss: 0.675326406955719
Epoch 11: Train err: 0.41725, Train loss: 0.6768348600183215 |Validation err: 0.4055, Validation loss: 0.6731758862733841
Epoch 12: Train err: 0.4115, Train loss: 0.6734158779893603 |Validation err: 0.406, Validation loss: 0.66983462870121
Epoch 13: Train err: 0.410625, Train loss: 0.6728492634637016 |Validation err: 0.399, Validation loss: 0.6670345813035965
Epoch 14: Train err: 0.40375, Train loss: 0.6698810756206512 |Validation err: 0.398, Validation loss: 0.6685822159051895
Epoch 15: Train err: 0.402, Train loss: 0.6673736444541386 |Validation err: 0.396, Validation loss: 0.6667110472917557
Epoch 16: Train err: 0.3965, Train loss: 0.666699869292123 |Validation err: 0.393, Validation loss: 0.6605713814496994
Epoch 17: Train err: 0.391625, Train loss: 0.6636356924261365 |Validation err: 0.388, Validation loss: 0.6615630835294724
Epoch 18: Train err: 0.387625, Train loss: 0.660739255802972 |Validation err: 0.386, Validation loss: 0.6584151983261108
Epoch 19: Train err: 0.380375, Train loss: 0.6601731862340655 |Validation err: 0.3835, Validation loss: 0.6515645533800125
Epoch 20: Train err: 0.381125, Train loss: 0.6550111302307674 |Validation err: 0.385, Validation loss: 0.6507856398820877
Epoch 21: Train err: 0.37675, Train loss: 0.6532992678029197 |Validation err: 0.3795, Validation loss: 0.6528979390859604
Epoch 22: Train err: 0.375625, Train loss: 0.6515133678913116 |Validation err: 0.3775, Validation loss: 0.6440322399139404
Epoch 23: Train err: 0.37075, Train loss: 0.6465449716363635 |Validation err: 0.376, Validation loss: 0.6455250829458237
Epoch 24: Train err: 0.372875, Train loss: 0.6446773793016162 |Validation err: 0.3725, Validation loss: 0.6424744576215744
Epoch 25: Train err: 0.369375, Train loss: 0.6420552986008781 |Validation err: 0.373, Validation loss: 0.637271910905838
Epoch 26: Train err: 0.367125, Train loss: 0.6387928426265717 |Validation err: 0.3725, Validation loss: 0.6392369419336319
Epoch 27: Train err: 0.3665, Train loss: 0.6379476870809283 |Validation err: 0.362, Validation loss: 0.6371587961912155
Epoch 28: Train err: 0.355375, Train loss: 0.6323854412351336 |Validation err: 0.358, Validation loss: 0.6382346153259277
Epoch 29: Train err: 0.353125, Train loss: 0.6287510352475303 |Validation err: 0.362, Validation loss: 0.6324562430381775
Epoch 30: Train err: 0.351, Train loss: 0.6285024498190198 |Validation err: 0.357, Validation loss: 0.625980481505394
Epoch 31: Train err: 0.34625, Train loss: 0.6279833231653486 |Validation err: 0.358, Validation loss: 0.629139631986618
Epoch 32: Train err: 0.345875, Train loss: 0.6240127682685852 |Validation err: 0.3515, Validation loss: 0.6262707859277725
Epoch 33: Train err: 0.342875, Train loss: 0.6207927933761052 |Validation err: 0.35, Validation loss: 0.634367898106575
Epoch 34: Train err: 0.342, Train loss: 0.6205492743424007 |Validation err: 0.3465, Validation loss: 0.6314166635274887
Epoch 35: Train err: 0.33725, Train loss: 0.6178786967481885 |Validation err: 0.345, Validation loss: 0.6222217828035355
Epoch 36: Train err: 0.339125, Train loss: 0.61335375053542 |Validation err: 0.3565, Validation loss: 0.6380141228437424
Epoch 37: Train err: 0.3415, Train loss: 0.6129806339740753 |Validation err: 0.341, Validation loss: 0.6153946220874786
Epoch 38: Train err: 0.333125, Train loss: 0.6071258187294006 |Validation err: 0.339, Validation loss: 0.6284419745206833
Epoch 39: Train err: 0.3295, Train loss: 0.6066818067005703 |Validation err: 0.336, Validation loss: 0.6224411278963089
Epoch 40: Train err: 0.331625, Train loss: 0.6062254990850177 |Validation err: 0.344, Validation loss: 0.6190605014562607
Epoch 41: Train err: 0.333625, Train loss: 0.6034703041825976 |Validation err: 0.345, Validation loss: 0.6310382187366486
Epoch 42: Train err: 0.325875, Train loss: 0.6001572353499276 |Validation err: 0.3315, Validation loss: 0.6137246489524841
Epoch 43: Train err: 0.323375, Train loss: 0.6020073550088065 |Validation err: 0.3265, Validation loss: 0.6205061674118042
Epoch 44: Train err: 0.32075, Train loss: 0.5990911849907467 |Validation err: 0.3225, Validation loss: 0.6104606837034225
Epoch 45: Train err: 0.318125, Train loss: 0.5902662149497441 |Validation err: 0.3365, Validation loss: 0.6165037006139755
