FinalProj

April 29, 2021

1 CS 5891: Final Project

1.0.1 Goal: Classify images of dogs into one of 133 breeds using Transfer Learning in Pytorch

[2]:
```
from google.colab import drive
drive.mount('/content/gdrive')
```

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

[3]:
```
%cd gdrive/MyDrive/Final\ Project
```

```
/content/gdrive/MyDrive/Final Project
```

Install Modules

[4]:
```
! pip install torch_utils
```

```
Collecting torch_utils
  Downloading https://files.pythonhosted.org/packages/f8/4d/d004b5af3acf5366b82c
192e459b5a52fba4ced92dccce5ea0541e560900/torch-utils-0.1.2.tar.gz
Requirement already satisfied: torch in /usr/local/lib/python3.7/dist-packages
(from torch_utils) (1.8.1+cu101)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages
(from torch->torch_utils) (1.19.5)
Requirement already satisfied: typing-extensions in
/usr/local/lib/python3.7/dist-packages (from torch->torch_utils) (3.7.4.3)
Building wheels for collected packages: torch-utils
  Building wheel for torch-utils (setup.py) … done
  Created wheel for torch-utils: filename=torch_utils-0.1.2-cp37-none-any.whl
size=6191
sha256=1408bf64c372d704a0be6fd48bb3ada5d6320238aa7d51618b4566f4d7eb9dde
  Stored in directory: /root/.cache/pip/wheels/95/61/06/139d254fa820bc1e45087dba
1d719bc7d4007aec98905179c7
Successfully built torch-utils
Installing collected packages: torch-utils
Successfully installed torch-utils-0.1.2
```

[5]:

```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

# import modules
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import torch
from torch import cuda
import torch.nn as nn
from torch.utils.data import TensorDataset, DataLoader, sampler
import torch.nn.functional as F
from torch_utils import AverageMeter
import math
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error
from sklearn.datasets import load_files
from numpy import inf
import torchvision

from sklearn.model_selection import train_test_split
import os
from glob import glob
from torchvision import transforms
from torchvision import datasets
from torchvision import models
from torch import optim, cuda, Tensor
import tqdm

# Data science tools
import numpy as np

import os

# Image manipulations
from PIL import Image
from timeit import default_timer as timer

# Visualizations
import matplotlib.pyplot as plt
#plt.rcParams['font.size'] = 14

import warnings
warnings.filterwarnings('ignore', category=FutureWarning)
```

[6]:

```
# Define paths and parameters
traindir = f"dogImages/train"
```

```
validdir = f"dogImages/valid"
testdir = f"dogImages/test"

# Change to fit hardware
batch_size = 8
```

1.0.2 Data Augmentation

[7]:

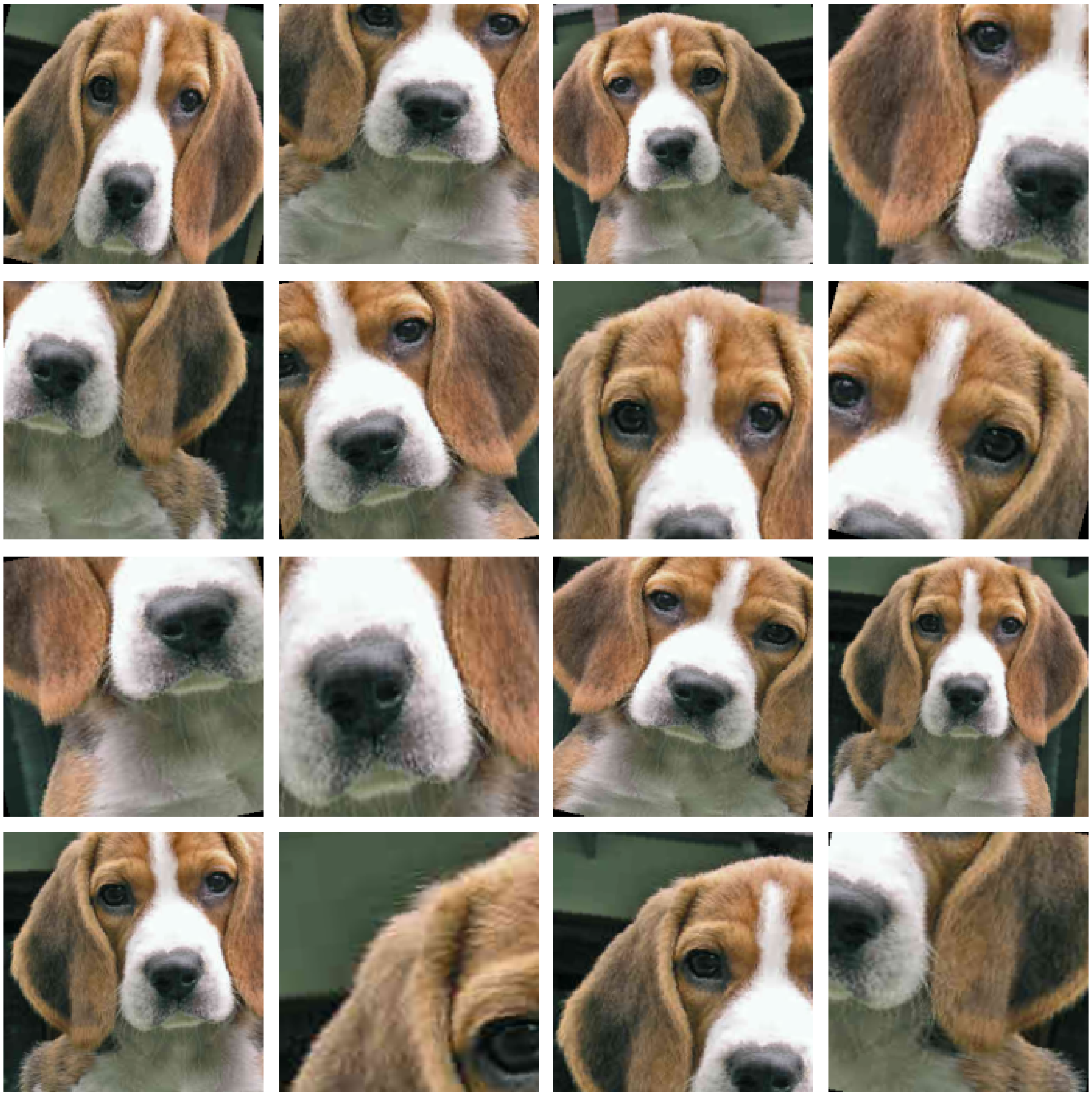
```
# Image transformations
image_transforms = {
    # Train uses data augmentation
    'train':
        transforms.Compose([
            transforms.RandomResizedCrop(256),
            transforms.RandomRotation(degrees=15),
            transforms.ColorJitter(),
            transforms.RandomHorizontalFlip(),
            transforms.CenterCrop(size=224),  # Image net standards
            transforms.ToTensor(),
            transforms.Normalize(mean=[0.485, 0.456, 0.406],
                                 std=[0.229, 0.224, 0.225])  # Imagenet␣
↪standards
        ]),
    # Validation does not use augmentation
    'valid':
        transforms.Compose([
            transforms.Resize(size=256),
            transforms.CenterCrop(size=224),
            transforms.ToTensor(),
            transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
        ]),
    # Test data does not use augmentation
        # Validation does not use augmentation
    'test':
        transforms.Compose([
            transforms.Resize(size=256),
            transforms.CenterCrop(size=224),
            transforms.ToTensor(),
            transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
        ])
}
```

Show Data Augmentation

[8]:

```
def imshow_tensor(image, ax=None, title=None):
    """Imshow for Tensor."""
    if ax is None:
        fig, ax = plt.subplots()

    # Set the color channel as the third dimension
    image = image.numpy().transpose((1, 2, 0))

    # Reverse the preprocessing steps
    mean = np.array([0.485, 0.456, 0.406])
    std = np.array([0.229, 0.224, 0.225])
    image = std * image + mean

    # Clip the image pixel values
    image = np.clip(image, 0, 1)
    ax.imshow(image)
    plt.axis('off')

    return ax, image


ex_img = Image.open('dogImages/train/016.Beagle/Beagle_01140.jpg')

t = image_transforms['train']
plt.figure(figsize=(10, 10))

for i in range(16):
    ax = plt.subplot(4, 4, i + 1)
    _ = imshow_tensor(t(ex_img), ax=ax)

plt.tight_layout()
plt.show()
plt.savefig('augmented_beagle.png')
```

```
[8]: <Figure size 720x720 with 0 Axes>
```



```
<Figure size 432x288 with 0 Axes>
```

1.0.3 Load Data and Extract Images

[9]:

```
# Datasets from folders
data = {
    'train':
    datasets.ImageFolder(root=traindir, transform=image_transforms['train']),
    'valid':
    datasets.ImageFolder(root=traindir, transform=image_transforms['valid']),
    'test':
    datasets.ImageFolder(root=traindir, transform=image_transforms['test'])
}
```

```
# Dataloader iterators, make sure to shuffle
dataloaders = {
    'train': DataLoader(data['train'], batch_size=batch_size,␣
 ↪shuffle=True,num_workers=10),
    'val': DataLoader(data['valid'], batch_size=batch_size,␣
 ↪shuffle=True,num_workers=10),
    'test': DataLoader(data['test'], batch_size=batch_size,␣
 ↪shuffle=True,num_workers=10)
}

# Iterate through the dataloader once
trainiter = iter(dataloaders['train'])
validationiter = iter(dataloaders['val'])
testiter = iter(dataloaders['test'])

categories = []
for d in os.listdir(traindir):
    categories.append(d)

n_classes = len(categories)
print(f'There are {n_classes} different classes.')
```

```
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:477:
UserWarning: This DataLoader will create 10 worker processes in total. Our
suggested max number of worker in current system is 4, which is smaller than
what this DataLoader is going to create. Please be aware that excessive worker
creation might get DataLoader running slow or even freeze, lower the worker
number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
```

```
There are 133 different classes.
```

1.0.4 Dog Detector

In this section, I use a pre-trained model to detect dogs in images. The model has been pre-trained on Imagenet.

[131]:
```
from PIL import ImageFile
ImageFile.LOAD_TRUNCATED_IMAGES = True
cuda.empty_cache()

# Define the network with pretrained model from imagenet
dog_model = models.resnet50(pretrained=True)
```

[11]:
```
# Check whether there is a gpu for cuda
train_on_gpu = cuda.is_available()
print(f'Train on gpu: {train_on_gpu}')
```

```
# Number of gpus
if train_on_gpu:
    gpu_count = cuda.device_count()
    print(f'{gpu_count} gpus detected.')
    if gpu_count > 1:
        multi_gpu = True
    else:
        multi_gpu = False
else:
    multi_gpu = False
print(train_on_gpu,multi_gpu)

if train_on_gpu:
    dog_model = dog_model.to('cuda')
```

```
Train on gpu: True
1 gpus detected.
True False
```

[12]:

```
# Read classes of ImageNet
with open('imagenet_classes.txt') as f:
  classes = [line.strip() for line in f.readlines()]
```

[13]:

```
def ResNet50_predict(data_path):
  '''
  Use pre-trained ResNet50 model to obtain index corresponding to
  predicted ImageNet class for image at specified path
  Args:
      data_path: path to an image
  Returns:
      Index corresponding to VGG-16 model's prediction
  '''
  # Pre-Process Data
  img_t = image_transforms['test'](Image.open(data_path))
  batch_t = torch.unsqueeze(img_t, 0)

  if train_on_gpu:
    batch_t = batch_t.cuda()
  out = dog_model(batch_t)

  # Get index of classification
  _, index = torch.max(out, 1)
  # percentage = torch.nn.functional.softmax(out, dim=1)[0] * 100
  return index[0]

def dog_detector(data_path):
  '''
  Uses ResNet50 predictor to determine if the corresponding
  ImageNet class for image at specified path is a dog
  Args:
    data_path: path to an image
  Returns:
    True if classified as a dog, False for other classifications
  '''
  pred = ResNet50_predict(data_path)
  return (pred <= 268) and (pred >= 151)

# Set model to evaluation mode
dog_model.eval()

# filename = 'dogImages/test/016.Beagle/Beagle_01197.jpg'

path = 'dogImages/test/'
listOfDir = os.listdir(path)
totalFiles = 0
notDogs = 0
for subdir in listOfDir:
  files = os.listdir(path + subdir)
  for filename in files:
    totalFiles += 1
    if not dog_detector(path + subdir + '/' + filename):
      notDogs += 1
      print("%s was not correctly identified as a dog." % (filename))

print("ImageNet classified %d images. %d were correctly classified as dogs." %␣
 ↪(totalFiles, totalFiles - notDogs))
print("ResNet50 Top-1 Error %d%% for images of dogs" % (((totalFiles - notDogs)␣
 ↪/ totalFiles) * 100))
```

```
[13]: ResNet(
        (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3),
      bias=False)
        (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
        (relu): ReLU(inplace=True)
        (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1,
      ceil_mode=False)
        (layer1): Sequential(
          (0): Bottleneck(
            (conv1): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
            (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
      bias=False)
            (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
            (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
            (relu): ReLU(inplace=True)
            (downsample): Sequential(
              (0): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
              (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
            )
          )
          (1): Bottleneck(
            (conv1): Conv2d(256, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
            (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
      bias=False)
            (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
            (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
            (relu): ReLU(inplace=True)
          )
          (2): Bottleneck(
            (conv1): Conv2d(256, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
            (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
      bias=False)
            (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
            (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
            (relu): ReLU(inplace=True)
          )
        )
        (layer2): Sequential(
          (0): Bottleneck(
            (conv1): Conv2d(256, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
            (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1,
      1), bias=False)
```

```
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (downsample): Sequential(
        (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): Bottleneck(
      (conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
    (2): Bottleneck(
      (conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
    (3): Bottleneck(
      (conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
  )
  (layer3): Sequential(
    (0): Bottleneck(
      (conv1): Conv2d(512, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (downsample): Sequential(
        (0): Conv2d(512, 1024, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): Bottleneck(
      (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
    (2): Bottleneck(
      (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
    (3): Bottleneck(
      (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
    (4): Bottleneck(
      (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
    (5): Bottleneck(
      (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
  )
  (layer4): Sequential(
    (0): Bottleneck(
      (conv1): Conv2d(1024, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (downsample): Sequential(
        (0): Conv2d(1024, 2048, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): Bottleneck(
      (conv1): Conv2d(2048, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
    (2): Bottleneck(
      (conv1): Conv2d(2048, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
  (fc): Linear(in_features=2048, out_features=1000, bias=True)
)
```

```
Parson_russell_terrier_07529.jpg was not correctly identified as a dog.
```

```
Norwegian_buhund_07120.jpg was not correctly identified as a dog.
Norwegian_buhund_07111.jpg was not correctly identified as a dog.
Akita_00276.jpg was not correctly identified as a dog.
Akita_00262.jpg was not correctly identified as a dog.
Norwegian_lundehund_07222.jpg was not correctly identified as a dog.
Australian_cattle_dog_00728.jpg was not correctly identified as a dog.
Australian_cattle_dog_00761.jpg was not correctly identified as a dog.
Australian_cattle_dog_00792.jpg was not correctly identified as a dog.
Canaan_dog_03066.jpg was not correctly identified as a dog.
Canaan_dog_03073.jpg was not correctly identified as a dog.
Canaan_dog_03084.jpg was not correctly identified as a dog.
ImageNet classified 835 images. 823 were correctly classified as dogs.
ResNet50 Top-1 Error 98% for images of dogs
```

1.1 Baseline Model (HW 4)

CNN consists of three convolutional layers, where each convolutional layer is followed by a max pooling layer and one ReLU layer. This used a kernel size of 3 and a stride of 1. The max pooling layer used a kernel size of 2 and a stride of 1.

Three fully connected layers followed the convolutional layers, which resulted in 133 output neurons that allowed us to classify the images into 133 categories (breeds) using one-hot encoding.

[14]:

```python
# Define model
class dmodel(nn.Module):
    def __init__(self):
        super(dmodel, self).__init__()

        # 2D Convolutional Neural Network
        self.conv1 = nn.Conv2d(3, 6, kernel_size=5, stride=1)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, kernel_size=5, stride=1)
        self.fc1 = nn.Linear(16 * 53 * 53, 4096)
        self.fc2 = nn.Linear(4096, 1024)
        self.fc3 = nn.Linear(1024, 133)

    def forward(self, x):
        # Convolutional Layers
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = x.view(-1, 16 * 53 * 53)

        # Fully Connected Layers
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

```
scratch_model = dmodel()
```

[19]:

```
# Set model to GPU
if train_on_gpu:
    scratch_model = scratch_model.to('cuda')

# Set up your criterion and optimizer
learning_rate = 1e-4
scratch_optimizer = optim.SGD(scratch_model.parameters(), lr = learning_rate)
scratch_criterion = nn.CrossEntropyLoss()
```

[20]:

```
# For results
print(scratch_model)
```

```
dmodel(
  (conv1): Conv2d(3, 6, kernel_size=(5, 5), stride=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  (conv2): Conv2d(6, 16, kernel_size=(5, 5), stride=(1, 1))
  (fc1): Linear(in_features=44944, out_features=4096, bias=True)
  (fc2): Linear(in_features=4096, out_features=1024, bias=True)
  (fc3): Linear(in_features=1024, out_features=133, bias=True)
)
```

Training the CNN Model

[21]:

```
def train(model,
          criterion,
          optimizer,
          train_loader,
          valid_loader,
          save_file_name,
          max_epochs_stop=3,
          n_epochs=10,
          print_every=1):
    """Train a PyTorch Model

    Params
    --------
        model (PyTorch model): cnn to train
        criterion (PyTorch loss): objective to minimize
        optimizer (PyTorch optimizier): optimizer to compute gradients of model␣
 ↪parameters
        train_loader (PyTorch dataloader): training dataloader to iterate␣
 ↪through
        valid_loader (PyTorch dataloader): validation dataloader used for early␣
 ↪stopping
```

```
        save_file_name (str ending in '.pt'): file path to save the model state dict
        max_epochs_stop (int): maximum number of epochs with no improvement in validation loss for early stopping
        n_epochs (int): maximum number of training epochs
        print_every (int): frequency of epochs to print training stats

    Returns
    --------
        model (PyTorch model): trained cnn with best weights
        history (DataFrame): history of train and validation loss and accuracy
    """

    # Early stopping intialization
    epochs_no_improve = 0
    valid_loss_min = np.Inf

    valid_max_acc = 0
    history = []

    # Number of epochs already trained (if using loaded in model weights)
    try:
        print(f'Model has been trained for: {model.epochs} epochs.\n')
    except:
        model.epochs = 0
        print(f'Starting Training from Scratch.\n')

    overall_start = timer()

    # Main loop
    for epoch in range(n_epochs):

        # keep track of training and validation loss each epoch
        train_loss = 0.0
        valid_loss = 0.0

        train_acc = 0
        valid_acc = 0


        # Set to training
        model.train()

        start = timer()

        # Training loop
        for ii, (data, target) in enumerate(train_loader):
            # Tensors to gpu
            if train_on_gpu:
                model = model.cuda()
                data, target = data.cuda(), target.cuda()


            # Clear gradients
            optimizer.zero_grad()

            # Get your output from your model
            model = model.float()
            output = model(data.float())

            # Loss and backpropagation of gradients
            loss = criterion(output, target.long())
            loss.backward()

            # Update the parameters
            optimizer.step()

            # Track train loss by multiplying average loss by number of examples in batch
            train_loss += loss.item() * data.size(0)

            # Calculate accuracy by finding max log probability
            _, pred = torch.max(output, dim=1)
            correct_tensor = pred.eq(target.data.view_as(pred))

            # Need to convert correct tensor from int to float to average
            accuracy = torch.mean(correct_tensor.type(torch.FloatTensor))

            # Multiply average accuracy times the number of examples in batch
            train_acc += accuracy.item() * data.size(0)

            # Track training progress
            print(
                f'Epoch: {epoch}\t{100 * (ii + 1) / len(train_loader):.2f}% complete. {timer() - start:.2f} seconds elapsed in epoch.',
                end='\r')

        # After training loops ends, start validation
        else:
            model.epochs += 1

            # Don't need to keep track of gradients
            with torch.no_grad():
                # Set to evaluation mode
                model.eval()

                # Validation loop
                for data, target in valid_loader:

                    # Tensors to gpu
                    if train_on_gpu:
                        model = model.cuda()
                        data, target = data.cuda(), target.cuda()