Epoch 46: Train err: 0.320125, Train loss: 0.5922935264451163 |Validation err: 0.3235, Validation loss: 0.6151655167341232
Epoch 47: Train err: 0.313625, Train loss: 0.5856708117893764 |Validation err: 0.3295, Validation loss: 0.608047142624855
Epoch 48: Train err: 0.31375, Train loss: 0.5857160687446594 |Validation err: 0.322, Validation loss: 0.6076683700084686
Epoch 49: Train err: 0.308375, Train loss: 0.5829244000571114 |Validation err: 0.3145, Validation loss: 0.6109799891710281
Epoch 50: Train err: 0.306875, Train loss: 0.5810238889285496 |Validation err: 0.3225, Validation loss: 0.6137353032827377
Epoch 51: Train err: 0.308, Train loss: 0.5804659596511296 |Validation err: 0.32, Validation loss: 0.6061370670795441
Epoch 52: Train err: 0.3045, Train loss: 0.5738609475748879 |Validation err: 0.3205, Validation loss: 0.6019366532564163
Epoch 53: Train err: 0.30225, Train loss: 0.5728193351200649 |Validation err: 0.3115, Validation loss: 0.604526624083519
Epoch 54: Train err: 0.302625, Train loss: 0.5715649766581399 |Validation err: 0.3155, Validation loss: 0.5962522476911545
Epoch 55: Train err: 0.302875, Train loss: 0.5695030816963741 |Validation err: 0.3155, Validation loss: 0.6038816422224045
Epoch 56: Train err: 0.29875, Train loss: 0.5650093896048409 |Validation err: 0.3145, Validation loss: 0.5947578698396683
Epoch 57: Train err: 0.296375, Train loss: 0.5632142424583435 |Validation err: 0.3255, Validation loss: 0.6167043149471283
Epoch 58: Train err: 0.295875, Train loss: 0.5629352373736245 |Validation err: 0.3105, Validation loss: 0.6017789244651794
Epoch 59: Train err: 0.2895, Train loss: 0.5568210099424634 |Validation err: 0.312, Validation loss: 0.6012299805879593
Epoch 60: Train err: 0.291875, Train loss: 0.5592537309442248 |Validation err: 0.3085, Validation loss: 0.5941370725631714
Epoch 61: Train err: 0.28725, Train loss: 0.5574760011264256 |Validation err: 0.317, Validation loss: 0.5970186740159988
Epoch 62: Train err: 0.286625, Train loss: 0.557020515203476 |Validation err: 0.31, Validation loss: 0.5906314700841904
Epoch 63: Train err: 0.291375, Train loss: 0.5557899389948163 |Validation err: 0.323, Validation loss: 0.6079666465520859
Epoch 64: Train err: 0.282375, Train loss: 0.5480087442057473 |Validation err: 0.312, Validation loss: 0.5955716669559479
Epoch 65: Train err: 0.28275, Train loss: 0.5470626481941768 |Validation err: 0.3015, Validation loss: 0.5920341610908508
Epoch 66: Train err: 0.278375, Train loss: 0.5412215292453766 |Validation err: 0.3155, Validation loss: 0.6168119162321091
Epoch 67: Train err: 0.278625, Train loss: 0.5421070115906852 |Validation err: 0.3185, Validation loss: 0.6007677763700485
Epoch 68: Train err: 0.273625, Train loss: 0.5396436878613063 |Validation err: 0.3095, Validation loss: 0.5934719443321228
Epoch 69: Train err: 0.274375, Train loss: 0.5357805703367505 |Validation err: 0.3115, Validation loss: 0.5927498638629913
Epoch 70: Train err: 0.270625, Train loss: 0.5317417021308627 |Validation err: 0.3325, Validation loss: 0.6074617952108383
Epoch 71: Train err: 0.2765, Train loss: 0.5377016195229122 |Validation err: 0.308, Validation loss: 0.589606985449791
Epoch 72: Train err: 0.271, Train loss: 0.5325990106378283 |Validation err: 0.31, Validation loss: 0.6023779809474945
Epoch 73: Train err: 0.269625, Train loss: 0.5296900400093624 |Validation err: 0.3125, Validation loss: 0.5996412485837936
Epoch 74: Train err: 0.268125, Train loss: 0.5270243457385472 |Validation err: 0.309, Validation loss: 0.5856755673885345
Epoch 75: Train err: 0.262125, Train loss: 0.5201992562838963 |Validation err: 0.3095, Validation loss: 0.5938215404748917
Epoch 76: Train err: 0.263375, Train loss: 0.519443929195404 |Validation err: 0.31, Validation loss: 0.5984406024217606
Epoch 77: Train err: 0.262875, Train loss: 0.5193515313523156 |Validation err: 0.303, Validation loss: 0.5899272710084915
Epoch 78: Train err: 0.269375, Train loss: 0.5294130274227687 |Validation err: 0.302, Validation loss: 0.5786434859037399
Epoch 79: Train err: 0.2625, Train loss: 0.5205053750957761 |Validation err: 0.3085, Validation loss: 0.5844258666038513
Epoch 80: Train err: 0.255625, Train loss: 0.5105787420400766 |Validation err: 0.315, Validation loss: 0.6007615227825083
```

```
Epoch 80: Train err: 0.255625, Train loss: 0.5105787430490766 |Validation err: 0.315, Validation loss: 0.6007615327835083
```

Part 4. Evaluating the Best Model [15 pt]

```
Epoch 85: Train err: 0.246, Train loss: 0.5009178923709052 |Validation err: 0.313, Validation loss: 0.5843250155448914
```

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`.

```
Epoch 94: Train err: 0.228875, Train loss: 0.47807577891009195 |Validation err: 0.308, Validation loss: 0.6008119732141495
```

Double-click (or enter) to edit

```
Epoch 97: Train err: 0.229625, Train loss: 0.4805907756090164 |Validation err: 0.313, Validation loss: 0.5936827957630157
```

```
net = LargeNet()
model_path = get_model_name(net.name, batch_size=600, learning_rate=0.01, epoch=99)
state = torch.load(model_path)
net.load_state_dict(state)
```

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

Part (b) - 2pt

Justify your choice of model from part (a).

I am picking the large net model that I trained last. This is because of its performance and ability to not overfit the validation error and loss. The curves show a trend of their slopes approaching 0 as the number of epochs increases. This implied that the data is not overfit.

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=600)

criterion = nn.BCEWithLogitsLoss()
testError, testLoss = evaluate(net, test_loader, criterion)

print("Test Error: {} \n Test Loss: {}.".format(testError, testLoss))
```

```
Files already downloaded and verified
Files already downloaded and verified
14
4
4
Test Error: 0.302
 Test Loss: 0.5813460350036621.
```

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.