                    # Forward pass
                    model = model.float()
                    output = model(data.float())

                    # Validation loss
                    loss = criterion(output, target.long())

                    # Multiply average loss times the number of examples in␣
↪batch

                    valid_loss += loss.item() * data.size(0)

                    # Calculate validation accuracy
                    _, pred = torch.max(output, dim=1)
                    correct_tensor = pred.eq(target.data.view_as(pred))
                    accuracy = torch.mean(
                        correct_tensor.type(torch.FloatTensor))

                    # Multiply average accuracy times the number of examples
                    valid_acc += accuracy.item() * data.size(0)


                # Calculate average losses
                train_loss = train_loss / len(train_loader.dataset)
                valid_loss = valid_loss / len(valid_loader.dataset)


                # Calculate average accuracy
                train_acc = train_acc / len(train_loader.dataset)
                valid_acc = valid_acc / len(valid_loader.dataset)
                history.append([train_loss, valid_loss, train_acc, valid_acc])

                # Print training and validation results
                if (epoch + 1) % print_every == 0:
                    print(
                        f'\nEpoch: {epoch} \tTraining Loss: {train_loss:.4f}␣
↪\tValidation Loss: {valid_loss:.4f}'
                    )
                    print(
                        f'\t\tTraining Accuracy: {100 * train_acc:.2f}%\t␣
↪Validation Accuracy: {100 * valid_acc:.2f}%'
                    )

                # Save the model if validation loss decreases
                if valid_loss < valid_loss_min:
                    # Save model
                    torch.save(model.state_dict(), save_file_name)
                    # Track improvement
                    epochs_no_improve = 0
                    valid_loss_min = valid_loss
                    valid_best_acc = valid_acc
                    best_epoch = epoch

                # Otherwise increment count of epochs with no improvement
                else:
                    epochs_no_improve += 1
                    # Trigger early stopping
                    if epochs_no_improve >= max_epochs_stop:
                        print(
                            f'\nEarly Stopping! Total epochs: {epoch}. Best␣
↪epoch: {best_epoch} with loss: {valid_loss_min:.2f} and acc: {100 *␣
↪valid_acc:.2f}%'
                        )
                        total_time = timer() - overall_start
                        print(
                            f'{total_time:.2f} total seconds elapsed.␣
↪{total_time / (epoch+1):.2f} seconds per epoch.'
                        )

                        # Load the best state dict
                        model.load_state_dict(torch.load(save_file_name))

                        # Attach the optimizer
                        model.optimizer = optimizer

                        # Format history
                        history = pd.DataFrame(
                            history,
                            columns=[
                                'train_loss', 'valid_loss', 'train_acc',
                                'valid_acc'
                            ])
                        return model, history

    # Attach the optimizer
    model.optimizer = optimizer
    # Record overall time and print out stats
    total_time = timer() - overall_start
    print(
        f'\nBest epoch: {best_epoch} with loss: {valid_loss_min:.2f} and acc:␣
↪{100 * valid_best_acc:.2f}%'
    )
    print(
        f'{total_time:.2f} total seconds elapsed. {total_time / (epoch+1):.2f}␣
↪seconds per epoch.'
    )
    # Format history
    history = pd.DataFrame(
        history,
        columns=['train_loss', 'valid_loss', 'train_acc', 'valid_acc'])
    return model, history
```

[16]:

```
from timeit import default_timer as timer
save_file_name = f'CNN_scratch_model.pt'
train_on_gpu = cuda.is_available()

scratch_model, scratch_history = train(scratch_model,
    scratch_criterion,
    scratch_optimizer,
    dataloaders['train'],
    dataloaders['val'],
    save_file_name=save_file_name,
    max_epochs_stop=5,
    n_epochs=500,
    print_every=1)
```

```
Starting Training from Scratch.


/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:477:
UserWarning: This DataLoader will create 10 worker processes in total. Our
suggested max number of worker in current system is 4, which is smaller than
what this DataLoader is going to create. Please be aware that excessive worker
creation might get DataLoader running slow or even freeze, lower the worker
number to avoid potential slowness/freeze if necessary.
  cpuset_checked))


Epoch: 0        Training Loss: 4.8906   Validation Loss: 4.8900
                Training Accuracy: 0.81%         Validation Accuracy: 0.73%
```

```
Epoch: 1        Training Loss: 4.8897   Validation Loss: 4.8889
                Training Accuracy: 0.82%         Validation Accuracy: 0.72%

Epoch: 2        Training Loss: 4.8883   Validation Loss: 4.8877
                Training Accuracy: 0.97%         Validation Accuracy: 0.81%

Epoch: 3        Training Loss: 4.8874   Validation Loss: 4.8865
                Training Accuracy: 0.99%         Validation Accuracy: 0.88%

Epoch: 4        Training Loss: 4.8861   Validation Loss: 4.8851
                Training Accuracy: 1.09%         Validation Accuracy: 0.88%

Epoch: 5        Training Loss: 4.8849   Validation Loss: 4.8837
                Training Accuracy: 0.99%         Validation Accuracy: 0.94%

Epoch: 6        Training Loss: 4.8832   Validation Loss: 4.8821
                Training Accuracy: 0.99%         Validation Accuracy: 1.09%

Epoch: 7        Training Loss: 4.8821   Validation Loss: 4.8803
                Training Accuracy: 1.05%         Validation Accuracy: 1.23%

Epoch: 8        Training Loss: 4.8802   Validation Loss: 4.8782
                Training Accuracy: 1.21%         Validation Accuracy: 1.30%

Epoch: 9        Training Loss: 4.8783   Validation Loss: 4.8758
                Training Accuracy: 1.50%         Validation Accuracy: 1.47%

Epoch: 10       Training Loss: 4.8760   Validation Loss: 4.8729
                Training Accuracy: 1.47%         Validation Accuracy: 1.77%

Epoch: 11       Training Loss: 4.8723   Validation Loss: 4.8693
                Training Accuracy: 1.65%         Validation Accuracy: 1.83%

Epoch: 12       Training Loss: 4.8690   Validation Loss: 4.8649
                Training Accuracy: 1.57%         Validation Accuracy: 1.99%

Epoch: 13       Training Loss: 4.8644   Validation Loss: 4.8597
                Training Accuracy: 1.81%         Validation Accuracy: 1.93%

Epoch: 14       Training Loss: 4.8597   Validation Loss: 4.8537
                Training Accuracy: 1.98%         Validation Accuracy: 1.95%

Epoch: 15       Training Loss: 4.8537   Validation Loss: 4.8469
                Training Accuracy: 1.95%         Validation Accuracy: 2.08%

Epoch: 16       Training Loss: 4.8479   Validation Loss: 4.8395
                Training Accuracy: 2.13%         Validation Accuracy: 2.13%
```

```
Epoch: 17       Training Loss: 4.8424   Validation Loss: 4.8317
                Training Accuracy: 2.07%         Validation Accuracy: 2.08%

Epoch: 18       Training Loss: 4.8344   Validation Loss: 4.8232
                Training Accuracy: 2.17%         Validation Accuracy: 2.13%

Epoch: 19       Training Loss: 4.8287   Validation Loss: 4.8143
                Training Accuracy: 2.10%         Validation Accuracy: 2.35%

Epoch: 20       Training Loss: 4.8255   Validation Loss: 4.8050
                Training Accuracy: 2.16%         Validation Accuracy: 2.81%

Epoch: 21       Training Loss: 4.8174   Validation Loss: 4.7945
                Training Accuracy: 1.95%         Validation Accuracy: 2.92%

Epoch: 22       Training Loss: 4.8102   Validation Loss: 4.7828
                Training Accuracy: 2.38%         Validation Accuracy: 3.11%

Epoch: 23       Training Loss: 4.8040   Validation Loss: 4.7704
                Training Accuracy: 2.47%         Validation Accuracy: 3.35%

Epoch: 24       Training Loss: 4.7914   Validation Loss: 4.7552
                Training Accuracy: 2.49%         Validation Accuracy: 3.47%

Epoch: 25       Training Loss: 4.7833   Validation Loss: 4.7393
                Training Accuracy: 2.66%         Validation Accuracy: 3.50%

Epoch: 26       Training Loss: 4.7731   Validation Loss: 4.7223
                Training Accuracy: 2.77%         Validation Accuracy: 3.64%

Epoch: 27       Training Loss: 4.7625   Validation Loss: 4.7040
                Training Accuracy: 2.72%         Validation Accuracy: 3.61%

Epoch: 28       Training Loss: 4.7450   Validation Loss: 4.6840
                Training Accuracy: 2.98%         Validation Accuracy: 3.55%

Epoch: 29       Training Loss: 4.7392   Validation Loss: 4.6661
                Training Accuracy: 2.63%         Validation Accuracy: 3.86%

Epoch: 30       Training Loss: 4.7311   Validation Loss: 4.6492
                Training Accuracy: 2.80%         Validation Accuracy: 4.07%

Epoch: 31       Training Loss: 4.7310   Validation Loss: 4.6351
                Training Accuracy: 2.81%         Validation Accuracy: 4.09%

Epoch: 32       Training Loss: 4.7152   Validation Loss: 4.6187
                Training Accuracy: 2.81%         Validation Accuracy: 4.16%
```

```
Epoch: 33       Training Loss: 4.7167   Validation Loss: 4.6070
                Training Accuracy: 2.50%         Validation Accuracy: 4.24%

Epoch: 34       Training Loss: 4.7127   Validation Loss: 4.5961
                Training Accuracy: 2.69%         Validation Accuracy: 4.22%

Epoch: 35       Training Loss: 4.7051   Validation Loss: 4.5847
                Training Accuracy: 2.84%         Validation Accuracy: 4.19%

Epoch: 36       Training Loss: 4.7073   Validation Loss: 4.5760
                Training Accuracy: 2.68%         Validation Accuracy: 4.24%

Epoch: 37       Training Loss: 4.6939   Validation Loss: 4.5658
                Training Accuracy: 2.84%         Validation Accuracy: 4.42%

Epoch: 38       Training Loss: 4.6957   Validation Loss: 4.5571
                Training Accuracy: 2.65%         Validation Accuracy: 4.39%

Epoch: 39       Training Loss: 4.6871   Validation Loss: 4.5474
                Training Accuracy: 3.20%         Validation Accuracy: 4.79%

Epoch: 40       Training Loss: 4.6785   Validation Loss: 4.5394
                Training Accuracy: 3.01%         Validation Accuracy: 4.54%

Epoch: 41       Training Loss: 4.6701   Validation Loss: 4.5304
                Training Accuracy: 3.01%         Validation Accuracy: 4.88%

Epoch: 42       Training Loss: 4.6758   Validation Loss: 4.5224
                Training Accuracy: 3.20%         Validation Accuracy: 4.96%

Epoch: 43       Training Loss: 4.6798   Validation Loss: 4.5186
                Training Accuracy: 3.10%         Validation Accuracy: 4.73%

Epoch: 44       Training Loss: 4.6688   Validation Loss: 4.5094
                Training Accuracy: 3.25%         Validation Accuracy: 5.22%

Epoch: 45       Training Loss: 4.6598   Validation Loss: 4.5001
                Training Accuracy: 3.26%         Validation Accuracy: 4.84%

Epoch: 46       Training Loss: 4.6521   Validation Loss: 4.4931
                Training Accuracy: 3.34%         Validation Accuracy: 5.22%

Epoch: 47       Training Loss: 4.6423   Validation Loss: 4.4860
                Training Accuracy: 3.65%         Validation Accuracy: 5.10%

Epoch: 48       Training Loss: 4.6521   Validation Loss: 4.4804
                Training Accuracy: 3.34%         Validation Accuracy: 5.45%
```

```
Epoch: 49       Training Loss: 4.6573   Validation Loss: 4.4754
                Training Accuracy: 3.44%         Validation Accuracy: 5.39%

Epoch: 50       Training Loss: 4.6546   Validation Loss: 4.4720
                Training Accuracy: 3.38%         Validation Accuracy: 5.40%

Epoch: 51       Training Loss: 4.6480   Validation Loss: 4.4638
                Training Accuracy: 3.49%         Validation Accuracy: 5.72%

Epoch: 52       Training Loss: 4.6426   Validation Loss: 4.4567
                Training Accuracy: 3.40%         Validation Accuracy: 5.64%

Epoch: 53       Training Loss: 4.6390   Validation Loss: 4.4475
                Training Accuracy: 3.29%         Validation Accuracy: 6.09%

Epoch: 54       Training Loss: 4.6269   Validation Loss: 4.4416
                Training Accuracy: 3.79%         Validation Accuracy: 6.08%

Epoch: 55       Training Loss: 4.6180   Validation Loss: 4.4312
                Training Accuracy: 3.80%         Validation Accuracy: 5.76%

Epoch: 56       Training Loss: 4.6106   Validation Loss: 4.4203
                Training Accuracy: 3.82%         Validation Accuracy: 6.32%

Epoch: 57       Training Loss: 4.6113   Validation Loss: 4.4147
                Training Accuracy: 3.74%         Validation Accuracy: 5.91%

Epoch: 58       Training Loss: 4.5972   Validation Loss: 4.4014
                Training Accuracy: 3.95%         Validation Accuracy: 6.27%

Epoch: 59       Training Loss: 4.5982   Validation Loss: 4.3898
                Training Accuracy: 4.01%         Validation Accuracy: 6.24%

Epoch: 60       Training Loss: 4.5801   Validation Loss: 4.3773
                Training Accuracy: 3.85%         Validation Accuracy: 6.62%

Epoch: 61       Training Loss: 4.5806   Validation Loss: 4.3644
                Training Accuracy: 4.13%         Validation Accuracy: 6.86%

Epoch: 62       Training Loss: 4.5788   Validation Loss: 4.3549
                Training Accuracy: 4.09%         Validation Accuracy: 6.81%

Epoch: 63       Training Loss: 4.5689   Validation Loss: 4.3389
                Training Accuracy: 4.82%         Validation Accuracy: 7.10%

Epoch: 64       Training Loss: 4.5492   Validation Loss: 4.3257
                Training Accuracy: 4.09%         Validation Accuracy: 6.89%
```

```
Epoch: 65       Training Loss: 4.5230   Validation Loss: 4.3103
                Training Accuracy: 4.66%         Validation Accuracy: 6.90%

Epoch: 66       Training Loss: 4.5343   Validation Loss: 4.2991
                Training Accuracy: 4.39%         Validation Accuracy: 7.29%

Epoch: 67       Training Loss: 4.5250   Validation Loss: 4.2875
                Training Accuracy: 4.58%         Validation Accuracy: 7.38%

Epoch: 68       Training Loss: 4.5166   Validation Loss: 4.2754
                Training Accuracy: 5.13%         Validation Accuracy: 7.50%

Epoch: 69       Training Loss: 4.5083   Validation Loss: 4.2654
                Training Accuracy: 4.66%         Validation Accuracy: 7.77%

Epoch: 70       Training Loss: 4.5072   Validation Loss: 4.2632
                Training Accuracy: 4.70%         Validation Accuracy: 8.11%

Epoch: 71       Training Loss: 4.4875   Validation Loss: 4.2502
                Training Accuracy: 4.93%         Validation Accuracy: 7.56%

Epoch: 72       Training Loss: 4.4879   Validation Loss: 4.2383
                Training Accuracy: 4.90%         Validation Accuracy: 7.93%

Epoch: 73       Training Loss: 4.4907   Validation Loss: 4.2323
                Training Accuracy: 4.88%         Validation Accuracy: 8.25%

Epoch: 74       Training Loss: 4.4878   Validation Loss: 4.2316
                Training Accuracy: 4.70%         Validation Accuracy: 8.13%

Epoch: 75       Training Loss: 4.4899   Validation Loss: 4.2252
                Training Accuracy: 4.78%         Validation Accuracy: 8.26%

Epoch: 76       Training Loss: 4.4752   Validation Loss: 4.2151
                Training Accuracy: 5.09%         Validation Accuracy: 8.02%

Epoch: 77       Training Loss: 4.4725   Validation Loss: 4.2123
                Training Accuracy: 4.75%         Validation Accuracy: 8.41%

Epoch: 78       Training Loss: 4.4669   Validation Loss: 4.2034
                Training Accuracy: 4.97%         Validation Accuracy: 8.59%

Epoch: 79       Training Loss: 4.4677   Validation Loss: 4.1982
                Training Accuracy: 5.27%         Validation Accuracy: 8.01%

Epoch: 80       Training Loss: 4.4633   Validation Loss: 4.1936
                Training Accuracy: 4.55%         Validation Accuracy: 8.56%
```

```
Epoch: 81       Training Loss: 4.4569   Validation Loss: 4.1839
                Training Accuracy: 5.12%         Validation Accuracy: 8.71%

Epoch: 82       Training Loss: 4.4662   Validation Loss: 4.1816
                Training Accuracy: 5.13%         Validation Accuracy: 8.44%

Epoch: 83       Training Loss: 4.4599   Validation Loss: 4.1734
                Training Accuracy: 5.28%         Validation Accuracy: 8.73%

Epoch: 84       Training Loss: 4.4387   Validation Loss: 4.1642
                Training Accuracy: 5.16%         Validation Accuracy: 8.95%

Epoch: 85       Training Loss: 4.4389   Validation Loss: 4.1570
                Training Accuracy: 5.52%         Validation Accuracy: 9.15%

Epoch: 86       Training Loss: 4.4407   Validation Loss: 4.1523
                Training Accuracy: 5.49%         Validation Accuracy: 8.61%

Epoch: 87       Training Loss: 4.4224   Validation Loss: 4.1440
                Training Accuracy: 5.60%         Validation Accuracy: 8.94%

Epoch: 88       Training Loss: 4.4220   Validation Loss: 4.1421
                Training Accuracy: 5.64%         Validation Accuracy: 9.00%

Epoch: 89       Training Loss: 4.4243   Validation Loss: 4.1360
                Training Accuracy: 5.10%         Validation Accuracy: 9.16%

Epoch: 90       Training Loss: 4.4143   Validation Loss: 4.1281
                Training Accuracy: 5.75%         Validation Accuracy: 9.18%

Epoch: 91       Training Loss: 4.4208   Validation Loss: 4.1279
                Training Accuracy: 5.76%         Validation Accuracy: 9.28%

Epoch: 92       Training Loss: 4.4193   Validation Loss: 4.1199
                Training Accuracy: 5.82%         Validation Accuracy: 9.78%

Epoch: 93       Training Loss: 4.4282   Validation Loss: 4.1148
                Training Accuracy: 5.45%         Validation Accuracy: 9.31%

Epoch: 94       Training Loss: 4.4185   Validation Loss: 4.1096
                Training Accuracy: 6.06%         Validation Accuracy: 10.00%

Epoch: 95       Training Loss: 4.4076   Validation Loss: 4.1013
                Training Accuracy: 5.76%         Validation Accuracy: 9.58%

Epoch: 96       Training Loss: 4.4013   Validation Loss: 4.0976
                Training Accuracy: 5.87%         Validation Accuracy: 10.01%
```

```
Epoch: 97       Training Loss: 4.4075   Validation Loss: 4.0922
                Training Accuracy: 5.57%         Validation Accuracy: 9.82%

Epoch: 98       Training Loss: 4.4027   Validation Loss: 4.0863
                Training Accuracy: 5.70%         Validation Accuracy: 9.85%

Epoch: 99       Training Loss: 4.3977   Validation Loss: 4.0902
                Training Accuracy: 5.99%         Validation Accuracy: 9.61%

Epoch: 100      Training Loss: 4.3889   Validation Loss: 4.0754
                Training Accuracy: 5.79%         Validation Accuracy: 10.37%

Epoch: 101      Training Loss: 4.3922   Validation Loss: 4.0673
                Training Accuracy: 5.88%         Validation Accuracy: 10.01%

Epoch: 102      Training Loss: 4.3823   Validation Loss: 4.0602
                Training Accuracy: 6.06%         Validation Accuracy: 10.33%

Epoch: 103      Training Loss: 4.3685   Validation Loss: 4.0499
                Training Accuracy: 5.75%         Validation Accuracy: 10.48%

Epoch: 104      Training Loss: 4.3839   Validation Loss: 4.0506
                Training Accuracy: 6.60%         Validation Accuracy: 10.30%

Epoch: 105      Training Loss: 4.3697   Validation Loss: 4.0432
                Training Accuracy: 6.57%         Validation Accuracy: 11.03%