I would expect the test error to be higher than the validation error because the model has never encountered the test set before. The validation data was used to sanity check the model in a way to ensure that we were on the right path and the model performed somwhat accurately. The test data is new data that has never been seen, so the model does not have the opportunity to learn that data set.

Part (e) - 2pt

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

we use the test data set at the very end because it is never introduced in the training of the model, hence it is completely new. The test data also represents the kind of data that will be fed into the model in a real world scenario. It is essential that the model does not learn the test data because then we have no means of verifying accuracy. This is why the test data should not be used frequently.

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 ArtNeuralNet(nn.Module):
    def __init__(self):
        super(ArtNeuralNet, self).__init__()
        self.name = "artNeuralNet"
        self.layer1 = nn.Linear(3 * 32 * 32, 32)
        self.layer2 = nn.Linear(32, 1)
    def forward(self, x):
        x = x.view(-1, 3 * 32 * 32)
        x = self.layer1(x)
        x = F.relu(x)
        x = self.layer2(x)
        x = x.squeeze(1)
        return x

ann_copy = ArtNeuralNet()
train_net(ann_copy, 600, 0.01, 100)
```

```
Files already downloaded and verified
Files already downloaded and verified
14
4
4
Epoch 1: Train err: 0.44475, Train loss: 0.6835844814777374 |Validation err: 0.4065, Validation loss: 0.6704833805561066
Epoch 2: Train err: 0.404875, Train loss: 0.6647350021771022 |Validation err: 0.401, Validation loss: 0.6569951176643372
Epoch 3: Train err: 0.3945, Train loss: 0.6558909288474492 |Validation err: 0.4025, Validation loss: 0.6573218107223511
Epoch 4: Train err: 0.382625, Train loss: 0.6489095134394509 |Validation err: 0.3965, Validation loss: 0.6529643386602402
Epoch 5: Train err: 0.37375, Train loss: 0.643923853124891 |Validation err: 0.398, Validation loss: 0.6496572643518448
Epoch 6: Train err: 0.3725, Train loss: 0.6409934631415776 |Validation err: 0.389, Validation loss: 0.6553976982831955
Epoch 7: Train err: 0.360875, Train loss: 0.6349429275308337 |Validation err: 0.3865, Validation loss: 0.6495446562767029
Epoch 8: Train err: 0.357375, Train loss: 0.6312594243458339 |Validation err: 0.389, Validation loss: 0.6486088633537292
Epoch 9: Train err: 0.35475, Train loss: 0.6252387676920209 |Validation err: 0.388, Validation loss: 0.642610639333725
Epoch 10: Train err: 0.3495, Train loss: 0.62368272457804 |Validation err: 0.3845, Validation loss: 0.6520161926746368
Epoch 11: Train err: 0.343375, Train loss: 0.6191873635564532 |Validation err: 0.3835, Validation loss: 0.645840048789978
Epoch 12: Train err: 0.336125, Train loss: 0.6132804410798209 |Validation err: 0.3845, Validation loss: 0.6470636278390884
Epoch 13: Train err: 0.33425, Train loss: 0.6124552701200757 |Validation err: 0.382, Validation loss: 0.6464066803455353
Epoch 14: Train err: 0.3295, Train loss: 0.6056928762367794 |Validation err: 0.383, Validation loss: 0.6526660770177841
Epoch 15: Train err: 0.328125, Train loss: 0.6029846795967647 |Validation err: 0.3805, Validation loss: 0.644532099366188
Epoch 16: Train err: 0.322875, Train loss: 0.6016088553837368 |Validation err: 0.378, Validation loss: 0.6471859365701675
Epoch 17: Train err: 0.31425, Train loss: 0.5982673934527806 |Validation err: 0.378, Validation loss: 0.6439819633960724
Epoch 18: Train err: 0.313625, Train loss: 0.5920894358839307 |Validation err: 0.374, Validation loss: 0.6443029642105103
Epoch 19: Train err: 0.310625, Train loss: 0.589869601385934 |Validation err: 0.3745, Validation loss: 0.6362830400466919
Epoch 20: Train err: 0.30675, Train loss: 0.583655812910625 |Validation err: 0.3745, Validation loss: 0.6432811170816422
Epoch 21: Train err: 0.305875, Train loss: 0.5818537771701813 |Validation err: 0.381, Validation loss: 0.6477367132902145
Epoch 22: Train err: 0.298375, Train loss: 0.5795289874076843 |Validation err: 0.3755, Validation loss: 0.6485597491264343
Epoch 23: Train err: 0.2965, Train loss: 0.5731717135225024 |Validation err: 0.368, Validation loss: 0.6488521546125412
Epoch 24: Train err: 0.28975, Train loss: 0.5663095968110221 |Validation err: 0.3665, Validation loss: 0.6497164219617844
Epoch 25: Train err: 0.288125, Train loss: 0.5665378059659686 |Validation err: 0.3705, Validation loss: 0.6378602981567383
Epoch 26: Train err: 0.2805, Train loss: 0.5603703856468201 |Validation err: 0.367, Validation loss: 0.64835025370121
Epoch 27: Train err: 0.280625, Train loss: 0.5552010153021131 |Validation err: 0.369, Validation loss: 0.6503701955080032
Epoch 28: Train err: 0.277875, Train loss: 0.5508362310273307 |Validation err: 0.3635, Validation loss: 0.6447155624628067
Epoch 29: Train err: 0.274875, Train loss: 0.5457595203604017 |Validation err: 0.363, Validation loss: 0.6479503214359283
Epoch 30: Train err: 0.275875, Train loss: 0.546186843088695 |Validation err: 0.374, Validation loss: 0.6533912718296051
Epoch 31: Train err: 0.268875, Train loss: 0.5388809570244381 |Validation err: 0.3725, Validation loss: 0.6480784118175507
Epoch 32: Train err: 0.263875, Train loss: 0.5362381849970136 |Validation err: 0.3655, Validation loss: 0.6466501802206039
Epoch 33: Train err: 0.260375, Train loss: 0.5316046008041927 |Validation err: 0.3655, Validation loss: 0.6547514647245407
Epoch 34: Train err: 0.257375, Train loss: 0.5241522022656032 |Validation err: 0.357, Validation loss: 0.6483922600746155
Epoch 35: Train err: 0.247875, Train loss: 0.5224803558417729 |Validation err: 0.366, Validation loss: 0.6598904579877853
Epoch 36: Train err: 0.25975, Train loss: 0.5242769356284823 |Validation err: 0.3675, Validation loss: 0.6588168740272522
Epoch 37: Train err: 0.253125, Train loss: 0.5140121323721749 |Validation err: 0.3605, Validation loss: 0.6431105434894562
Epoch 38: Train err: 0.23575, Train loss: 0.5046580668006625 |Validation err: 0.3625, Validation loss: 0.6629527807235718
Epoch 39: Train err: 0.2365, Train loss: 0.4995929151773453 |Validation err: 0.3705, Validation loss: 0.6731493324041367
Epoch 40: Train err: 0.23025, Train loss: 0.49566556087562014 |Validation err: 0.3665, Validation loss: 0.6571827381849289
Epoch 41: Train err: 0.227125, Train loss: 0.49486234358378817 |Validation err: 0.363, Validation loss: 0.666951447725296
Epoch 42: Train err: 0.22375, Train loss: 0.4861618684870856 |Validation err: 0.367, Validation loss: 0.663611888885498