Epoch: 106      Training Loss: 4.3747   Validation Loss: 4.0370
                Training Accuracy: 5.99%         Validation Accuracy: 10.96%

Epoch: 107      Training Loss: 4.3718   Validation Loss: 4.0284
                Training Accuracy: 6.18%         Validation Accuracy: 10.72%

Epoch: 108      Training Loss: 4.3736   Validation Loss: 4.0302
                Training Accuracy: 6.05%         Validation Accuracy: 10.57%

Epoch: 109      Training Loss: 4.3615   Validation Loss: 4.0187
                Training Accuracy: 6.39%         Validation Accuracy: 11.36%

Epoch: 110      Training Loss: 4.3557   Validation Loss: 4.0107
                Training Accuracy: 6.11%         Validation Accuracy: 11.11%

Epoch: 111      Training Loss: 4.3498   Validation Loss: 4.0015
                Training Accuracy: 6.83%         Validation Accuracy: 10.82%

Epoch: 112      Training Loss: 4.3580   Validation Loss: 3.9940
                Training Accuracy: 5.91%         Validation Accuracy: 11.62%
```

```
Epoch: 113      Training Loss: 4.3355   Validation Loss: 3.9887
                Training Accuracy: 6.63%         Validation Accuracy: 11.18%

Epoch: 114      Training Loss: 4.3366   Validation Loss: 3.9830
                Training Accuracy: 6.68%         Validation Accuracy: 12.10%

Epoch: 115      Training Loss: 4.3416   Validation Loss: 3.9831
                Training Accuracy: 6.69%         Validation Accuracy: 11.15%

Epoch: 116      Training Loss: 4.3245   Validation Loss: 3.9709
                Training Accuracy: 6.42%         Validation Accuracy: 11.86%

Epoch: 117      Training Loss: 4.3266   Validation Loss: 3.9640
                Training Accuracy: 6.93%         Validation Accuracy: 11.78%

Epoch: 118      Training Loss: 4.3212   Validation Loss: 3.9561
                Training Accuracy: 6.77%         Validation Accuracy: 12.35%

Epoch: 119      Training Loss: 4.3234   Validation Loss: 3.9522
                Training Accuracy: 6.63%         Validation Accuracy: 12.05%

Epoch: 120      Training Loss: 4.3239   Validation Loss: 3.9554
                Training Accuracy: 6.24%         Validation Accuracy: 11.66%

Epoch: 121      Training Loss: 4.3348   Validation Loss: 3.9384
                Training Accuracy: 6.77%         Validation Accuracy: 11.89%

Epoch: 122      Training Loss: 4.3160   Validation Loss: 3.9318
                Training Accuracy: 6.92%         Validation Accuracy: 12.44%

Epoch: 123      Training Loss: 4.3096   Validation Loss: 3.9287
                Training Accuracy: 6.78%         Validation Accuracy: 11.42%

Epoch: 124      Training Loss: 4.3112   Validation Loss: 3.9289
                Training Accuracy: 6.96%         Validation Accuracy: 12.41%

Epoch: 125      Training Loss: 4.3090   Validation Loss: 3.9216
                Training Accuracy: 7.05%         Validation Accuracy: 12.47%

Epoch: 126      Training Loss: 4.2985   Validation Loss: 3.9036
                Training Accuracy: 6.81%         Validation Accuracy: 12.25%

Epoch: 127      Training Loss: 4.3130   Validation Loss: 3.9072
                Training Accuracy: 6.96%         Validation Accuracy: 12.57%

Epoch: 128      Training Loss: 4.3021   Validation Loss: 3.8963
                Training Accuracy: 6.89%         Validation Accuracy: 12.93%
```

```
Epoch: 129      Training Loss: 4.2856   Validation Loss: 3.8874
                Training Accuracy: 6.80%         Validation Accuracy: 12.84%

Epoch: 130      Training Loss: 4.2928   Validation Loss: 3.8784
                Training Accuracy: 6.89%         Validation Accuracy: 13.65%

Epoch: 131      Training Loss: 4.3021   Validation Loss: 3.8805
                Training Accuracy: 7.29%         Validation Accuracy: 13.52%

Epoch: 132      Training Loss: 4.2945   Validation Loss: 3.8662
                Training Accuracy: 7.16%         Validation Accuracy: 13.37%

Epoch: 133      Training Loss: 4.2704   Validation Loss: 3.8627
                Training Accuracy: 6.98%         Validation Accuracy: 12.99%

Epoch: 134      Training Loss: 4.2709   Validation Loss: 3.8626
                Training Accuracy: 7.80%         Validation Accuracy: 12.57%

Epoch: 135      Training Loss: 4.2711   Validation Loss: 3.8547
                Training Accuracy: 7.41%         Validation Accuracy: 13.07%

Epoch: 136      Training Loss: 4.2718   Validation Loss: 3.8453
                Training Accuracy: 7.56%         Validation Accuracy: 13.76%

Epoch: 137      Training Loss: 4.2554   Validation Loss: 3.8345
                Training Accuracy: 7.59%         Validation Accuracy: 13.50%

Epoch: 138      Training Loss: 4.2813   Validation Loss: 3.8330
                Training Accuracy: 6.93%         Validation Accuracy: 13.56%

Epoch: 139      Training Loss: 4.2591   Validation Loss: 3.8319
                Training Accuracy: 7.72%         Validation Accuracy: 13.88%

Epoch: 140      Training Loss: 4.2562   Validation Loss: 3.8235
                Training Accuracy: 7.53%         Validation Accuracy: 14.22%

Epoch: 141      Training Loss: 4.2387   Validation Loss: 3.8117
                Training Accuracy: 7.74%         Validation Accuracy: 13.74%

Epoch: 142      Training Loss: 4.2490   Validation Loss: 3.8075
                Training Accuracy: 7.95%         Validation Accuracy: 14.45%

Epoch: 143      Training Loss: 4.2474   Validation Loss: 3.8021
                Training Accuracy: 7.62%         Validation Accuracy: 14.12%

Epoch: 144      Training Loss: 4.2550   Validation Loss: 3.7972
                Training Accuracy: 7.32%         Validation Accuracy: 14.12%
```

```
Epoch: 145      Training Loss: 4.2354   Validation Loss: 3.7882
                Training Accuracy: 7.90%         Validation Accuracy: 14.67%

Epoch: 146      Training Loss: 4.2349   Validation Loss: 3.7777
                Training Accuracy: 7.65%         Validation Accuracy: 14.57%

Epoch: 147      Training Loss: 4.2456   Validation Loss: 3.7705
                Training Accuracy: 7.86%         Validation Accuracy: 14.78%

Epoch: 148      Training Loss: 4.2315   Validation Loss: 3.7740
                Training Accuracy: 7.98%         Validation Accuracy: 14.85%

Epoch: 149      Training Loss: 4.2347   Validation Loss: 3.7638
                Training Accuracy: 8.08%         Validation Accuracy: 15.09%

Epoch: 150      Training Loss: 4.2094   Validation Loss: 3.7516
                Training Accuracy: 8.08%         Validation Accuracy: 15.45%

Epoch: 151      Training Loss: 4.2183   Validation Loss: 3.7477
                Training Accuracy: 8.20%         Validation Accuracy: 15.37%

Epoch: 152      Training Loss: 4.2198   Validation Loss: 3.7431
                Training Accuracy: 8.01%         Validation Accuracy: 15.03%

Epoch: 153      Training Loss: 4.2094   Validation Loss: 3.7428
                Training Accuracy: 8.16%         Validation Accuracy: 14.84%

Epoch: 154      Training Loss: 4.1979   Validation Loss: 3.7280
                Training Accuracy: 7.98%         Validation Accuracy: 15.73%

Epoch: 155      Training Loss: 4.2122   Validation Loss: 3.7250
                Training Accuracy: 8.16%         Validation Accuracy: 15.27%

Epoch: 156      Training Loss: 4.1888   Validation Loss: 3.7121
                Training Accuracy: 8.47%         Validation Accuracy: 15.52%

Epoch: 157      Training Loss: 4.2141   Validation Loss: 3.7165
                Training Accuracy: 8.28%         Validation Accuracy: 15.64%

Epoch: 158      Training Loss: 4.1923   Validation Loss: 3.7219
                Training Accuracy: 8.44%         Validation Accuracy: 15.48%

Epoch: 159      Training Loss: 4.2083   Validation Loss: 3.7141
                Training Accuracy: 8.32%         Validation Accuracy: 15.55%

Epoch: 160      Training Loss: 4.2006   Validation Loss: 3.7016
                Training Accuracy: 8.73%         Validation Accuracy: 15.67%
```

```
Epoch: 161      Training Loss: 4.1840   Validation Loss: 3.6872
                Training Accuracy: 8.16%         Validation Accuracy: 16.71%

Epoch: 162      Training Loss: 4.2104   Validation Loss: 3.6882
                Training Accuracy: 8.56%         Validation Accuracy: 16.26%

Epoch: 163      Training Loss: 4.1994   Validation Loss: 3.6915
                Training Accuracy: 7.93%         Validation Accuracy: 16.18%

Epoch: 164      Training Loss: 4.1941   Validation Loss: 3.6887
                Training Accuracy: 8.37%         Validation Accuracy: 16.12%

Epoch: 165      Training Loss: 4.1926   Validation Loss: 3.6916
                Training Accuracy: 8.23%         Validation Accuracy: 16.26%

Epoch: 166      Training Loss: 4.1660   Validation Loss: 3.6690
                Training Accuracy: 8.70%         Validation Accuracy: 16.50%

Epoch: 167      Training Loss: 4.1800   Validation Loss: 3.6511
                Training Accuracy: 8.44%         Validation Accuracy: 17.07%

Epoch: 168      Training Loss: 4.1828   Validation Loss: 3.6531
                Training Accuracy: 8.94%         Validation Accuracy: 16.95%

Epoch: 169      Training Loss: 4.1732   Validation Loss: 3.6489
                Training Accuracy: 8.59%         Validation Accuracy: 17.10%

Epoch: 170      Training Loss: 4.1681   Validation Loss: 3.6491
                Training Accuracy: 8.95%         Validation Accuracy: 16.77%

Epoch: 171      Training Loss: 4.1842   Validation Loss: 3.6475
                Training Accuracy: 8.67%         Validation Accuracy: 17.05%

Epoch: 172      Training Loss: 4.1663   Validation Loss: 3.6349
                Training Accuracy: 8.83%         Validation Accuracy: 17.26%

Epoch: 173      Training Loss: 4.1733   Validation Loss: 3.6363
                Training Accuracy: 8.68%         Validation Accuracy: 17.40%

Epoch: 174      Training Loss: 4.1505   Validation Loss: 3.6279
                Training Accuracy: 8.74%         Validation Accuracy: 17.08%

Epoch: 175      Training Loss: 4.1522   Validation Loss: 3.6159
                Training Accuracy: 8.85%         Validation Accuracy: 17.63%

Epoch: 176      Training Loss: 4.1516   Validation Loss: 3.6021
                Training Accuracy: 8.80%         Validation Accuracy: 17.89%
```

```
Epoch: 177      Training Loss: 4.1565   Validation Loss: 3.5978
                Training Accuracy: 8.94%         Validation Accuracy: 18.05%

Epoch: 178      Training Loss: 4.1549   Validation Loss: 3.5992
                Training Accuracy: 8.86%         Validation Accuracy: 18.07%

Epoch: 179      Training Loss: 4.1635   Validation Loss: 3.5991
                Training Accuracy: 9.04%         Validation Accuracy: 18.23%

Epoch: 180      Training Loss: 4.1497   Validation Loss: 3.6080
                Training Accuracy: 8.83%         Validation Accuracy: 17.62%

Epoch: 181      Training Loss: 4.1583   Validation Loss: 3.5854
                Training Accuracy: 9.30%         Validation Accuracy: 18.98%

Epoch: 182      Training Loss: 4.1306   Validation Loss: 3.5799
                Training Accuracy: 9.45%         Validation Accuracy: 18.35%

Epoch: 183      Training Loss: 4.1382   Validation Loss: 3.5640
                Training Accuracy: 9.30%         Validation Accuracy: 19.45%

Epoch: 184      Training Loss: 4.1397   Validation Loss: 3.5657
                Training Accuracy: 9.67%         Validation Accuracy: 18.64%

Epoch: 185      Training Loss: 4.1342   Validation Loss: 3.5564
                Training Accuracy: 9.39%         Validation Accuracy: 19.27%

Epoch: 186      Training Loss: 4.1492   Validation Loss: 3.5682
                Training Accuracy: 9.30%         Validation Accuracy: 18.05%

Epoch: 187      Training Loss: 4.1396   Validation Loss: 3.5556
                Training Accuracy: 9.15%         Validation Accuracy: 18.98%

Epoch: 188      Training Loss: 4.1327   Validation Loss: 3.5560
                Training Accuracy: 8.67%         Validation Accuracy: 19.15%

Epoch: 189      Training Loss: 4.1071   Validation Loss: 3.5424
                Training Accuracy: 9.94%         Validation Accuracy: 19.21%

Epoch: 190      Training Loss: 4.1211   Validation Loss: 3.5381
                Training Accuracy: 9.30%         Validation Accuracy: 18.77%

Epoch: 191      Training Loss: 4.1268   Validation Loss: 3.5366
                Training Accuracy: 9.09%         Validation Accuracy: 18.89%

Epoch: 192      Training Loss: 4.1256   Validation Loss: 3.5327
                Training Accuracy: 9.75%         Validation Accuracy: 19.48%
```

```
Epoch: 193      Training Loss: 4.1096   Validation Loss: 3.5240
                Training Accuracy: 9.78%         Validation Accuracy: 18.79%

Epoch: 194      Training Loss: 4.1268   Validation Loss: 3.5273
                Training Accuracy: 9.76%         Validation Accuracy: 19.94%

Epoch: 195      Training Loss: 4.1032   Validation Loss: 3.5082
                Training Accuracy: 9.91%         Validation Accuracy: 19.73%

Epoch: 196      Training Loss: 4.1075   Validation Loss: 3.5174
                Training Accuracy: 10.00%        Validation Accuracy: 19.34%

Epoch: 197      Training Loss: 4.0880   Validation Loss: 3.5026
                Training Accuracy: 10.07%        Validation Accuracy: 19.91%

Epoch: 198      Training Loss: 4.1068   Validation Loss: 3.4930
                Training Accuracy: 9.64%         Validation Accuracy: 20.33%

Epoch: 199      Training Loss: 4.1078   Validation Loss: 3.4841
                Training Accuracy: 10.12%        Validation Accuracy: 20.52%

Epoch: 200      Training Loss: 4.0863   Validation Loss: 3.4768
                Training Accuracy: 9.54%         Validation Accuracy: 20.46%

Epoch: 201      Training Loss: 4.1186   Validation Loss: 3.4961
                Training Accuracy: 9.60%         Validation Accuracy: 20.40%

Epoch: 202      Training Loss: 4.0892   Validation Loss: 3.4772
                Training Accuracy: 10.22%        Validation Accuracy: 20.70%

Epoch: 203      Training Loss: 4.0840   Validation Loss: 3.4613
                Training Accuracy: 10.24%        Validation Accuracy: 20.76%

Epoch: 204      Training Loss: 4.0647   Validation Loss: 3.4590
                Training Accuracy: 10.18%        Validation Accuracy: 19.82%

Epoch: 205      Training Loss: 4.0951   Validation Loss: 3.4543
                Training Accuracy: 9.94%         Validation Accuracy: 20.43%

Epoch: 206      Training Loss: 4.0705   Validation Loss: 3.4491
                Training Accuracy: 10.70%        Validation Accuracy: 20.40%

Epoch: 207      Training Loss: 4.0968   Validation Loss: 3.4416
                Training Accuracy: 9.70%         Validation Accuracy: 21.00%

Epoch: 208      Training Loss: 4.0751   Validation Loss: 3.4447
                Training Accuracy: 10.64%        Validation Accuracy: 20.96%
```

```
Epoch: 209      Training Loss: 4.0807   Validation Loss: 3.4452
                Training Accuracy: 10.07%        Validation Accuracy: 20.93%

Epoch: 210      Training Loss: 4.0597   Validation Loss: 3.4322
                Training Accuracy: 10.85%        Validation Accuracy: 20.63%

Epoch: 211      Training Loss: 4.0582   Validation Loss: 3.4244
                Training Accuracy: 10.72%        Validation Accuracy: 21.93%

Epoch: 212      Training Loss: 4.0701   Validation Loss: 3.4080
                Training Accuracy: 10.07%        Validation Accuracy: 22.41%

Epoch: 213      Training Loss: 4.0874   Validation Loss: 3.4200
                Training Accuracy: 9.82%         Validation Accuracy: 21.51%

Epoch: 214      Training Loss: 4.0569   Validation Loss: 3.4049
                Training Accuracy: 10.52%        Validation Accuracy: 21.59%

Epoch: 215      Training Loss: 4.0709   Validation Loss: 3.4086
                Training Accuracy: 10.48%        Validation Accuracy: 21.92%

Epoch: 216      Training Loss: 4.0557   Validation Loss: 3.4005
                Training Accuracy: 10.75%        Validation Accuracy: 22.25%

Epoch: 217      Training Loss: 4.0435   Validation Loss: 3.3999
                Training Accuracy: 10.66%        Validation Accuracy: 22.07%

Epoch: 218      Training Loss: 4.0245   Validation Loss: 3.3803
                Training Accuracy: 10.66%        Validation Accuracy: 21.71%

Epoch: 219      Training Loss: 4.0661   Validation Loss: 3.3820
                Training Accuracy: 10.69%        Validation Accuracy: 22.74%

Epoch: 220      Training Loss: 4.0704   Validation Loss: 3.3846
                Training Accuracy: 10.64%        Validation Accuracy: 22.11%

Epoch: 221      Training Loss: 4.0469   Validation Loss: 3.3695
                Training Accuracy: 10.72%        Validation Accuracy: 22.65%

Epoch: 222      Training Loss: 4.0287   Validation Loss: 3.3672
                Training Accuracy: 11.17%        Validation Accuracy: 22.72%

Epoch: 223      Training Loss: 4.0638   Validation Loss: 3.3621
                Training Accuracy: 10.54%        Validation Accuracy: 22.56%

Epoch: 224      Training Loss: 4.0326   Validation Loss: 3.3517
                Training Accuracy: 11.83%        Validation Accuracy: 23.64%
```

```
Epoch: 225      Training Loss: 4.0227   Validation Loss: 3.3450
                Training Accuracy: 11.32%        Validation Accuracy: 22.60%

Epoch: 226      Training Loss: 4.0231   Validation Loss: 3.3453
                Training Accuracy: 11.11%        Validation Accuracy: 23.16%

Epoch: 227      Training Loss: 4.0344   Validation Loss: 3.3398
                Training Accuracy: 11.51%        Validation Accuracy: 23.02%

Epoch: 228      Training Loss: 4.0171   Validation Loss: 3.3359
                Training Accuracy: 10.85%        Validation Accuracy: 23.02%

Epoch: 229      Training Loss: 4.0008   Validation Loss: 3.3236
                Training Accuracy: 11.42%        Validation Accuracy: 23.71%

Epoch: 230      Training Loss: 4.0172   Validation Loss: 3.3252
                Training Accuracy: 11.24%        Validation Accuracy: 23.14%

Epoch: 231      Training Loss: 4.0192   Validation Loss: 3.3183
                Training Accuracy: 11.27%        Validation Accuracy: 23.29%

Epoch: 232      Training Loss: 4.0305   Validation Loss: 3.3199
                Training Accuracy: 11.23%        Validation Accuracy: 23.49%

Epoch: 233      Training Loss: 4.0007   Validation Loss: 3.2981
                Training Accuracy: 11.62%        Validation Accuracy: 24.45%

Epoch: 234      Training Loss: 4.0328   Validation Loss: 3.2985
                Training Accuracy: 11.18%        Validation Accuracy: 24.66%

Epoch: 235      Training Loss: 3.9970   Validation Loss: 3.2978
                Training Accuracy: 11.78%        Validation Accuracy: 24.57%

Epoch: 236      Training Loss: 4.0046   Validation Loss: 3.2853
                Training Accuracy: 11.71%        Validation Accuracy: 24.06%

Epoch: 237      Training Loss: 3.9934   Validation Loss: 3.2752
                Training Accuracy: 11.74%        Validation Accuracy: 24.15%

Epoch: 238      Training Loss: 4.0037   Validation Loss: 3.2766
                Training Accuracy: 11.39%        Validation Accuracy: 24.82%

Epoch: 239      Training Loss: 4.0197   Validation Loss: 3.2805
                Training Accuracy: 10.75%        Validation Accuracy: 24.24%

Epoch: 240      Training Loss: 3.9745   Validation Loss: 3.2566
                Training Accuracy: 11.95%        Validation Accuracy: 25.01%
```

```
Epoch: 241      Training Loss: 3.9924   Validation Loss: 3.2603
                Training Accuracy: 11.42%        Validation Accuracy: 24.91%

Epoch: 242      Training Loss: 4.0068   Validation Loss: 3.2614
                Training Accuracy: 11.36%        Validation Accuracy: 24.63%

Epoch: 243      Training Loss: 3.9952   Validation Loss: 3.2537
                Training Accuracy: 11.66%        Validation Accuracy: 25.42%

Epoch: 244      Training Loss: 3.9865   Validation Loss: 3.2499
                Training Accuracy: 11.60%        Validation Accuracy: 24.90%

Epoch: 245      Training Loss: 3.9711   Validation Loss: 3.2328
                Training Accuracy: 12.04%        Validation Accuracy: 25.34%

Epoch: 246      Training Loss: 3.9528   Validation Loss: 3.2306
                Training Accuracy: 12.86%        Validation Accuracy: 24.96%

Epoch: 247      Training Loss: 3.9604   Validation Loss: 3.2160
                Training Accuracy: 11.78%        Validation Accuracy: 25.70%

Epoch: 248      Training Loss: 3.9643   Validation Loss: 3.2108
                Training Accuracy: 12.26%        Validation Accuracy: 25.97%

Epoch: 249      Training Loss: 3.9657   Validation Loss: 3.2133
                Training Accuracy: 12.05%        Validation Accuracy: 26.05%

Epoch: 250      Training Loss: 3.9684   Validation Loss: 3.2053
                Training Accuracy: 12.13%        Validation Accuracy: 26.00%

Epoch: 251      Training Loss: 3.9595   Validation Loss: 3.2138
                Training Accuracy: 12.20%        Validation Accuracy: 25.85%

Epoch: 252      Training Loss: 3.9752   Validation Loss: 3.1935
                Training Accuracy: 12.17%        Validation Accuracy: 26.36%

Epoch: 253      Training Loss: 3.9688   Validation Loss: 3.1918
                Training Accuracy: 12.40%        Validation Accuracy: 25.42%

Epoch: 254      Training Loss: 3.9519   Validation Loss: 3.1963
                Training Accuracy: 12.34%        Validation Accuracy: 26.59%

Epoch: 255      Training Loss: 3.9426   Validation Loss: 3.1815
                Training Accuracy: 12.17%        Validation Accuracy: 27.01%

Epoch: 256      Training Loss: 3.9671   Validation Loss: 3.1647
                Training Accuracy: 12.05%        Validation Accuracy: 26.47%
```

```
Epoch: 257      Training Loss: 3.9627   Validation Loss: 3.1661
                Training Accuracy: 11.99%        Validation Accuracy: 28.11%

Epoch: 258      Training Loss: 3.9740   Validation Loss: 3.1650
                Training Accuracy: 12.17%        Validation Accuracy: 26.68%

Epoch: 259      Training Loss: 3.9466   Validation Loss: 3.1641
                Training Accuracy: 12.53%        Validation Accuracy: 27.25%

Epoch: 260      Training Loss: 3.9507   Validation Loss: 3.1655
                Training Accuracy: 12.59%        Validation Accuracy: 27.11%

Epoch: 261      Training Loss: 3.9499   Validation Loss: 3.1505
                Training Accuracy: 12.26%        Validation Accuracy: 27.13%

Epoch: 262      Training Loss: 3.9093   Validation Loss: 3.1295
                Training Accuracy: 12.99%        Validation Accuracy: 26.80%

Epoch: 263      Training Loss: 3.9124   Validation Loss: 3.1321
                Training Accuracy: 12.99%        Validation Accuracy: 27.63%

Epoch: 264      Training Loss: 3.9315   Validation Loss: 3.1302
                Training Accuracy: 13.22%        Validation Accuracy: 27.43%

Epoch: 265      Training Loss: 3.9251   Validation Loss: 3.1280
                Training Accuracy: 13.29%        Validation Accuracy: 28.41%

Epoch: 266      Training Loss: 3.8927   Validation Loss: 3.0961
                Training Accuracy: 13.05%        Validation Accuracy: 27.54%

Epoch: 267      Training Loss: 3.9423   Validation Loss: 3.1121
                Training Accuracy: 12.78%        Validation Accuracy: 27.81%

Epoch: 268      Training Loss: 3.9177   Validation Loss: 3.1053
                Training Accuracy: 12.59%        Validation Accuracy: 27.10%

Epoch: 269      Training Loss: 3.9028   Validation Loss: 3.0863
                Training Accuracy: 13.08%        Validation Accuracy: 28.65%

Epoch: 270      Training Loss: 3.9033   Validation Loss: 3.0927
                Training Accuracy: 13.38%        Validation Accuracy: 28.31%

Epoch: 271      Training Loss: 3.9136   Validation Loss: 3.0745
                Training Accuracy: 12.65%        Validation Accuracy: 29.16%

Epoch: 272      Training Loss: 3.9325   Validation Loss: 3.0855
                Training Accuracy: 12.87%        Validation Accuracy: 29.67%
```

```
Epoch: 273      Training Loss: 3.9043   Validation Loss: 3.0903
                Training Accuracy: 13.13%        Validation Accuracy: 28.40%

Epoch: 274      Training Loss: 3.8882   Validation Loss: 3.0610
                Training Accuracy: 14.12%        Validation Accuracy: 29.48%

Epoch: 275      Training Loss: 3.9054   Validation Loss: 3.0728
                Training Accuracy: 13.61%        Validation Accuracy: 28.82%

Epoch: 276      Training Loss: 3.9105   Validation Loss: 3.0513
                Training Accuracy: 13.04%        Validation Accuracy: 29.81%

Epoch: 277      Training Loss: 3.9033   Validation Loss: 3.0486
                Training Accuracy: 13.17%        Validation Accuracy: 30.04%

Epoch: 278      Training Loss: 3.8740   Validation Loss: 3.0307
                Training Accuracy: 13.53%        Validation Accuracy: 30.55%

Epoch: 279      Training Loss: 3.8726   Validation Loss: 3.0206
                Training Accuracy: 13.85%        Validation Accuracy: 30.58%

Epoch: 280      Training Loss: 3.8943   Validation Loss: 3.0362
                Training Accuracy: 13.50%        Validation Accuracy: 30.24%

Epoch: 281      Training Loss: 3.8648   Validation Loss: 3.0225
                Training Accuracy: 13.80%        Validation Accuracy: 30.40%

Epoch: 282      Training Loss: 3.8706   Validation Loss: 3.0136
                Training Accuracy: 14.00%        Validation Accuracy: 30.36%

Epoch: 283      Training Loss: 3.8578   Validation Loss: 3.0092
                Training Accuracy: 13.85%        Validation Accuracy: 30.01%

Epoch: 284      Training Loss: 3.8988   Validation Loss: 2.9974
                Training Accuracy: 13.07%        Validation Accuracy: 31.75%

Epoch: 285      Training Loss: 3.8556   Validation Loss: 2.9790
                Training Accuracy: 14.27%        Validation Accuracy: 31.69%

Epoch: 286      Training Loss: 3.8666   Validation Loss: 2.9862
                Training Accuracy: 13.88%        Validation Accuracy: 30.70%

Epoch: 287      Training Loss: 3.8654   Validation Loss: 2.9868
                Training Accuracy: 14.10%        Validation Accuracy: 31.44%

Epoch: 288      Training Loss: 3.8652   Validation Loss: 2.9866
                Training Accuracy: 13.86%        Validation Accuracy: 30.91%
```

```
Epoch: 289      Training Loss: 3.8646   Validation Loss: 2.9687
                Training Accuracy: 13.97%        Validation Accuracy: 31.75%

Epoch: 290      Training Loss: 3.8393   Validation Loss: 2.9579
                Training Accuracy: 14.19%        Validation Accuracy: 31.90%

Epoch: 291      Training Loss: 3.8607   Validation Loss: 2.9622
                Training Accuracy: 13.80%        Validation Accuracy: 31.84%

Epoch: 292      Training Loss: 3.8481   Validation Loss: 2.9500
                Training Accuracy: 13.86%        Validation Accuracy: 32.43%

Epoch: 293      Training Loss: 3.8390   Validation Loss: 2.9449
                Training Accuracy: 14.36%        Validation Accuracy: 31.98%

Epoch: 294      Training Loss: 3.8295   Validation Loss: 2.9274
                Training Accuracy: 13.83%        Validation Accuracy: 33.19%

Epoch: 295      Training Loss: 3.8447   Validation Loss: 2.9198
                Training Accuracy: 14.30%        Validation Accuracy: 32.49%

Epoch: 296      Training Loss: 3.8387   Validation Loss: 2.9120
                Training Accuracy: 13.94%        Validation Accuracy: 33.23%

Epoch: 297      Training Loss: 3.8361   Validation Loss: 2.9073
                Training Accuracy: 14.46%        Validation Accuracy: 32.63%

Epoch: 298      Training Loss: 3.8268   Validation Loss: 2.9008
                Training Accuracy: 14.60%        Validation Accuracy: 32.68%

Epoch: 299      Training Loss: 3.8264   Validation Loss: 2.8918
                Training Accuracy: 14.25%        Validation Accuracy: 33.49%

Epoch: 300      Training Loss: 3.8019   Validation Loss: 2.8954
                Training Accuracy: 15.28%        Validation Accuracy: 32.80%

Epoch: 301      Training Loss: 3.8021   Validation Loss: 2.8937
                Training Accuracy: 15.10%        Validation Accuracy: 32.77%

Epoch: 302      Training Loss: 3.8171   Validation Loss: 2.8938
                Training Accuracy: 14.16%        Validation Accuracy: 32.56%

Epoch: 303      Training Loss: 3.8095   Validation Loss: 2.8766
                Training Accuracy: 14.85%        Validation Accuracy: 33.52%

Epoch: 304      Training Loss: 3.8093   Validation Loss: 2.8743
                Training Accuracy: 15.24%        Validation Accuracy: 33.25%
```

```
Epoch: 305      Training Loss: 3.8111   Validation Loss: 2.8573
                Training Accuracy: 14.72%        Validation Accuracy: 35.00%

Epoch: 306      Training Loss: 3.7980   Validation Loss: 2.8496
                Training Accuracy: 14.79%        Validation Accuracy: 34.91%

Epoch: 307      Training Loss: 3.7849   Validation Loss: 2.8418
                Training Accuracy: 15.46%        Validation Accuracy: 34.10%

Epoch: 308      Training Loss: 3.7872   Validation Loss: 2.8307
                Training Accuracy: 15.18%        Validation Accuracy: 34.94%

Epoch: 309      Training Loss: 3.8122   Validation Loss: 2.8393
                Training Accuracy: 14.48%        Validation Accuracy: 34.67%

Epoch: 310      Training Loss: 3.7791   Validation Loss: 2.8289
                Training Accuracy: 15.81%        Validation Accuracy: 34.48%

Epoch: 311      Training Loss: 3.7971   Validation Loss: 2.8257
                Training Accuracy: 15.58%        Validation Accuracy: 35.54%

Epoch: 312      Training Loss: 3.7825   Validation Loss: 2.8159
                Training Accuracy: 15.57%        Validation Accuracy: 34.82%

Epoch: 313      Training Loss: 3.7864   Validation Loss: 2.8074
                Training Accuracy: 14.85%        Validation Accuracy: 34.91%

Epoch: 314      Training Loss: 3.7874   Validation Loss: 2.8152
                Training Accuracy: 15.31%        Validation Accuracy: 34.97%

Epoch: 315      Training Loss: 3.7846   Validation Loss: 2.7987
                Training Accuracy: 15.30%        Validation Accuracy: 35.63%

Epoch: 316      Training Loss: 3.7783   Validation Loss: 2.7904
                Training Accuracy: 15.10%        Validation Accuracy: 35.85%

Epoch: 317      Training Loss: 3.7811   Validation Loss: 2.8055
                Training Accuracy: 15.33%        Validation Accuracy: 35.25%

Epoch: 318      Training Loss: 3.7720   Validation Loss: 2.7708
                Training Accuracy: 15.60%        Validation Accuracy: 35.70%

Epoch: 319      Training Loss: 3.7591   Validation Loss: 2.7710
                Training Accuracy: 16.12%        Validation Accuracy: 36.15%

Epoch: 320      Training Loss: 3.7675   Validation Loss: 2.7771
                Training Accuracy: 15.55%        Validation Accuracy: 36.23%
```

```
Epoch: 321      Training Loss: 3.7565   Validation Loss: 2.7590
                Training Accuracy: 16.05%        Validation Accuracy: 37.07%

Epoch: 322      Training Loss: 3.7480   Validation Loss: 2.7512
                Training Accuracy: 15.94%        Validation Accuracy: 36.12%

Epoch: 323      Training Loss: 3.7364   Validation Loss: 2.7403
                Training Accuracy: 16.27%        Validation Accuracy: 37.63%

Epoch: 324      Training Loss: 3.7550   Validation Loss: 2.7491
                Training Accuracy: 15.49%        Validation Accuracy: 37.40%

Epoch: 325      Training Loss: 3.7416   Validation Loss: 2.7396
                Training Accuracy: 15.93%        Validation Accuracy: 36.24%

Epoch: 326      Training Loss: 3.7484   Validation Loss: 2.7493
                Training Accuracy: 16.26%        Validation Accuracy: 35.91%

Epoch: 327      Training Loss: 3.7324   Validation Loss: 2.7329
                Training Accuracy: 16.69%        Validation Accuracy: 36.99%

Epoch: 328      Training Loss: 3.7304   Validation Loss: 2.7214
                Training Accuracy: 16.09%        Validation Accuracy: 36.90%

Epoch: 329      Training Loss: 3.7338   Validation Loss: 2.7239
                Training Accuracy: 16.54%        Validation Accuracy: 37.81%

Epoch: 330      Training Loss: 3.7260   Validation Loss: 2.6926
                Training Accuracy: 16.74%        Validation Accuracy: 38.73%

Epoch: 331      Training Loss: 3.7182   Validation Loss: 2.6784
                Training Accuracy: 16.53%        Validation Accuracy: 39.87%

Epoch: 332      Training Loss: 3.6952   Validation Loss: 2.6937
                Training Accuracy: 17.17%        Validation Accuracy: 38.26%

Epoch: 333      Training Loss: 3.7339   Validation Loss: 2.6835
                Training Accuracy: 16.68%        Validation Accuracy: 39.13%

Epoch: 334      Training Loss: 3.7329   Validation Loss: 2.6776
                Training Accuracy: 16.02%        Validation Accuracy: 38.53%

Epoch: 335      Training Loss: 3.7135   Validation Loss: 2.6673
                Training Accuracy: 16.84%        Validation Accuracy: 39.64%

Epoch: 336      Training Loss: 3.7156   Validation Loss: 2.6595
                Training Accuracy: 16.51%        Validation Accuracy: 39.48%
```

```
Epoch: 337      Training Loss: 3.7147   Validation Loss: 2.6571
                Training Accuracy: 16.99%       Validation Accuracy: 39.30%

Epoch: 338      Training Loss: 3.6963   Validation Loss: 2.6747
                Training Accuracy: 17.10%       Validation Accuracy: 38.98%

Epoch: 339      Training Loss: 3.6934   Validation Loss: 2.6495
                Training Accuracy: 17.10%       Validation Accuracy: 39.97%

Epoch: 340      Training Loss: 3.6990   Validation Loss: 2.6350
                Training Accuracy: 17.51%       Validation Accuracy: 39.51%

Epoch: 341      Training Loss: 3.6945   Validation Loss: 2.6339
                Training Accuracy: 16.84%       Validation Accuracy: 40.09%

Epoch: 342      Training Loss: 3.6820   Validation Loss: 2.6279
                Training Accuracy: 16.87%       Validation Accuracy: 39.43%

Epoch: 343      Training Loss: 3.6877   Validation Loss: 2.6158
                Training Accuracy: 17.54%       Validation Accuracy: 40.40%

Epoch: 344      Training Loss: 3.6860   Validation Loss: 2.5943
                Training Accuracy: 17.37%       Validation Accuracy: 40.69%

Epoch: 345      Training Loss: 3.6814   Validation Loss: 2.6002
                Training Accuracy: 17.25%       Validation Accuracy: 40.61%

Epoch: 346      Training Loss: 3.6886   Validation Loss: 2.5876
                Training Accuracy: 17.34%       Validation Accuracy: 41.23%

Epoch: 347      Training Loss: 3.6439   Validation Loss: 2.5768
                Training Accuracy: 17.99%       Validation Accuracy: 41.63%

Epoch: 348      Training Loss: 3.6809   Validation Loss: 2.5839
                Training Accuracy: 17.75%       Validation Accuracy: 40.63%

Epoch: 349      Training Loss: 3.6709   Validation Loss: 2.5654
                Training Accuracy: 17.57%       Validation Accuracy: 41.00%

Epoch: 350      Training Loss: 3.6753   Validation Loss: 2.5793
                Training Accuracy: 17.57%       Validation Accuracy: 41.90%

Epoch: 351      Training Loss: 3.6642   Validation Loss: 2.5517
                Training Accuracy: 17.28%       Validation Accuracy: 42.53%

Epoch: 352      Training Loss: 3.6562   Validation Loss: 2.5659
                Training Accuracy: 17.93%       Validation Accuracy: 41.42%
```

```
Epoch: 353      Training Loss: 3.6692   Validation Loss: 2.5556
                Training Accuracy: 17.46%        Validation Accuracy: 41.93%

Epoch: 354      Training Loss: 3.6572   Validation Loss: 2.5421
                Training Accuracy: 17.56%        Validation Accuracy: 42.78%

Epoch: 355      Training Loss: 3.6409   Validation Loss: 2.5349
                Training Accuracy: 17.77%        Validation Accuracy: 42.54%

Epoch: 356      Training Loss: 3.6200   Validation Loss: 2.5341
                Training Accuracy: 18.34%        Validation Accuracy: 41.95%

Epoch: 357      Training Loss: 3.6181   Validation Loss: 2.5283
                Training Accuracy: 18.23%        Validation Accuracy: 41.47%

Epoch: 358      Training Loss: 3.6314   Validation Loss: 2.5063
                Training Accuracy: 17.78%        Validation Accuracy: 42.53%

Epoch: 359      Training Loss: 3.6074   Validation Loss: 2.5105
                Training Accuracy: 18.92%        Validation Accuracy: 42.62%

Epoch: 360      Training Loss: 3.6234   Validation Loss: 2.5135
                Training Accuracy: 19.16%        Validation Accuracy: 43.22%

Epoch: 361      Training Loss: 3.6262   Validation Loss: 2.4889
                Training Accuracy: 18.20%        Validation Accuracy: 44.03%

Epoch: 362      Training Loss: 3.6129   Validation Loss: 2.5038
                Training Accuracy: 18.41%        Validation Accuracy: 43.14%

Epoch: 363      Training Loss: 3.6155   Validation Loss: 2.4805
                Training Accuracy: 19.10%        Validation Accuracy: 43.91%

Epoch: 364      Training Loss: 3.5972   Validation Loss: 2.4827
                Training Accuracy: 18.83%        Validation Accuracy: 43.89%

Epoch: 365      Training Loss: 3.6194   Validation Loss: 2.4660
                Training Accuracy: 18.74%        Validation Accuracy: 44.88%

Epoch: 366      Training Loss: 3.6221   Validation Loss: 2.4660
                Training Accuracy: 18.19%        Validation Accuracy: 45.07%

Epoch: 367      Training Loss: 3.5884   Validation Loss: 2.4443
                Training Accuracy: 19.58%        Validation Accuracy: 44.03%

Epoch: 368      Training Loss: 3.6021   Validation Loss: 2.4416
                Training Accuracy: 19.24%        Validation Accuracy: 44.64%
```

```
Epoch: 369      Training Loss: 3.5710   Validation Loss: 2.4384
                Training Accuracy: 19.01%        Validation Accuracy: 44.54%

Epoch: 370      Training Loss: 3.6062   Validation Loss: 2.4242
                Training Accuracy: 18.77%        Validation Accuracy: 45.81%