Epoch 43: Train err: 0.220875, Train loss: 0.48063813149929047 |Validation err: 0.3625, Validation loss: 0.6697879433631897
Epoch 44: Train err: 0.215625, Train loss: 0.47712886546339306 |Validation err: 0.3595, Validation loss: 0.655163362622261
Epoch 45: Train err: 0.213125, Train loss: 0.46960605680942535 |Validation err: 0.3575, Validation loss: 0.6673751771450043
Epoch 46: Train err: 0.212625, Train loss: 0.46618335374764036 |Validation err: 0.365, Validation loss: 0.6630972176790237
Epoch 47: Train err: 0.204125, Train loss: 0.45941127623830524 |Validation err: 0.3635, Validation loss: 0.6654366850852966
Epoch 48: Train err: 0.205375, Train loss: 0.45405976687158855 |Validation err: 0.3565, Validation loss: 0.6716503798961639
Epoch 49: Train err: 0.202, Train loss: 0.4549829810857773 |Validation err: 0.359, Validation loss: 0.6690658628940582
Epoch 50: Train err: 0.195875, Train loss: 0.4450520213161196 |Validation err: 0.362, Validation loss: 0.6717953979969025
Epoch 51: Train err: 0.19225, Train loss: 0.43960377148219515 |Validation err: 0.362, Validation loss: 0.6684551686048508
Epoch 52: Train err: 0.192625, Train loss: 0.4373672774859837 |Validation err: 0.368, Validation loss: 0.6774982959032059
Epoch 53: Train err: 0.183375, Train loss: 0.43072535949093954 |Validation err: 0.3555, Validation loss: 0.6800525039434433
Epoch 54: Train err: 0.181625, Train loss: 0.4250603084053312 |Validation err: 0.367, Validation loss: 0.6838447749614716
Epoch 55: Train err: 0.17775, Train loss: 0.41841711955411093 |Validation err: 0.364, Validation loss: 0.6751594692468643
Epoch 56: Train err: 0.17675, Train loss: 0.41809276172092985 |Validation err: 0.3645, Validation loss: 0.6837043911218643
Epoch 57: Train err: 0.170625, Train loss: 0.40901475719043184 |Validation err: 0.3645, Validation loss: 0.6952266246080399
Epoch 58: Train err: 0.169875, Train loss: 0.40624524652957916 |Validation err: 0.361, Validation loss: 0.6992810219526291
Epoch 59: Train err: 0.16975, Train loss: 0.4032360847507204 |Validation err: 0.3745, Validation loss: 0.7441032081842422
Epoch 60: Train err: 0.175125, Train loss: 0.40667166241577696 |Validation err: 0.363, Validation loss: 0.6975512951612473
Epoch 61: Train err: 0.161375, Train loss: 0.39660242412771496 |Validation err: 0.372, Validation loss: 0.715140700340271
Epoch 62: Train err: 0.158375, Train loss: 0.38700421367372784 |Validation err: 0.364, Validation loss: 0.7028486728668213
Epoch 63: Train err: 0.15575, Train loss: 0.3783236082111086 |Validation err: 0.367, Validation loss: 0.7212184369564056
Epoch 64: Train err: 0.1455, Train loss: 0.36930012702941895 |Validation err: 0.3695, Validation loss: 0.7056530267000198
Epoch 65: Train err: 0.14725, Train loss: 0.3711610223565783 |Validation err: 0.372, Validation loss: 0.731415405869484
Epoch 66: Train err: 0.145, Train loss: 0.3671093157359532 |Validation err: 0.357, Validation loss: 0.7218768745660782
Epoch 67: Train err: 0.14175, Train loss: 0.35821991520268576 |Validation err: 0.364, Validation loss: 0.7201818376779556
Epoch 68: Train err: 0.139375, Train loss: 0.3558743510927473 |Validation err: 0.371, Validation loss: 0.7335847914218903
Epoch 69: Train err: 0.13525, Train loss: 0.35396125061171396 |Validation err: 0.365, Validation loss: 0.7279279679059982