Epoch: 371      Training Loss: 3.5842   Validation Loss: 2.4222
                Training Accuracy: 19.25%        Validation Accuracy: 46.33%

Epoch: 372      Training Loss: 3.5898   Validation Loss: 2.4341
                Training Accuracy: 19.09%        Validation Accuracy: 44.31%

Epoch: 373      Training Loss: 3.5818   Validation Loss: 2.4053
                Training Accuracy: 18.61%        Validation Accuracy: 45.69%

Epoch: 374      Training Loss: 3.5729   Validation Loss: 2.4098
                Training Accuracy: 18.98%        Validation Accuracy: 45.39%

Epoch: 375      Training Loss: 3.5726   Validation Loss: 2.4200
                Training Accuracy: 19.70%        Validation Accuracy: 44.96%

Epoch: 376      Training Loss: 3.5813   Validation Loss: 2.4060
                Training Accuracy: 19.22%        Validation Accuracy: 45.72%

Epoch: 377      Training Loss: 3.5993   Validation Loss: 2.3902
                Training Accuracy: 18.79%        Validation Accuracy: 46.74%

Epoch: 378      Training Loss: 3.5763   Validation Loss: 2.3835
                Training Accuracy: 19.46%        Validation Accuracy: 46.72%

Epoch: 379      Training Loss: 3.5963   Validation Loss: 2.3761
                Training Accuracy: 19.34%        Validation Accuracy: 47.01%

Epoch: 380      Training Loss: 3.5618   Validation Loss: 2.3938
                Training Accuracy: 19.04%        Validation Accuracy: 46.05%

Epoch: 381      Training Loss: 3.5631   Validation Loss: 2.3813
                Training Accuracy: 19.94%        Validation Accuracy: 45.99%

Epoch: 382      Training Loss: 3.5626   Validation Loss: 2.3740
                Training Accuracy: 20.43%        Validation Accuracy: 45.25%

Epoch: 383      Training Loss: 3.5494   Validation Loss: 2.3620
                Training Accuracy: 18.95%        Validation Accuracy: 46.60%

Epoch: 384      Training Loss: 3.5532   Validation Loss: 2.3418
                Training Accuracy: 20.34%        Validation Accuracy: 46.71%
```

```
Epoch: 385      Training Loss: 3.5646   Validation Loss: 2.3554
                Training Accuracy: 19.34%        Validation Accuracy: 46.17%

Epoch: 386      Training Loss: 3.5431   Validation Loss: 2.3392
                Training Accuracy: 19.64%        Validation Accuracy: 46.95%

Epoch: 387      Training Loss: 3.5117   Validation Loss: 2.3178
                Training Accuracy: 20.39%        Validation Accuracy: 48.04%

Epoch: 388      Training Loss: 3.5239   Validation Loss: 2.3089
                Training Accuracy: 19.85%        Validation Accuracy: 48.43%

Epoch: 389      Training Loss: 3.5363   Validation Loss: 2.3311
                Training Accuracy: 20.75%        Validation Accuracy: 48.20%

Epoch: 390      Training Loss: 3.5291   Validation Loss: 2.3142
                Training Accuracy: 20.42%        Validation Accuracy: 48.44%

Epoch: 391      Training Loss: 3.5272   Validation Loss: 2.3001
                Training Accuracy: 20.52%        Validation Accuracy: 48.53%

Epoch: 392      Training Loss: 3.5159   Validation Loss: 2.3037
                Training Accuracy: 20.60%        Validation Accuracy: 47.50%

Epoch: 393      Training Loss: 3.5178   Validation Loss: 2.3020
                Training Accuracy: 20.39%        Validation Accuracy: 47.66%

Epoch: 394      Training Loss: 3.5174   Validation Loss: 2.2670
                Training Accuracy: 20.52%        Validation Accuracy: 49.54%

Epoch: 395      Training Loss: 3.5241   Validation Loss: 2.2859
                Training Accuracy: 21.06%        Validation Accuracy: 47.65%

Epoch: 396      Training Loss: 3.4842   Validation Loss: 2.2543
                Training Accuracy: 20.76%        Validation Accuracy: 48.77%

Epoch: 397      Training Loss: 3.5147   Validation Loss: 2.2643
                Training Accuracy: 21.29%        Validation Accuracy: 49.24%

Epoch: 398      Training Loss: 3.4908   Validation Loss: 2.2464
                Training Accuracy: 21.26%        Validation Accuracy: 50.63%

Epoch: 399      Training Loss: 3.5003   Validation Loss: 2.2596
                Training Accuracy: 20.82%        Validation Accuracy: 49.67%

Epoch: 400      Training Loss: 3.4919   Validation Loss: 2.2318
                Training Accuracy: 21.21%        Validation Accuracy: 49.64%
```

```
Epoch: 401      Training Loss: 3.5122   Validation Loss: 2.2309
                Training Accuracy: 21.12%        Validation Accuracy: 50.01%

Epoch: 402      Training Loss: 3.4828   Validation Loss: 2.2324
                Training Accuracy: 21.39%        Validation Accuracy: 50.78%

Epoch: 403      Training Loss: 3.4876   Validation Loss: 2.2219
                Training Accuracy: 21.05%        Validation Accuracy: 50.76%

Epoch: 404      Training Loss: 3.4799   Validation Loss: 2.2045
                Training Accuracy: 21.32%        Validation Accuracy: 51.23%

Epoch: 405      Training Loss: 3.4761   Validation Loss: 2.1928
                Training Accuracy: 21.68%        Validation Accuracy: 51.51%

Epoch: 406      Training Loss: 3.4591   Validation Loss: 2.1929
                Training Accuracy: 22.38%        Validation Accuracy: 49.94%

Epoch: 407      Training Loss: 3.4653   Validation Loss: 2.1938
                Training Accuracy: 21.95%        Validation Accuracy: 51.09%

Epoch: 408      Training Loss: 3.4826   Validation Loss: 2.1850
                Training Accuracy: 20.99%        Validation Accuracy: 50.61%

Epoch: 409      Training Loss: 3.4328   Validation Loss: 2.1759
                Training Accuracy: 22.56%        Validation Accuracy: 50.87%

Epoch: 410      Training Loss: 3.4427   Validation Loss: 2.1784
                Training Accuracy: 22.23%        Validation Accuracy: 50.34%

Epoch: 411      Training Loss: 3.4732   Validation Loss: 2.1571
                Training Accuracy: 21.26%        Validation Accuracy: 51.92%

Epoch: 412      Training Loss: 3.4501   Validation Loss: 2.1515
                Training Accuracy: 21.65%        Validation Accuracy: 52.26%

Epoch: 413      Training Loss: 3.4537   Validation Loss: 2.1600
                Training Accuracy: 22.10%        Validation Accuracy: 50.96%

Epoch: 414      Training Loss: 3.4689   Validation Loss: 2.1326
                Training Accuracy: 22.04%        Validation Accuracy: 52.89%

Epoch: 415      Training Loss: 3.4379   Validation Loss: 2.1207
                Training Accuracy: 21.60%        Validation Accuracy: 52.68%

Epoch: 416      Training Loss: 3.4220   Validation Loss: 2.1141
                Training Accuracy: 22.63%        Validation Accuracy: 53.38%
```

```
Epoch: 417      Training Loss: 3.4363   Validation Loss: 2.1142
                Training Accuracy: 22.17%        Validation Accuracy: 53.02%

Epoch: 418      Training Loss: 3.4310   Validation Loss: 2.1236
                Training Accuracy: 22.20%        Validation Accuracy: 52.26%

Epoch: 419      Training Loss: 3.4399   Validation Loss: 2.0940
                Training Accuracy: 22.25%        Validation Accuracy: 54.15%

Epoch: 420      Training Loss: 3.4183   Validation Loss: 2.1157
                Training Accuracy: 22.22%        Validation Accuracy: 52.68%

Epoch: 421      Training Loss: 3.4102   Validation Loss: 2.0981
                Training Accuracy: 22.16%        Validation Accuracy: 53.04%

Epoch: 422      Training Loss: 3.4124   Validation Loss: 2.0899
                Training Accuracy: 22.77%        Validation Accuracy: 52.99%

Epoch: 423      Training Loss: 3.4236   Validation Loss: 2.0881
                Training Accuracy: 22.05%        Validation Accuracy: 54.34%

Epoch: 424      Training Loss: 3.4117   Validation Loss: 2.0772
                Training Accuracy: 23.34%        Validation Accuracy: 54.45%

Epoch: 425      Training Loss: 3.4155   Validation Loss: 2.0792
                Training Accuracy: 22.92%        Validation Accuracy: 54.12%

Epoch: 426      Training Loss: 3.4174   Validation Loss: 2.0571
                Training Accuracy: 22.89%        Validation Accuracy: 55.22%

Epoch: 427      Training Loss: 3.4040   Validation Loss: 2.0394
                Training Accuracy: 23.20%        Validation Accuracy: 55.10%

Epoch: 428      Training Loss: 3.3928   Validation Loss: 2.0401
                Training Accuracy: 23.31%        Validation Accuracy: 55.61%

Epoch: 429      Training Loss: 3.4170   Validation Loss: 2.0310
                Training Accuracy: 22.63%        Validation Accuracy: 56.50%

Epoch: 430      Training Loss: 3.3944   Validation Loss: 2.0270
                Training Accuracy: 23.31%        Validation Accuracy: 56.02%

Epoch: 431      Training Loss: 3.3912   Validation Loss: 2.0211
                Training Accuracy: 22.93%        Validation Accuracy: 55.88%

Epoch: 432      Training Loss: 3.3964   Validation Loss: 2.0082
                Training Accuracy: 22.83%        Validation Accuracy: 56.06%
```

```
Epoch: 433      Training Loss: 3.4160   Validation Loss: 2.0093
                Training Accuracy: 22.34%        Validation Accuracy: 55.87%

Epoch: 434      Training Loss: 3.3526   Validation Loss: 2.0119
                Training Accuracy: 24.06%        Validation Accuracy: 56.32%

Epoch: 435      Training Loss: 3.3902   Validation Loss: 2.0051
                Training Accuracy: 23.47%        Validation Accuracy: 57.05%

Epoch: 436      Training Loss: 3.3810   Validation Loss: 1.9822
                Training Accuracy: 23.83%        Validation Accuracy: 57.22%

Epoch: 437      Training Loss: 3.3527   Validation Loss: 1.9830
                Training Accuracy: 24.24%        Validation Accuracy: 57.08%

Epoch: 438      Training Loss: 3.3265   Validation Loss: 1.9841
                Training Accuracy: 24.28%        Validation Accuracy: 57.46%

Epoch: 439      Training Loss: 3.3914   Validation Loss: 1.9863
                Training Accuracy: 23.83%        Validation Accuracy: 56.23%

Epoch: 440      Training Loss: 3.3296   Validation Loss: 1.9364
                Training Accuracy: 24.27%        Validation Accuracy: 57.98%

Epoch: 441      Training Loss: 3.3458   Validation Loss: 1.9587
                Training Accuracy: 24.24%        Validation Accuracy: 58.04%

Epoch: 442      Training Loss: 3.3546   Validation Loss: 1.9486
                Training Accuracy: 24.39%        Validation Accuracy: 58.02%

Epoch: 443      Training Loss: 3.3623   Validation Loss: 1.9633
                Training Accuracy: 24.00%        Validation Accuracy: 56.62%

Epoch: 444      Training Loss: 3.3242   Validation Loss: 1.9509
                Training Accuracy: 24.75%        Validation Accuracy: 57.04%

Epoch: 445      Training Loss: 3.3285   Validation Loss: 1.9392
                Training Accuracy: 24.66%        Validation Accuracy: 58.52%

Early Stopping! Total epochs: 445. Best epoch: 440 with loss: 1.94 and acc:
58.52%
37686.34 total seconds elapsed. 84.50 seconds per epoch.
```

Training and Validation Losses

[55]:
```
save_file_name = f'CNN_scratch_model.pt'
```

```
# Load saved model due to Colab timeout
scratch_model.load_state_dict(torch.load(save_file_name))
scratch_model.eval()
```

[55]:
```
<All keys matched successfully>
```

[55]:
```
dmodel(
  (conv1): Conv2d(3, 6, kernel_size=(5, 5), stride=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  (conv2): Conv2d(6, 16, kernel_size=(5, 5), stride=(1, 1))
  (fc1): Linear(in_features=44944, out_features=4096, bias=True)
  (fc2): Linear(in_features=4096, out_features=1024, bias=True)
  (fc3): Linear(in_features=1024, out_features=133, bias=True)
)
```

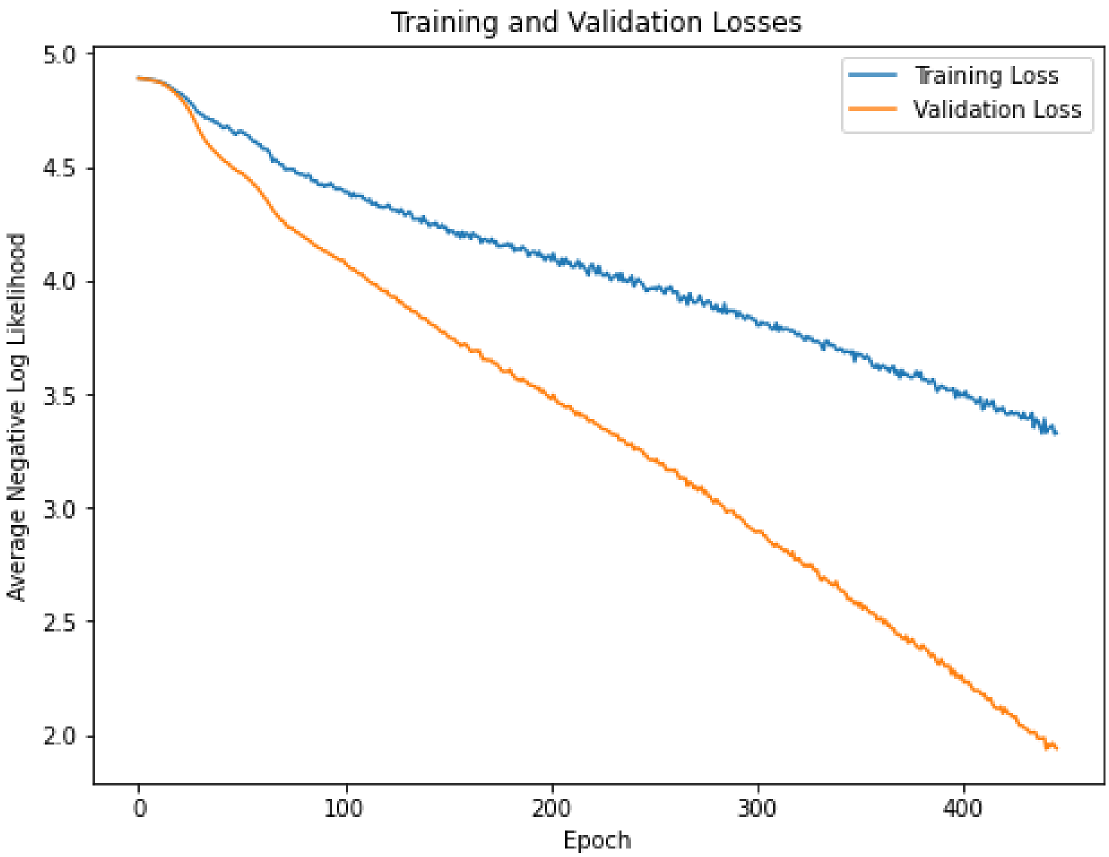
[75]:
```
epoch = []
training_loss = []
validation_loss = []
training_accuracy = []
validation_accuracy = []

# Open results file
file1 = open('scratch_results.txt', 'r')

# Parse lines of text
while True:
  # Line 1: Epoch, Trainging Loss, Validation Loss
  line1 = file1.readline()

  # Line 2: Trainging Accuracy, Validation Accuracy
  line2 = file1.readline()

  # End of file
  if not line2:
    break

  # Line 3: Empty line
  line3 = file1.readline()
  line3 = line3.strip()

  # Get data
  line1 = line1.strip().split('\t')
  epoch.append(int((line1[0].replace('Epoch: ', '')).strip()))
  training_loss.append(float((line1[1].replace('Training Loss: ', '')).strip()))
  validation_loss.append(float((line1[2].replace('Validation Loss: ', '')).
↪strip()))
```

```
    line2 = line2.strip().split('\t')
    training_accuracy.append(float(((line2[0].replace('Training Accuracy: ', '')).
↪strip()).replace('%', '')))
    validation_accuracy.append(float((line2[1].replace('Validation Accuracy: ',␣
↪'')).strip().replace('%', '')))
```

[79]:

```
plt.figure(figsize=(8, 6))
plt.plot(epoch, training_loss, label = 'Training Loss')
plt.plot(epoch, validation_loss, label = 'Validation Loss')
plt.legend()
plt.xlabel('Epoch')
plt.ylabel('Average Negative Log Likelihood')
plt.title('Training and Validation Losses')
plt.show()
plt.savefig('sratch_loss.png')
```

```
[79]: <Figure size 576x432 with 0 Axes>

[79]: [<matplotlib.lines.Line2D at 0x7fd5ce420a90>]

[79]: [<matplotlib.lines.Line2D at 0x7fd5ce420b90>]

[79]: <matplotlib.legend.Legend at 0x7fd660f81c50>

[79]: Text(0.5, 0, 'Epoch')

[79]: Text(0, 0.5, 'Average Negative Log Likelihood')

[79]: Text(0.5, 1.0, 'Training and Validation Losses')
```



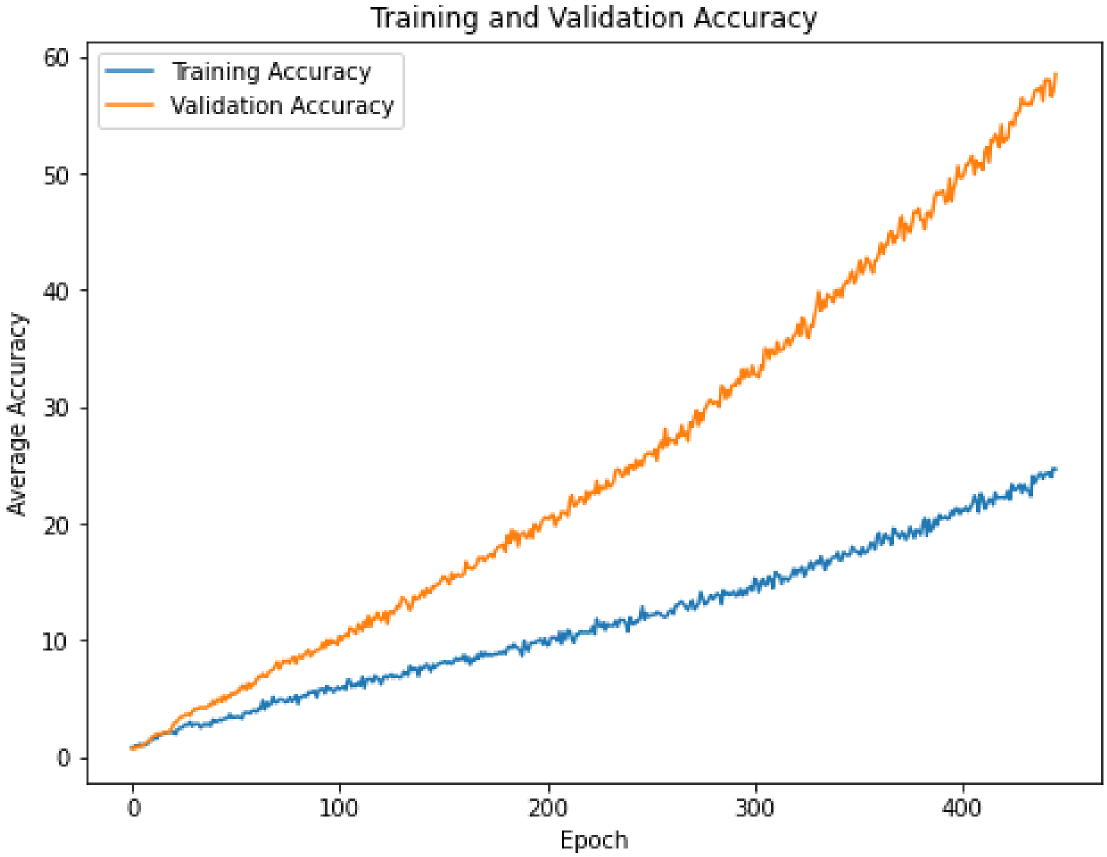
```
<Figure size 432x288 with 0 Axes>
```

Training and Validation Accuracy

```
[80]: plt.figure(figsize=(8, 6))
      plt.plot(epoch, training_accuracy, label = 'Training Accuracy')
      plt.plot(epoch, validation_accuracy, label = 'Validation Accuracy')
      plt.legend()
      plt.xlabel('Epoch')
      plt.ylabel('Average Accuracy')
      plt.title('Training and Validation Accuracy')
      plt.show()
```

```
[80]: <Figure size 576x432 with 0 Axes>

[80]: [<matplotlib.lines.Line2D at 0x7fd5ce3b1b10>]

[80]: [<matplotlib.lines.Line2D at 0x7fd5ce3b1e10>]

[80]: <matplotlib.legend.Legend at 0x7fd5ce3f8f90>
```

```
[80]: Text(0.5, 0, 'Epoch')

[80]: Text(0, 0.5, 'Average Accuracy')

[80]: Text(0.5, 1.0, 'Training and Validation Accuracy')
```



1.1.1 Testing

```
[128]: # Run test data through model
def test(model, criterion, optimizer, test_loader, train_on_gpu):

    # Test loss and accuracy
    test_loss = 0.
    correct = 0.
    total = 0.

    # Set model to evaluation mode
    model.eval()
    for batch_idx, (data, target) in enumerate(test_loader):
        # Set tensors to GPU
        if train_on_gpu:
            data, target = data.cuda(), target.cuda()

        # Forward pass: compute predicted outputs by passing inputs to the model
        output = model(data)

        # Calculate the loss
        loss = criterion(output, target)

        # Update average test loss
        test_loss = test_loss + ((1 / (batch_idx + 1)) * (loss.data -␣
 ↪test_loss))

        # convert output probabilities to predicted class
        pred = output.data.max(1, keepdim=True)[1]

        # compare predictions to true label
        correct += np.sum(np.squeeze(pred.eq(target.data.view_as(pred))).cpu().
 ↪numpy())
        total += data.size(0)

    print('Test Loss: {:.6f}\n'.format(test_loss))

    print('\nTest Accuracy: %2d%% (%2d/%2d)' % (
        100. * correct / total, correct, total))

# call test function
test(scratch_model, scratch_criterion, scratch_optimizer, dataloaders['test'],␣
 ↪train_on_gpu)
```

```
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:477:
UserWarning: This DataLoader will create 10 worker processes in total. Our
suggested max number of worker in current system is 4, which is smaller than
what this DataLoader is going to create. Please be aware that excessive worker
creation might get DataLoader running slow or even freeze, lower the worker
number to avoid potential slowness/freeze if necessary.
  cpuset_checked))

Test Loss: 1.936387



Test Accuracy: 57% (3873/6680)
```

[194]:

```
def wrong(data_path):
  # Pre-Process Data
  img_t = image_transforms['test'](Image.open(data_path).convert('RGB'))
```

```python
  batch_t = torch.unsqueeze(img_t, 0)

  if train_on_gpu:
    batch_t = batch_t.cuda()
  out = scratch_model(batch_t)

  # Get index of classification
  # _, index = torch.max(out, 1)
  index = out.data.max(1, keepdim=True)[1]
  percentage = torch.nn.functional.softmax(out, dim=1)[0] * 100
  return index[0] + 1, percentage

path = 'dogImages/test/'
listOfDir = os.listdir(path)
totalFiles = 0
wrongDogs = 0
breeds = {} # Dictionary of breeds
for subdir in listOfDir:
  files = os.listdir(path + subdir)
  subdir_str = subdir.split('.')
  key = int(subdir_str[0].strip())
  value = subdir_str[1].strip()
  breeds[key] = value
  # print(subdir_str)
  for filename in files:
    totalFiles += 1
    dogBreed, _ = wrong(path + subdir + '/' + filename)
    if dogBreed != key:
      print("Actual: %s \tPredicted: %d" % (key, dogBreed))
      wrongDogs += 1

print("ResNet50 Top-1 Error %d%% for breeds of dogs" % (((totalFiles -␣
 ↪wrongDogs) / totalFiles) * 100))
print("Correctly classified: %d / %d" % ((totalFiles - wrongDogs), totalFiles))
```

```
Actual: 90      Predicted: 43
Actual: 90      Predicted: 68
Actual: 90      Predicted: 4
Actual: 90      Predicted: 74
Actual: 90      Predicted: 21
Actual: 90      Predicted: 62
Actual: 90      Predicted: 37
Actual: 90      Predicted: 117
Actual: 95      Predicted: 37
Actual: 95      Predicted: 89
Actual: 95      Predicted: 99
Actual: 95      Predicted: 42
```

```
Actual: 95      Predicted: 94
Actual: 95      Predicted: 124
Actual: 116     Predicted: 131
Actual: 116     Predicted: 7
Actual: 116     Predicted: 62
Actual: 116     Predicted: 31
Actual: 92      Predicted: 109
Actual: 92      Predicted: 94
Actual: 92      Predicted: 94
Actual: 92      Predicted: 40
Actual: 92      Predicted: 75
Actual: 21      Predicted: 22
Actual: 21      Predicted: 68
Actual: 21      Predicted: 29
Actual: 21      Predicted: 29
Actual: 21      Predicted: 68
Actual: 21      Predicted: 109
Actual: 21      Predicted: 93
Actual: 100     Predicted: 80
Actual: 100     Predicted: 94
Actual: 100     Predicted: 74
Actual: 100     Predicted: 17
Actual: 12      Predicted: 64
Actual: 12      Predicted: 36
Actual: 12      Predicted: 98
Actual: 12      Predicted: 83
Actual: 12      Predicted: 31
Actual: 12      Predicted: 16
Actual: 12      Predicted: 11
Actual: 12      Predicted: 58
Actual: 120     Predicted: 127
Actual: 120     Predicted: 127
Actual: 120     Predicted: 76
Actual: 120     Predicted: 61
Actual: 52      Predicted: 133
Actual: 52      Predicted: 43
Actual: 52      Predicted: 17
Actual: 52      Predicted: 76
Actual: 52      Predicted: 100
Actual: 52      Predicted: 31
Actual: 44      Predicted: 103
Actual: 44      Predicted: 55
Actual: 44      Predicted: 27
Actual: 44      Predicted: 55
Actual: 44      Predicted: 33
Actual: 44      Predicted: 25
Actual: 44      Predicted: 55
Actual: 44      Predicted: 73
```

```
Actual: 63      Predicted: 11
Actual: 63      Predicted: 37
Actual: 63      Predicted: 48
Actual: 63      Predicted: 15
Actual: 63      Predicted: 58
Actual: 63      Predicted: 12
Actual: 80      Predicted: 71
Actual: 80      Predicted: 56
Actual: 80      Predicted: 65
Actual: 80      Predicted: 65
Actual: 6       Predicted: 8
Actual: 6       Predicted: 124
Actual: 6       Predicted: 38
Actual: 6       Predicted: 5
Actual: 6       Predicted: 116
Actual: 6       Predicted: 117
Actual: 38      Predicted: 10
Actual: 38      Predicted: 13
Actual: 38      Predicted: 117
Actual: 38      Predicted: 39
Actual: 38      Predicted: 117
Actual: 48      Predicted: 103
Actual: 48      Predicted: 40
Actual: 48      Predicted: 50
Actual: 48      Predicted: 75
Actual: 48      Predicted: 128
Actual: 48      Predicted: 47
Actual: 48      Predicted: 34
Actual: 66      Predicted: 9
Actual: 66      Predicted: 94
Actual: 66      Predicted: 44
Actual: 66      Predicted: 88
Actual: 50      Predicted: 38
Actual: 50      Predicted: 72
Actual: 50      Predicted: 132
Actual: 50      Predicted: 37
Actual: 50      Predicted: 76
Actual: 56      Predicted: 73
Actual: 56      Predicted: 86
Actual: 56      Predicted: 27
Actual: 56      Predicted: 72
Actual: 56      Predicted: 71
Actual: 56      Predicted: 77
Actual: 56      Predicted: 112
Actual: 97      Predicted: 1
Actual: 97      Predicted: 53
Actual: 97      Predicted: 3
Actual: 97      Predicted: 105
```

```
Actual: 97      Predicted: 53
Actual: 55      Predicted: 21
Actual: 55      Predicted: 68
Actual: 55      Predicted: 5
Actual: 55      Predicted: 32
Actual: 55      Predicted: 68
Actual: 55      Predicted: 44
Actual: 55      Predicted: 60
Actual: 47      Predicted: 38
Actual: 47      Predicted: 87
Actual: 47      Predicted: 117
Actual: 47      Predicted: 38
Actual: 47      Predicted: 9
Actual: 47      Predicted: 46
Actual: 114     Predicted: 97
Actual: 114     Predicted: 13
Actual: 114     Predicted: 7
Actual: 114     Predicted: 89
Actual: 73      Predicted: 17
Actual: 73      Predicted: 98
Actual: 73      Predicted: 40
Actual: 73      Predicted: 107
Actual: 73      Predicted: 72
Actual: 5       Predicted: 43
Actual: 5       Predicted: 9
Actual: 5       Predicted: 11
Actual: 5       Predicted: 57
Actual: 5       Predicted: 72
Actual: 5       Predicted: 57
Actual: 5       Predicted: 40
Actual: 5       Predicted: 62
Actual: 5       Predicted: 62
Actual: 15      Predicted: 16
Actual: 15      Predicted: 9
Actual: 15      Predicted: 14
Actual: 15      Predicted: 16
Actual: 15      Predicted: 130
Actual: 15      Predicted: 44
Actual: 15      Predicted: 86
Actual: 106     Predicted: 73
Actual: 106     Predicted: 103
Actual: 106     Predicted: 25
Actual: 106     Predicted: 92
Actual: 106     Predicted: 105
Actual: 106     Predicted: 38
Actual: 130     Predicted: 116
Actual: 130     Predicted: 13
Actual: 130     Predicted: 15
```

```
Actual: 130     Predicted: 37
Actual: 31      Predicted: 24
Actual: 31      Predicted: 36
Actual: 31      Predicted: 43
Actual: 31      Predicted: 12
Actual: 31      Predicted: 40
Actual: 31      Predicted: 34
Actual: 31      Predicted: 39
Actual: 14      Predicted: 115
Actual: 14      Predicted: 37
Actual: 14      Predicted: 16
Actual: 14      Predicted: 47
Actual: 14      Predicted: 65
Actual: 14      Predicted: 76
Actual: 14      Predicted: 48
Actual: 14      Predicted: 57
Actual: 128     Predicted: 29
Actual: 128     Predicted: 65
Actual: 128     Predicted: 72
Actual: 128     Predicted: 87
Actual: 22      Predicted: 80
Actual: 22      Predicted: 88
Actual: 22      Predicted: 42
Actual: 22      Predicted: 98
Actual: 22      Predicted: 27
Actual: 108     Predicted: 72
Actual: 108     Predicted: 103
Actual: 108     Predicted: 109
Actual: 30      Predicted: 87
Actual: 30      Predicted: 76
Actual: 30      Predicted: 103
Actual: 16      Predicted: 46
Actual: 16      Predicted: 14
Actual: 16      Predicted: 35
Actual: 16      Predicted: 53
Actual: 16      Predicted: 15
Actual: 16      Predicted: 83
Actual: 16      Predicted: 88
Actual: 16      Predicted: 63
Actual: 33      Predicted: 61
Actual: 33      Predicted: 1
Actual: 33      Predicted: 77
Actual: 33      Predicted: 35
Actual: 33      Predicted: 1
Actual: 4       Predicted: 110
Actual: 4       Predicted: 10
Actual: 4       Predicted: 127
Actual: 4       Predicted: 24
```

```
Actual: 4       Predicted: 24
Actual: 4       Predicted: 28
Actual: 9       Predicted: 33
Actual: 9       Predicted: 1
Actual: 9       Predicted: 73
Actual: 9       Predicted: 21
Actual: 72      Predicted: 44
Actual: 72      Predicted: 18
Actual: 72      Predicted: 109
Actual: 72      Predicted: 57
Actual: 72      Predicted: 98
Actual: 72      Predicted: 63
Actual: 39      Predicted: 43
Actual: 39      Predicted: 91
Actual: 39      Predicted: 5
Actual: 39      Predicted: 113
Actual: 39      Predicted: 70
Actual: 39      Predicted: 24
Actual: 39      Predicted: 100
Actual: 39      Predicted: 97
Actual: 117     Predicted: 24
Actual: 117     Predicted: 27
Actual: 117     Predicted: 127
Actual: 117     Predicted: 82
Actual: 117     Predicted: 98
Actual: 117     Predicted: 38
Actual: 25      Predicted: 56
Actual: 25      Predicted: 98
Actual: 25      Predicted: 36
Actual: 25      Predicted: 74
Actual: 102     Predicted: 109
Actual: 102     Predicted: 44
Actual: 102     Predicted: 16
Actual: 64      Predicted: 72
Actual: 64      Predicted: 86
Actual: 64      Predicted: 7
Actual: 64      Predicted: 77
Actual: 64      Predicted: 27
Actual: 96      Predicted: 41
Actual: 96      Predicted: 2
Actual: 96      Predicted: 4
Actual: 96      Predicted: 107
Actual: 96      Predicted: 83
Actual: 20      Predicted: 72
Actual: 20      Predicted: 98
Actual: 20      Predicted: 30
Actual: 20      Predicted: 98
Actual: 20      Predicted: 98
```

```
Actual: 79      Predicted: 101
Actual: 79      Predicted: 42
Actual: 79      Predicted: 40
Actual: 79      Predicted: 5
Actual: 79      Predicted: 19
Actual: 79      Predicted: 43
Actual: 79      Predicted: 113
Actual: 79      Predicted: 24
Actual: 51      Predicted: 76
Actual: 51      Predicted: 1
Actual: 51      Predicted: 117
Actual: 51      Predicted: 71
Actual: 51      Predicted: 38
Actual: 113     Predicted: 72
Actual: 113     Predicted: 37
Actual: 113     Predicted: 38
Actual: 113     Predicted: 123
Actual: 113     Predicted: 24
Actual: 103     Predicted: 44
Actual: 103     Predicted: 72
Actual: 103     Predicted: 58
Actual: 103     Predicted: 47
Actual: 103     Predicted: 63
Actual: 103     Predicted: 109
Actual: 103     Predicted: 98
Actual: 85      Predicted: 63
Actual: 85      Predicted: 37
Actual: 85      Predicted: 130
Actual: 85      Predicted: 64
Actual: 129     Predicted: 51
Actual: 129     Predicted: 98
Actual: 129     Predicted: 105
Actual: 129     Predicted: 44
Actual: 129     Predicted: 60
Actual: 105     Predicted: 47
Actual: 105     Predicted: 73
Actual: 105     Predicted: 35
Actual: 131     Predicted: 127
Actual: 131     Predicted: 107
Actual: 131     Predicted: 88
Actual: 115     Predicted: 37
Actual: 115     Predicted: 76
Actual: 115     Predicted: 34
Actual: 115     Predicted: 86
Actual: 115     Predicted: 91
Actual: 115     Predicted: 42
Actual: 110     Predicted: 76
Actual: 110     Predicted: 15
```

```
Actual: 110     Predicted: 72
Actual: 57      Predicted: 49
Actual: 57      Predicted: 44
Actual: 57      Predicted: 31
Actual: 57      Predicted: 28
Actual: 57      Predicted: 109
Actual: 98      Predicted: 109
Actual: 98      Predicted: 22
Actual: 98      Predicted: 5
Actual: 98      Predicted: 38
Actual: 98      Predicted: 27
Actual: 46      Predicted: 29
Actual: 46      Predicted: 23
Actual: 46      Predicted: 23
Actual: 46      Predicted: 16
Actual: 46      Predicted: 34
Actual: 46      Predicted: 32
Actual: 46      Predicted: 13
Actual: 46      Predicted: 85
Actual: 46      Predicted: 77
Actual: 60      Predicted: 112
Actual: 60      Predicted: 47
Actual: 60      Predicted: 51
Actual: 60      Predicted: 47
Actual: 60      Predicted: 86
Actual: 60      Predicted: 47
Actual: 59      Predicted: 68
Actual: 59      Predicted: 21
Actual: 59      Predicted: 65
Actual: 59      Predicted: 56
Actual: 59      Predicted: 21
Actual: 121     Predicted: 15
Actual: 121     Predicted: 115
Actual: 121     Predicted: 10
Actual: 32      Predicted: 16
Actual: 32      Predicted: 56
Actual: 32      Predicted: 29
Actual: 32      Predicted: 5
Actual: 32      Predicted: 109
Actual: 32      Predicted: 95
Actual: 53      Predicted: 46
Actual: 53      Predicted: 113
Actual: 53      Predicted: 107
Actual: 53      Predicted: 88
Actual: 53      Predicted: 72
Actual: 40      Predicted: 119
Actual: 40      Predicted: 80
Actual: 40      Predicted: 7
```

```
Actual: 40      Predicted: 5
Actual: 40      Predicted: 108
Actual: 62      Predicted: 94
Actual: 62      Predicted: 32
Actual: 62      Predicted: 32
Actual: 62      Predicted: 58
Actual: 62      Predicted: 81
Actual: 62      Predicted: 70
Actual: 111     Predicted: 103
Actual: 111     Predicted: 51
Actual: 111     Predicted: 118
Actual: 111     Predicted: 42
Actual: 111     Predicted: 51
Actual: 126     Predicted: 15
Actual: 126     Predicted: 130
Actual: 126     Predicted: 110
Actual: 81      Predicted: 80
Actual: 81      Predicted: 57
Actual: 81      Predicted: 86
Actual: 81      Predicted: 21
Actual: 81      Predicted: 72
Actual: 81      Predicted: 98
Actual: 81      Predicted: 78
Actual: 41      Predicted: 71
Actual: 41      Predicted: 76
Actual: 41      Predicted: 38
Actual: 41      Predicted: 42
Actual: 41      Predicted: 103
Actual: 41      Predicted: 117
Actual: 41      Predicted: 88
Actual: 41      Predicted: 71
Actual: 41      Predicted: 13
Actual: 93      Predicted: 9
Actual: 93      Predicted: 1
Actual: 93      Predicted: 44
Actual: 93      Predicted: 44
Actual: 68      Predicted: 74
Actual: 68      Predicted: 44
Actual: 68      Predicted: 1
Actual: 68      Predicted: 106
Actual: 68      Predicted: 74
Actual: 99      Predicted: 29
Actual: 99      Predicted: 57
Actual: 99      Predicted: 130
Actual: 99      Predicted: 37
Actual: 11      Predicted: 83
Actual: 11      Predicted: 95
Actual: 11      Predicted: 5
```

```
Actual: 11      Predicted: 28
Actual: 11      Predicted: 72
Actual: 11      Predicted: 89
Actual: 11      Predicted: 89
Actual: 11      Predicted: 22
Actual: 11      Predicted: 4
Actual: 94      Predicted: 75
Actual: 94      Predicted: 127
Actual: 94      Predicted: 31
Actual: 94      Predicted: 75
Actual: 83      Predicted: 62
Actual: 83      Predicted: 115
Actual: 83      Predicted: 16
Actual: 83      Predicted: 7
Actual: 83      Predicted: 39
Actual: 29      Predicted: 5
Actual: 29      Predicted: 104
Actual: 29      Predicted: 115
Actual: 29      Predicted: 28
Actual: 29      Predicted: 5
Actual: 29      Predicted: 14
Actual: 13      Predicted: 76
Actual: 13      Predicted: 41
Actual: 13      Predicted: 38
Actual: 13      Predicted: 38
Actual: 13      Predicted: 3
Actual: 13      Predicted: 56
Actual: 17      Predicted: 115
Actual: 17      Predicted: 126
Actual: 17      Predicted: 68
Actual: 17      Predicted: 119
Actual: 17      Predicted: 113
Actual: 17      Predicted: 58
Actual: 17      Predicted: 82
Actual: 17      Predicted: 80
Actual: 45      Predicted: 48
Actual: 45      Predicted: 37
Actual: 45      Predicted: 7
Actual: 45      Predicted: 39
Actual: 45      Predicted: 89
Actual: 45      Predicted: 15
Actual: 45      Predicted: 14
Actual: 107     Predicted: 86
Actual: 107     Predicted: 38
Actual: 107     Predicted: 86
Actual: 107     Predicted: 117
Actual: 107     Predicted: 50
Actual: 107     Predicted: 86
```