Epoch 70: Train err: 0.1305, Train loss: 0.34102966529982243 |Validation err: 0.362, Validation loss: 0.7355955690145493
Epoch 71: Train err: 0.135375, Train loss: 0.3452525330441339 |Validation err: 0.363, Validation loss: 0.720957413315773
Epoch 72: Train err: 0.13625, Train loss: 0.3432692821536745 |Validation err: 0.37, Validation loss: 0.7440607994794846
Epoch 73: Train err: 0.124125, Train loss: 0.33268677975450245 |Validation err: 0.355, Validation loss: 0.7460689097642899
Epoch 74: Train err: 0.120875, Train loss: 0.32528423837253023 |Validation err: 0.3675, Validation loss: 0.7537982314825058
Epoch 75: Train err: 0.119125, Train loss: 0.32003405903066906 |Validation err: 0.3715, Validation loss: 0.7343990206718445
Epoch 76: Train err: 0.113875, Train loss: 0.3188432774373463 |Validation err: 0.363, Validation loss: 0.7546228021383286
Epoch 77: Train err: 0.119375, Train loss: 0.3180921013866152 |Validation err: 0.3695, Validation loss: 0.7722697556018829
Epoch 78: Train err: 0.1355, Train loss: 0.33312238114220755 |Validation err: 0.3495, Validation loss: 0.7647104263305664
Epoch 79: Train err: 0.128375, Train loss: 0.3224815811429705 |Validation err: 0.3675, Validation loss: 0.7983797341585159
Epoch 80: Train err: 0.113625, Train loss: 0.3004528252592576 |Validation err: 0.3575, Validation loss: 0.762037845402362
```

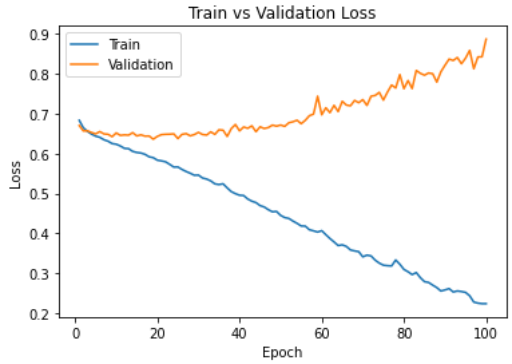
```
Epoch 80: Train err: 0.113625, Train loss: 0.3094528253589376 |Validation err: 0.3575, Validation loss: 0.7630378454923634
Epoch 81: Train err: 0.111625, Train loss: 0.3035372495651245 |Validation err: 0.373, Validation loss: 0.783494308590889
Epoch 82: Train err: 0.1045, Train loss: 0.2964001851422446 |Validation err: 0.3515, Validation loss: 0.763307586312294
Epoch 83: Train err: 0.117125, Train loss: 0.30156005918979645 |Validation err: 0.375, Validation loss: 0.8087167739868164
Epoch 84: Train err: 0.102, Train loss: 0.28869461374623434 |Validation err: 0.3785, Validation loss: 0.8015952855348587
Epoch 85: Train err: 0.09475, Train loss: 0.27871845343283247 |Validation err: 0.367, Validation loss: 0.7968735545873642
Epoch 86: Train err: 0.097, Train loss: 0.27672321668692995 |Validation err: 0.366, Validation loss: 0.8024735003709793
Epoch 87: Train err: 0.092, Train loss: 0.27002129171575817 |Validation err: 0.3625, Validation loss: 0.800553560256958
```

```
plot_training_curve(get_model_name(ann_copy.name, 600, 0.01, 99))

train_loader, val_loader, test_loader, classes = get_data_loader(
    target_classes=["cat", "dog"],
    batch_size=600)

criterion = nn.BCEWithLogitsLoss()
testError, testLoss = evaluate(ann_copy, test_loader, criterion)

print("Test Error: {} \n Test Loss: {}.".format(testError, testLoss))
```




```
Files already downloaded and verified
Files already downloaded and verified
14
4
4
Test Error: 0.372
 Test Loss: 0.8551519513130188.
```

The 2-layer ANN with the same hyperparameters used on the CNN resulted in a 37% test error rate. This is higher than that of the CNN which was 30%. This did not perform too bad but I did not expect it to perform better then the CNN. with finer tuning I suppose it could perform as well. But it is important to note that the test loss is significantly higher than that of the CNN.