```
Actual: 18      Predicted: 55
Actual: 18      Predicted: 9
Actual: 18      Predicted: 59
Actual: 18      Predicted: 23
Actual: 71      Predicted: 76
Actual: 71      Predicted: 5
Actual: 71      Predicted: 13
Actual: 71      Predicted: 20
Actual: 71      Predicted: 87
Actual: 71      Predicted: 72
Actual: 71      Predicted: 103
Actual: 71      Predicted: 98
Actual: 37      Predicted: 63
Actual: 37      Predicted: 130
Actual: 37      Predicted: 46
Actual: 37      Predicted: 42
Actual: 91      Predicted: 52
Actual: 91      Predicted: 115
Actual: 91      Predicted: 9
Actual: 91      Predicted: 31
Actual: 1       Predicted: 98
Actual: 1       Predicted: 30
Actual: 1       Predicted: 22
Actual: 1       Predicted: 44
Actual: 1       Predicted: 73
Actual: 1       Predicted: 53
Actual: 10      Predicted: 60
Actual: 10      Predicted: 32
Actual: 10      Predicted: 42
Actual: 10      Predicted: 50
Actual: 10      Predicted: 76
Actual: 10      Predicted: 32
Actual: 127     Predicted: 89
Actual: 127     Predicted: 13
Actual: 127     Predicted: 19
Actual: 127     Predicted: 56
Actual: 127     Predicted: 133
Actual: 104     Predicted: 32
Actual: 104     Predicted: 97
Actual: 104     Predicted: 55
Actual: 104     Predicted: 9
Actual: 104     Predicted: 99
Actual: 119     Predicted: 41
Actual: 119     Predicted: 40
Actual: 119     Predicted: 92
Actual: 118     Predicted: 115
Actual: 118     Predicted: 16
Actual: 118     Predicted: 76
```

```
Actual: 118     Predicted: 92
Actual: 118     Predicted: 80
Actual: 118     Predicted: 76
Actual: 88      Predicted: 35
Actual: 88      Predicted: 42
Actual: 88      Predicted: 35
Actual: 88      Predicted: 109
Actual: 88      Predicted: 68
Actual: 42      Predicted: 76
Actual: 42      Predicted: 5
Actual: 42      Predicted: 118
Actual: 42      Predicted: 22
Actual: 42      Predicted: 94
Actual: 42      Predicted: 99
Actual: 42      Predicted: 103
Actual: 23      Predicted: 29
Actual: 23      Predicted: 106
Actual: 23      Predicted: 32
Actual: 23      Predicted: 80
Actual: 23      Predicted: 80
Actual: 23      Predicted: 53
Actual: 65      Predicted: 5
Actual: 65      Predicted: 23
Actual: 65      Predicted: 51
Actual: 65      Predicted: 23
Actual: 65      Predicted: 74
Actual: 28      Predicted: 11
Actual: 28      Predicted: 18
Actual: 28      Predicted: 43
Actual: 28      Predicted: 68
Actual: 67      Predicted: 127
Actual: 67      Predicted: 130
Actual: 67      Predicted: 36
Actual: 67      Predicted: 112
Actual: 3       Predicted: 87
Actual: 3       Predicted: 8
Actual: 3       Predicted: 112
Actual: 3       Predicted: 52
Actual: 34      Predicted: 91
Actual: 34      Predicted: 59
Actual: 34      Predicted: 98
Actual: 34      Predicted: 126
Actual: 34      Predicted: 72
Actual: 34      Predicted: 128
Actual: 34      Predicted: 72
Actual: 19      Predicted: 89
Actual: 19      Predicted: 46
Actual: 19      Predicted: 79
```

```
Actual: 19      Predicted: 94
Actual: 19      Predicted: 5
Actual: 19      Predicted: 99
Actual: 101     Predicted: 40
Actual: 101     Predicted: 58
Actual: 101     Predicted: 58
Actual: 101     Predicted: 94
Actual: 101     Predicted: 43
Actual: 133     Predicted: 81
Actual: 133     Predicted: 47
Actual: 133     Predicted: 58
Actual: 133     Predicted: 61
Actual: 74      Predicted: 9
Actual: 74      Predicted: 42
Actual: 74      Predicted: 93
Actual: 74      Predicted: 68
Actual: 74      Predicted: 17
Actual: 27      Predicted: 86
Actual: 27      Predicted: 3
Actual: 27      Predicted: 86
Actual: 27      Predicted: 48
Actual: 27      Predicted: 5
Actual: 49      Predicted: 75
Actual: 49      Predicted: 56
Actual: 49      Predicted: 72
Actual: 49      Predicted: 24
Actual: 49      Predicted: 28
Actual: 2       Predicted: 20
Actual: 2       Predicted: 38
Actual: 2       Predicted: 71
Actual: 2       Predicted: 94
Actual: 2       Predicted: 103
Actual: 2       Predicted: 89
Actual: 2       Predicted: 57
Actual: 2       Predicted: 30
Actual: 132     Predicted: 81
Actual: 132     Predicted: 77
Actual: 132     Predicted: 68
Actual: 82      Predicted: 81
Actual: 82      Predicted: 49
Actual: 82      Predicted: 23
Actual: 82      Predicted: 94
Actual: 82      Predicted: 113
Actual: 82      Predicted: 94
Actual: 82      Predicted: 132
Actual: 82      Predicted: 72
Actual: 87      Predicted: 47
Actual: 87      Predicted: 117
```

```
Actual: 87      Predicted: 13
Actual: 87      Predicted: 13
Actual: 87      Predicted: 38
Actual: 87      Predicted: 38
Actual: 26      Predicted: 21
Actual: 26      Predicted: 74
Actual: 26      Predicted: 74
Actual: 26      Predicted: 28
Actual: 26      Predicted: 35
Actual: 123     Predicted: 48
Actual: 123     Predicted: 86
Actual: 123     Predicted: 67
Actual: 123     Predicted: 91
Actual: 8       Predicted: 40
Actual: 8       Predicted: 84
Actual: 8       Predicted: 44
Actual: 8       Predicted: 15
Actual: 8       Predicted: 41
Actual: 8       Predicted: 118
Actual: 8       Predicted: 89
Actual: 8       Predicted: 83
Actual: 70      Predicted: 27
Actual: 70      Predicted: 18
Actual: 70      Predicted: 59
Actual: 70      Predicted: 56
Actual: 70      Predicted: 68
Actual: 70      Predicted: 54
Actual: 125     Predicted: 74
Actual: 125     Predicted: 78
Actual: 125     Predicted: 44
Actual: 125     Predicted: 90
Actual: 35      Predicted: 58
Actual: 35      Predicted: 21
Actual: 35      Predicted: 86
Actual: 35      Predicted: 56
Actual: 54      Predicted: 14
Actual: 54      Predicted: 13
Actual: 54      Predicted: 123
Actual: 54      Predicted: 117
Actual: 54      Predicted: 14
Actual: 54      Predicted: 34
Actual: 76      Predicted: 112
Actual: 76      Predicted: 133
Actual: 76      Predicted: 94
Actual: 76      Predicted: 72
Actual: 76      Predicted: 98
Actual: 76      Predicted: 56
Actual: 76      Predicted: 42
```

```
Actual: 76      Predicted: 110
Actual: 89      Predicted: 33
Actual: 89      Predicted: 99
Actual: 89      Predicted: 49
Actual: 89      Predicted: 88
Actual: 89      Predicted: 109
Actual: 89      Predicted: 68
Actual: 89      Predicted: 133
Actual: 77      Predicted: 56
Actual: 77      Predicted: 98
Actual: 77      Predicted: 18
Actual: 77      Predicted: 15
Actual: 84      Predicted: 116
Actual: 84      Predicted: 117
Actual: 84      Predicted: 16
Actual: 84      Predicted: 103
Actual: 84      Predicted: 95
Actual: 109     Predicted: 32
Actual: 109     Predicted: 62
Actual: 109     Predicted: 76
Actual: 109     Predicted: 5
Actual: 58      Predicted: 22
Actual: 58      Predicted: 5
Actual: 58      Predicted: 42
Actual: 58      Predicted: 95
Actual: 58      Predicted: 127
Actual: 7       Predicted: 76
Actual: 7       Predicted: 16
Actual: 7       Predicted: 108
Actual: 7       Predicted: 99
Actual: 7       Predicted: 16
Actual: 7       Predicted: 37
Actual: 7       Predicted: 52
Actual: 75      Predicted: 39
Actual: 75      Predicted: 42
Actual: 75      Predicted: 62
Actual: 75      Predicted: 39
Actual: 75      Predicted: 114
Actual: 61      Predicted: 63
Actual: 61      Predicted: 73
Actual: 61      Predicted: 36
Actual: 61      Predicted: 1
Actual: 61      Predicted: 10
Actual: 61      Predicted: 22
Actual: 61      Predicted: 35
Actual: 61      Predicted: 53
Actual: 78      Predicted: 41
Actual: 78      Predicted: 81
```

```
Actual: 78      Predicted: 57
Actual: 78      Predicted: 53
Actual: 78      Predicted: 5
Actual: 86      Predicted: 88
Actual: 86      Predicted: 60
Actual: 86      Predicted: 73
Actual: 86      Predicted: 36
Actual: 86      Predicted: 88
Actual: 86      Predicted: 36
Actual: 86      Predicted: 120
Actual: 124     Predicted: 94
Actual: 124     Predicted: 76
Actual: 124     Predicted: 1
Actual: 124     Predicted: 102
Actual: 124     Predicted: 9
Actual: 124     Predicted: 94
Actual: 122     Predicted: 63
Actual: 122     Predicted: 55
Actual: 122     Predicted: 49
Actual: 122     Predicted: 59
Actual: 43      Predicted: 76
Actual: 43      Predicted: 20
Actual: 43      Predicted: 5
Actual: 43      Predicted: 47
Actual: 43      Predicted: 29
Actual: 43      Predicted: 31
Actual: 24      Predicted: 123
Actual: 24      Predicted: 94
Actual: 24      Predicted: 91
Actual: 24      Predicted: 79
Actual: 24      Predicted: 101
Actual: 24      Predicted: 101
Actual: 69      Predicted: 109
Actual: 69      Predicted: 24
Actual: 69      Predicted: 5
Actual: 69      Predicted: 37
Actual: 69      Predicted: 19
Actual: 69      Predicted: 93
Actual: 112     Predicted: 89
Actual: 112     Predicted: 14
Actual: 112     Predicted: 61
Actual: 112     Predicted: 76
Actual: 112     Predicted: 42
Actual: 112     Predicted: 83
Actual: 112     Predicted: 60
Actual: 36      Predicted: 30
Actual: 36      Predicted: 51
Actual: 36      Predicted: 114
```

```
Actual: 36      Predicted: 131
Actual: 36      Predicted: 10
Actual: 36      Predicted: 57
ResNet50 Top-1 Error 11% for breeds of dogs
Correctly classified: 100 / 835
```

1.2 Pretrained Model (HW 5)

[150]:

```
# Freeze model weights
# You need to go throught all the parameters in model.parameters()
# You need to set "requires_grad" to "False" for all parameters
for param in dog_model.parameters():
  param.requires_grad = False

# You may get the number of the features from the feature layer of the␣
 ↪pretrained network
# You can use model.fc.in_features to get the feature number
# n_inputs = dog_model.fc.in_features

dog_model.fc = nn.Linear(n_inputs, n_classes, bias=True)

# = nn.Sequential(
#                       # Define the last block of the nework for our dataset.
#                       # This block may have two linear layers, one dropout␣
 ↪layer, and one softmax layer.
#                       # You may design your own classifier with a discription.
#                       nn.Linear(n_inputs, 256),
#                       nn.ReLU(),
#                       nn.Dropout(0.4),
#                       nn.Linear(256, n_classes),
#                       nn.LogSoftmax(dim=1))
dog_model.fc

total_params = sum(p.numel() for p in dog_model.parameters())
print(f'{total_params:,} total parameters.')
total_trainable_params = sum(
    p.numel() for p in dog_model.parameters() if p.requires_grad)
print(f'{total_trainable_params:,} training parameters.')
```

[150]:

```
Linear(in_features=2048, out_features=133, bias=True)

23,780,549 total parameters.
272,517 training parameters.
```

[151]:

```
# Check whether there is a gpu for cuda
train_on_gpu = cuda.is_available()
print(f'Train on gpu: {train_on_gpu}')
```

```
# Number of gpus
if train_on_gpu:
    gpu_count = cuda.device_count()
    print(f'{gpu_count} gpus detected.')
    if gpu_count > 1:
        multi_gpu = True
    else:
        multi_gpu = False
else:
    multi_gpu = False
print(train_on_gpu,multi_gpu)

if train_on_gpu:
    dog_model = dog_model.to('cuda')
```

```
Train on gpu: True
1 gpus detected.
True False
```

Set up hyper parameters for our network.

[152]:

```
dog_model.class_to_idx = data['train'].class_to_idx
dog_model.idx_to_class = {
    idx: class_
    for class_, idx in dog_model.class_to_idx.items()
}

list(dog_model.idx_to_class.items())


# Set up your criterion and optimizer
dog_criterion = nn.CrossEntropyLoss()
dog_optimizer = optim.Adam(dog_model.parameters(), lr = 0.001)

for p in dog_optimizer.param_groups[0]['params']:
    if p.requires_grad:
        print(p.shape)
```

[152]:

```
[(0, '001.Affenpinscher'),
 (1, '002.Afghan_hound'),
 (2, '003.Airedale_terrier'),
 (3, '004.Akita'),
 (4, '005.Alaskan_malamute'),
 (5, '006.American_eskimo_dog'),
 (6, '007.American_foxhound'),
 (7, '008.American_staffordshire_terrier'),
 (8, '009.American_water_spaniel'),
 (9, '010.Anatolian_shepherd_dog'),
 (10, '011.Australian_cattle_dog'),
 (11, '012.Australian_shepherd'),
 (12, '013.Australian_terrier'),
 (13, '014.Basenji'),
 (14, '015.Basset_hound'),
 (15, '016.Beagle'),
 (16, '017.Bearded_collie'),
 (17, '018.Beauceron'),
 (18, '019.Bedlington_terrier'),
 (19, '020.Belgian_malinois'),
 (20, '021.Belgian_sheepdog'),
 (21, '022.Belgian_tervuren'),
 (22, '023.Bernese_mountain_dog'),
 (23, '024.Bichon_frise'),
 (24, '025.Black_and_tan_coonhound'),
 (25, '026.Black_russian_terrier'),
 (26, '027.Bloodhound'),
 (27, '028.Bluetick_coonhound'),
 (28, '029.Border_collie'),
 (29, '030.Border_terrier'),
 (30, '031.Borzoi'),
 (31, '032.Boston_terrier'),
 (32, '033.Bouvier_des_flandres'),
 (33, '034.Boxer'),
 (34, '035.Boykin_spaniel'),
 (35, '036.Briard'),
 (36, '037.Brittany'),
 (37, '038.Brussels_griffon'),
 (38, '039.Bull_terrier'),
 (39, '040.Bulldog'),
 (40, '041.Bullmastiff'),
 (41, '042.Cairn_terrier'),
 (42, '043.Canaan_dog'),
 (43, '044.Cane_corso'),
 (44, '045.Cardigan_welsh_corgi'),
 (45, '046.Cavalier_king_charles_spaniel'),
 (46, '047.Chesapeake_bay_retriever'),
 (47, '048.Chihuahua'),
 (48, '049.Chinese_crested'),
 (49, '050.Chinese_shar-pei'),
 (50, '051.Chow_chow'),
 (51, '052.Clumber_spaniel'),
 (52, '053.Cocker_spaniel'),
 (53, '054.Collie'),
 (54, '055.Curly-coated_retriever'),
 (55, '056.Dachshund'),
 (56, '057.Dalmatian'),
 (57, '058.Dandie_dinmont_terrier'),
 (58, '059.Doberman_pinscher'),
 (59, '060.Dogue_de_bordeaux'),
 (60, '061.English_cocker_spaniel'),
 (61, '062.English_setter'),
 (62, '063.English_springer_spaniel'),
 (63, '064.English_toy_spaniel'),
 (64, '065.Entlebucher_mountain_dog'),
 (65, '066.Field_spaniel'),
 (66, '067.Finnish_spitz'),
 (67, '068.Flat-coated_retriever'),
 (68, '069.French_bulldog'),
 (69, '070.German_pinscher'),
 (70, '071.German_shepherd_dog'),
 (71, '072.German_shorthaired_pointer'),
 (72, '073.German_wirehaired_pointer'),
 (73, '074.Giant_schnauzer'),
 (74, '075.Glen_of_imaal_terrier'),
 (75, '076.Golden_retriever'),
 (76, '077.Gordon_setter'),
 (77, '078.Great_dane'),
 (78, '079.Great_pyrenees'),
 (79, '080.Greater_swiss_mountain_dog'),
 (80, '081.Greyhound'),
 (81, '082.Havanese'),
 (82, '083.Ibizan_hound'),
 (83, '084.Icelandic_sheepdog'),
 (84, '085.Irish_red_and_white_setter'),
 (85, '086.Irish_setter'),
 (86, '087.Irish_terrier'),
 (87, '088.Irish_water_spaniel'),
 (88, '089.Irish_wolfhound'),
 (89, '090.Italian_greyhound'),
 (90, '091.Japanese_chin'),
 (91, '092.Keeshond'),
 (92, '093.Kerry_blue_terrier'),
 (93, '094.Komondor'),
 (94, '095.Kuvasz'),
 (95, '096.Labrador_retriever'),
 (96, '097.Lakeland_terrier'),
 (97, '098.Leonberger'),
 (98, '099.Lhasa_apso'),
 (99, '100.Lowchen'),
 (100, '101.Maltese'),
 (101, '102.Manchester_terrier'),
 (102, '103.Mastiff'),
 (103, '104.Miniature_schnauzer'),
 (104, '105.Neapolitan_mastiff'),
 (105, '106.Newfoundland'),
 (106, '107.Norfolk_terrier'),
 (107, '108.Norwegian_buhund'),
 (108, '109.Norwegian_elkhound'),
 (109, '110.Norwegian_lundehund'),
 (110, '111.Norwich_terrier'),
 (111, '112.Nova_scotia_duck_tolling_retriever'),
 (112, '113.Old_english_sheepdog'),
 (113, '114.Otterhound'),
 (114, '115.Papillon'),
 (115, '116.Parson_russell_terrier'),
 (116, '117.Pekingese'),
 (117, '118.Pembroke_welsh_corgi'),
 (118, '119.Petit_basset_griffon_vendeen'),
 (119, '120.Pharaoh_hound'),
 (120, '121.Plott'),
 (121, '122.Pointer'),
 (122, '123.Pomeranian'),
 (123, '124.Poodle'),
 (124, '125.Portuguese_water_dog'),
 (125, '126.Saint_bernard'),
 (126, '127.Silky_terrier'),
 (127, '128.Smooth_fox_terrier'),
 (128, '129.Tibetan_mastiff'),
 (129, '130.Welsh_springer_spaniel'),
 (130, '131.Wirehaired_pointing_griffon'),
 (131, '132.Xoloitzcuintli'),
 (132, '133.Yorkshire_terrier')]
```

```
torch.Size([133, 2048])
torch.Size([133])
```

1.2.1 Training Process

Experiment 1: Early stop after 5 epochs, 500 epochs possible, lr = 1e-3

Experiment 2: Continue training after 46 epochs, early stop after 10 epochs, 500 epochs possible, lr = 1e-3

Experiment 3: Continue training after 77 epochs, early stop after 5 epochs, 500 epochs possible, lr = 1e-4

Experiment 4: Continue trainging after 86 epochs, lr = 1e-5

Experiment 5: Continue training after 99 epochs, lr = 1e-3, output layer only has one linear layer now

[153]:

```
from timeit import default_timer as timer
save_file_name = f'resnet-50_model_best_model.pt'
train_on_gpu = cuda.is_available()

model, history = train(dog_model,
    dog_criterion,
    dog_optimizer,
    dataloaders['train'],
    dataloaders['val'],
    save_file_name=save_file_name,
    max_epochs_stop=5,
    n_epochs=500,
    print_every=1)
```

```
Model has been trained for: 99 epochs.


/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:477:
UserWarning: This DataLoader will create 10 worker processes in total. Our
suggested max number of worker in current system is 4, which is smaller than
what this DataLoader is going to create. Please be aware that excessive worker
creation might get DataLoader running slow or even freeze, lower the worker
number to avoid potential slowness/freeze if necessary.
  cpuset_checked))


Epoch: 0        Training Loss: 3.1907   Validation Loss: 0.8021
                Training Accuracy: 30.12%        Validation Accuracy: 75.21%

Epoch: 1        Training Loss: 1.9865   Validation Loss: 0.6660
                Training Accuracy: 49.19%        Validation Accuracy: 79.51%

Epoch: 2        Training Loss: 1.8975   Validation Loss: 0.5720
                Training Accuracy: 51.90%        Validation Accuracy: 82.11%

Epoch: 3        Training Loss: 1.8235   Validation Loss: 0.5092
                Training Accuracy: 55.24%        Validation Accuracy: 83.97%

Epoch: 4        Training Loss: 1.8534   Validation Loss: 0.4722
                Training Accuracy: 56.14%        Validation Accuracy: 84.94%

Epoch: 5        Training Loss: 1.7519   Validation Loss: 0.4643
                Training Accuracy: 57.68%        Validation Accuracy: 85.69%

Epoch: 6        Training Loss: 1.7591   Validation Loss: 0.4544
                Training Accuracy: 58.61%        Validation Accuracy: 86.09%

Epoch: 7        Training Loss: 1.7512   Validation Loss: 0.4770
```

```
                Training Accuracy: 59.40%        Validation Accuracy: 85.78%

Epoch: 8        Training Loss: 1.7475   Validation Loss: 0.3884
                Training Accuracy: 58.97%        Validation Accuracy: 87.83%

Epoch: 9        Training Loss: 1.7022   Validation Loss: 0.4441
                Training Accuracy: 60.70%        Validation Accuracy: 87.17%

Epoch: 10       Training Loss: 1.7423   Validation Loss: 0.3456
                Training Accuracy: 59.72%        Validation Accuracy: 89.55%

Epoch: 11       Training Loss: 1.6979   Validation Loss: 0.4411
                Training Accuracy: 62.05%        Validation Accuracy: 87.25%

Epoch: 12       Training Loss: 1.6831   Validation Loss: 0.3608
                Training Accuracy: 62.47%        Validation Accuracy: 89.09%

Epoch: 13       Training Loss: 1.6337   Validation Loss: 0.3957
                Training Accuracy: 63.49%        Validation Accuracy: 88.13%

Epoch: 14       Training Loss: 1.6734   Validation Loss: 0.3545
                Training Accuracy: 62.81%        Validation Accuracy: 89.54%

Epoch: 15       Training Loss: 1.7096   Validation Loss: 0.2735
                Training Accuracy: 62.56%        Validation Accuracy: 91.65%

Epoch: 16       Training Loss: 1.6063   Validation Loss: 0.3743
                Training Accuracy: 63.97%        Validation Accuracy: 89.52%

Epoch: 17       Training Loss: 1.6796   Validation Loss: 0.3954
                Training Accuracy: 63.94%        Validation Accuracy: 88.68%

Epoch: 18       Training Loss: 1.6810   Validation Loss: 0.2993
                Training Accuracy: 63.19%        Validation Accuracy: 91.26%

Epoch: 19       Training Loss: 1.7098   Validation Loss: 0.2770
                Training Accuracy: 62.98%        Validation Accuracy: 91.23%

Epoch: 20       Training Loss: 1.6363   Validation Loss: 0.2739
                Training Accuracy: 64.13%        Validation Accuracy: 91.81%

Early Stopping! Total epochs: 20. Best epoch: 15 with loss: 0.27 and acc: 91.81%
1532.53 total seconds elapsed. 72.98 seconds per epoch.
```

1.2.2 Training and Validation Losses

[154]:
```
plt.figure(figsize=(8, 6))
for c in ['train_loss', 'valid_loss']:
    plt.plot(
        history[c], label=c)
plt.legend()
plt.xlabel('Epoch')
plt.ylabel('Average Negative Log Likelihood')
plt.title('Training and Validation Losses')
plt.show()
```

[154]:
```
<Figure size 576x432 with 0 Axes>
```

[154]:
```
[<matplotlib.lines.Line2D at 0x7fd5bec7f310>]
```

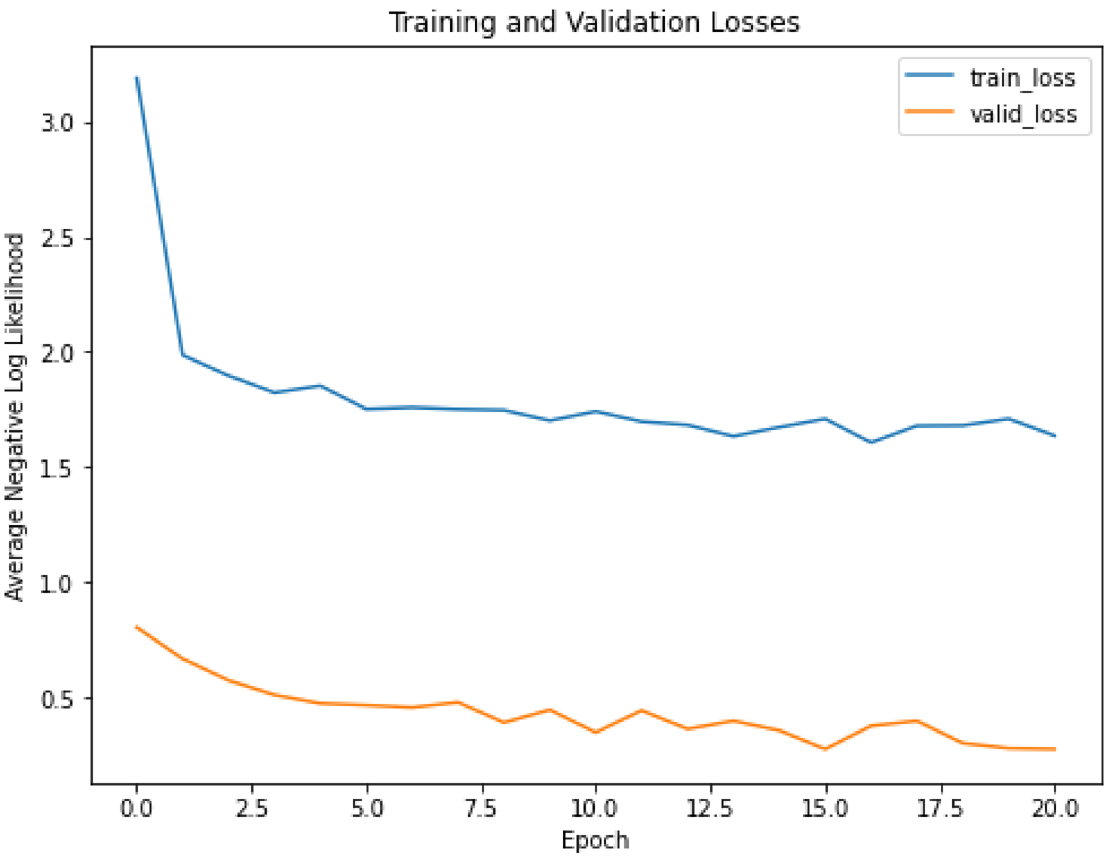
[154]:
```
[<matplotlib.lines.Line2D at 0x7fd5bec7f690>]
```

[154]:
```
<matplotlib.legend.Legend at 0x7fd5beca3590>
```

[154]:
```
Text(0.5, 0, 'Epoch')
```

[154]:
```
Text(0, 0.5, 'Average Negative Log Likelihood')
```

[154]:
```
Text(0.5, 1.0, 'Training and Validation Losses')
```

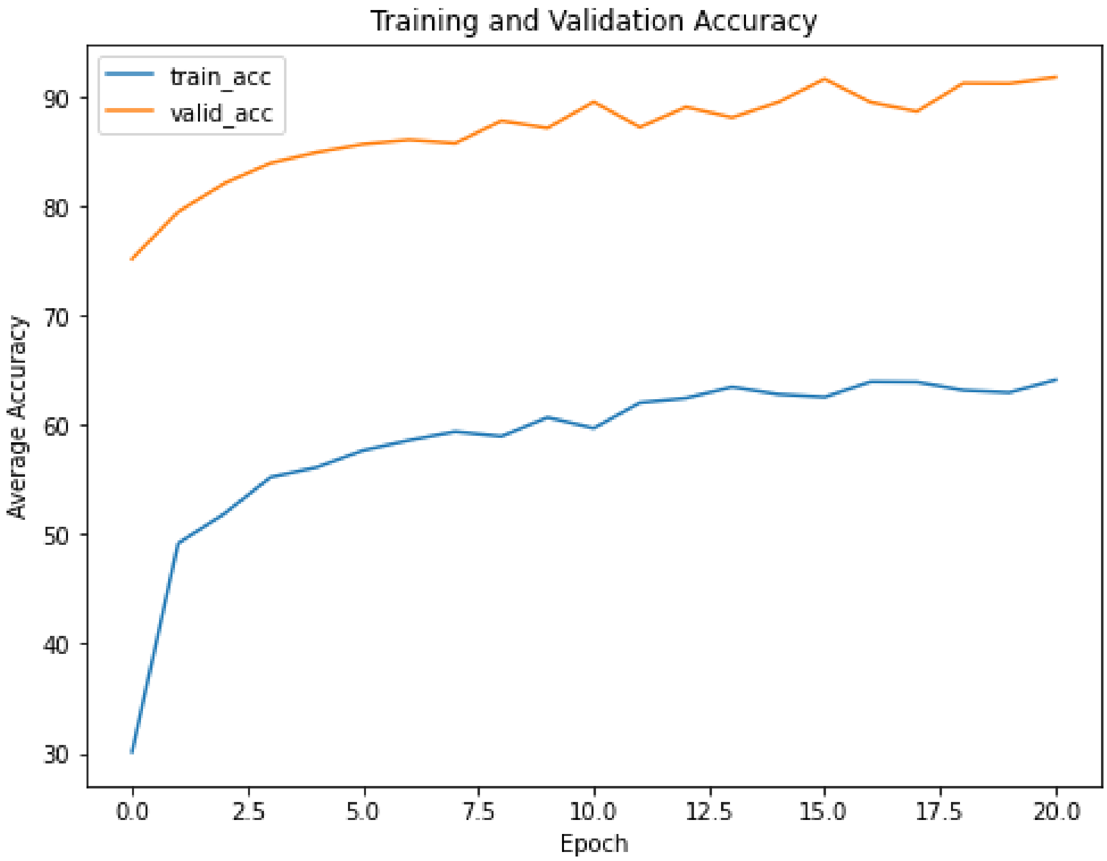


1.2.3 Training and Validation Accuracy

```
[155]: plt.figure(figsize=(8, 6))
       for c in ['train_acc', 'valid_acc']:
           plt.plot(
               100 * history[c], label=c)
       plt.legend()
       plt.xlabel('Epoch')
       plt.ylabel('Average Accuracy')
       plt.title('Training and Validation Accuracy')
       plt.show()
```

```
[155]: <Figure size 576x432 with 0 Axes>

[155]: [<matplotlib.lines.Line2D at 0x7fd5bec00250>]

[155]: [<matplotlib.lines.Line2D at 0x7fd5bec4bed0>]

[155]: <matplotlib.legend.Legend at 0x7fd5bec53a10>
```

```
[155]: Text(0.5, 0, 'Epoch')

[155]: Text(0, 0.5, 'Average Accuracy')

[155]: Text(0.5, 1.0, 'Training and Validation Accuracy')
```



```
[156]: test(dog_model, dog_criterion, dog_optimizer, dataloaders['test'], train_on_gpu)
```

```
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:477:
UserWarning: This DataLoader will create 10 worker processes in total. Our
suggested max number of worker in current system is 4, which is smaller than
what this DataLoader is going to create. Please be aware that excessive worker
creation might get DataLoader running slow or even freeze, lower the worker
number to avoid potential slowness/freeze if necessary.
  cpuset_checked))

Test Loss: 0.273531


Test Accuracy: 91% (6122/6680)
```

1.3 Predictions

```
[157]: print(breeds)
```

```
{90: 'Italian_greyhound', 95: 'Kuvasz', 116: 'Parson_russell_terrier', 92:
'Keeshond', 21: 'Belgian_sheepdog', 100: 'Lowchen', 12: 'Australian_shepherd',
120: 'Pharaoh_hound', 52: 'Clumber_spaniel', 44: 'Cane_corso', 63:
'English_springer_spaniel', 80: 'Greater_swiss_mountain_dog', 6:
'American_eskimo_dog', 38: 'Brussels_griffon', 48: 'Chihuahua', 66:
'Field_spaniel', 50: 'Chinese_shar-pei', 56: 'Dachshund', 97:
'Lakeland_terrier', 55: 'Curly-coated_retriever', 47:
'Chesapeake_bay_retriever', 114: 'Otterhound', 73: 'German_wirehaired_pointer',
5: 'Alaskan_malamute', 15: 'Basset_hound', 106: 'Newfoundland', 130:
'Welsh_springer_spaniel', 31: 'Borzoi', 14: 'Basenji', 128:
'Smooth_fox_terrier', 22: 'Belgian_tervuren', 108: 'Norwegian_buhund', 30:
'Border_terrier', 16: 'Beagle', 33: 'Bouvier_des_flandres', 4: 'Akita', 9:
'American_water_spaniel', 72: 'German_shorthaired_pointer', 39: 'Bull_terrier',
117: 'Pekingese', 25: 'Black_and_tan_coonhound', 102: 'Manchester_terrier', 64:
'English_toy_spaniel', 96: 'Labrador_retriever', 20: 'Belgian_malinois', 79:
'Great_pyrenees', 51: 'Chow_chow', 113: 'Old_english_sheepdog', 103: 'Mastiff',
85: 'Irish_red_and_white_setter', 129: 'Tibetan_mastiff', 105:
'Neapolitan_mastiff', 131: 'Wirehaired_pointing_griffon', 115: 'Papillon', 110:
'Norwegian_lundehund', 57: 'Dalmatian', 98: 'Leonberger', 46:
'Cavalier_king_charles_spaniel', 60: 'Dogue_de_bordeaux', 59:
'Doberman_pinscher', 121: 'Plott', 32: 'Boston_terrier', 53: 'Cocker_spaniel',
40: 'Bulldog', 62: 'English_setter', 111: 'Norwich_terrier', 126:
'Saint_bernard', 81: 'Greyhound', 41: 'Bullmastiff', 93: 'Kerry_blue_terrier',
68: 'Flat-coated_retriever', 99: 'Lhasa_apso', 11: 'Australian_cattle_dog', 94:
'Komondor', 83: 'Ibizan_hound', 29: 'Border_collie', 13: 'Australian_terrier',
17: 'Bearded_collie', 45: 'Cardigan_welsh_corgi', 107: 'Norfolk_terrier', 18:
'Beauceron', 71: 'German_shepherd_dog', 37: 'Brittany', 91: 'Japanese_chin', 1:
'Affenpinscher', 10: 'Anatolian_shepherd_dog', 127: 'Silky_terrier', 104:
'Miniature_schnauzer', 119: 'Petit_basset_griffon_vendeen', 118:
'Pembroke_welsh_corgi', 88: 'Irish_water_spaniel', 42: 'Cairn_terrier', 23:
'Bernese_mountain_dog', 65: 'Entlebucher_mountain_dog', 28:
'Bluetick_coonhound', 67: 'Finnish_spitz', 3: 'Airedale_terrier', 34: 'Boxer',
19: 'Bedlington_terrier', 101: 'Maltese', 133: 'Yorkshire_terrier', 74:
'Giant_schnauzer', 27: 'Bloodhound', 49: 'Chinese_crested', 2: 'Afghan_hound',
132: 'Xoloitzcuintli', 82: 'Havanese', 87: 'Irish_terrier', 26:
'Black_russian_terrier', 123: 'Pomeranian', 8: 'American_staffordshire_terrier',
70: 'German_pinscher', 125: 'Portuguese_water_dog', 35: 'Boykin_spaniel', 54:
'Collie', 76: 'Golden_retriever', 89: 'Irish_wolfhound', 77: 'Gordon_setter',
84: 'Icelandic_sheepdog', 109: 'Norwegian_elkhound', 58:
'Dandie_dinmont_terrier', 7: 'American_foxhound', 75: 'Glen_of_imaal_terrier',
61: 'English_cocker_spaniel', 78: 'Great_dane', 86: 'Irish_setter', 124:
'Poodle', 122: 'Pointer', 43: 'Canaan_dog', 24: 'Bichon_frise', 69:
'French_bulldog', 112: 'Nova_scotia_duck_tolling_retriever', 36: 'Briard'}
```

[167]:

```python
# Beagle -> wrong
filename = 'jane.jpg'
pred = wrong(filename)
print(breeds[pred.item()])
```

```
Golden_retriever
```

[168]:

```python
# Rottweiler Mutt -> NO rotties
archie = 'archie.jpg'
pred = wrong(archie)
print(breeds[pred.item()])
```

```
Australian_cattle_dog
```

[172]:

```python
# Pitbull -> NO rotties
filename = 'suzie.jpg'
pred = wrong(filename)
print(breeds[pred.item()])
```

```
Dandie_dinmont_terrier
```

[173]:

```python
# Labradoodle -> NO labradoodles
filename = 'beegee.jpeg'
pred = wrong(filename)
print(breeds[pred.item()])
```

```
Bichon_frise
```

[174]:

```python
# Me!
filename = 'alex.jpg'
pred = wrong(filename)
print(breeds[pred.item()])
```

```
German_shorthaired_pointer
```

[200]:

```python
# Major
filename = 'major.jpeg'
pred, percentage = wrong(filename)
print(breeds[pred.item()])
print(percentage)
```

```
Mastiff
tensor([ 0.2424,  0.4152,  0.0519,  0.5618,  6.4545,  0.0384,  0.7119,  0.4037,
         0.2191,  2.0431,  1.1301,  0.2628,  0.1071,  0.2780,  0.7466,  0.1093,
         0.1469,  0.0252,  0.2882,  0.4263,  0.1210,  0.7633,  0.0836,  0.0388,
         0.1812,  0.3160,  0.0579,  2.3230,  0.1487,  0.4777,  1.4030,  0.2582,
         1.1718,  0.2077,  0.2576,  0.4065,  0.0860,  0.5973,  0.4302,  1.3915,
         1.5195,  2.0272,  1.6886,  1.1793,  0.6563,  0.6875,  0.0993,  0.3675,
         1.2145,  0.2033,  0.2927,  0.2865,  1.3699,  1.5958,  0.3486,  0.2947,
         0.9267,  1.7507,  0.1254,  0.0967,  0.5374,  2.1240,  0.1517,  0.6974,
         0.9189,  0.1095,  0.1158,  0.3019,  1.3127,  0.0685,  0.0622,  1.3643,
         0.8427,  0.0849,  1.2512,  0.6288,  0.3742,  1.5285,  0.1812,  0.4017,
         0.7981,  0.4452,  0.1029,  1.6401,  0.0962,  0.0130,  0.9429,  0.2117,
         1.4182,  0.6666,  0.0581,  0.4530,  0.1081,  2.4805,  1.1855,  0.8397,
         1.0737,  0.1807,  0.7747,  0.5806,  0.3116,  0.3121, 10.3804,  1.3091,
         0.7561,  0.6192,  0.5035,  0.5863,  0.6939,  0.5782,  0.4027,  0.4352,
         0.6534,  0.9175,  0.1320,  0.7220,  0.7147,  0.4054,  0.4567,  0.0425,
         0.9643,  0.9852,  0.5867,  0.9137,  0.1594,  0.0955,  2.1420,  0.2702,
         0.3529,  0.5708,  2.4118,  0.2406,  0.1331], device='cuda:0',
       grad_fn=<MulBackward0>)
```

[176]:

```python
# Charlie
filename = 'charlie.jpeg'
pred = wrong(filename)
print(breeds[pred.item()])
```

```
Alaskan_malamute
```

[177]:

```python
# Charlie1
filename = 'charlie1.jpeg'
pred = wrong(filename)
print(breeds[pred.item()])
```

```
Welsh_springer_spaniel
```

[202]:

```python
# Luke
filename = 'luke.jpg'
pred, _ = wrong(filename)
print(breeds[pred.item()])
```

```
Gordon_setter
```