Building a Dog Breed Classifier

Udacity Machine Learning Engineer Nanodegree Capstone Report

ChihYu Yeh

I. Definition

Project Overview

Building a dog breed classifier is a computer vision task. Since the ImageNet challenge, image classification task performance using deep learning techniques becomes better and better and even outperforms human performance. Besides better performant neural network architectures coming out each year, one special technique called transfer learning really makes applying deep learning to real life much easier. With transfer learning, we can take one predefined architecture with pretrained weight, and fine-tune only last few layers of the model to quickly bring the new task performance to a quite accurate level. In this project, I will apply the latest research accomplishment called Big Transfer(BiT) released by Google, which is a large-scale pretrained model for computer vision tasks.

Problem Statement

This is a dog breed classification task. This is a multi-classification problem, since there are multiple dog breeds(133) out there. Besides classifying dog breeds, if a human is inside an input image, we will predict a dog breed that is the most similar to that human. Otherwise, the output will print out a message that indicates error. Since there are multiple stages in this problem, I'll solve the problem using a "pipeline" approach, that is to break a problem into multiple steps and each step solves a part of the problem.

Metrics

The metrics of the project is the prediction accuracy given the dog images. In this project, I pay more attention to the dog breed classifier, so the metrics is given to the dog breed classifier.(Assume only dog images are given to the dog breed classifier) The reason why I choose using prediction accuracy as the performance metric is because training data is not severely imbalanced. The definition of prediction accuracy is as the following:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

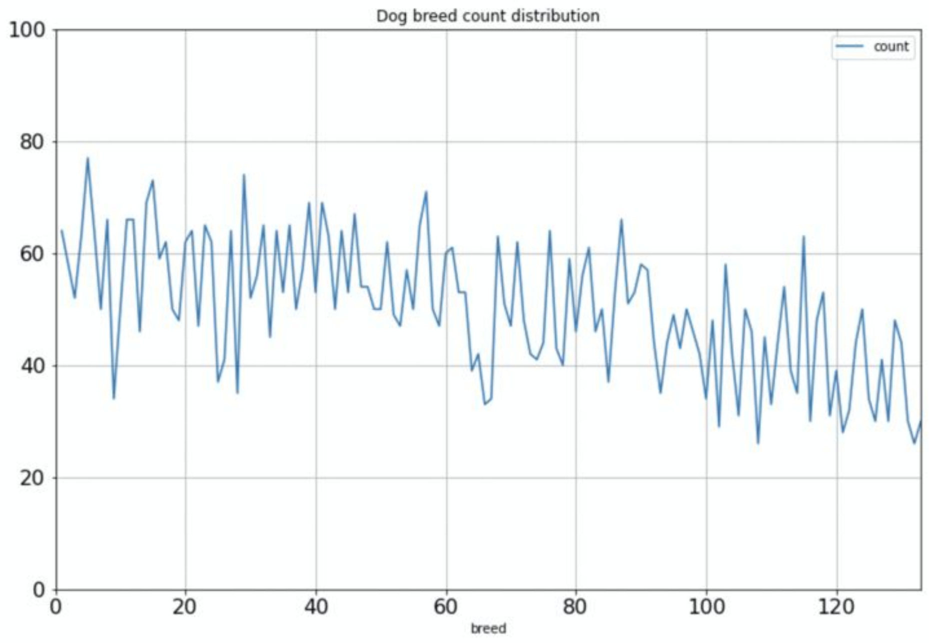
where TP = True Positives, TN = True Negatives, FP = False Positives, FN = False Negatives.

II. Analysis

Data Exploration

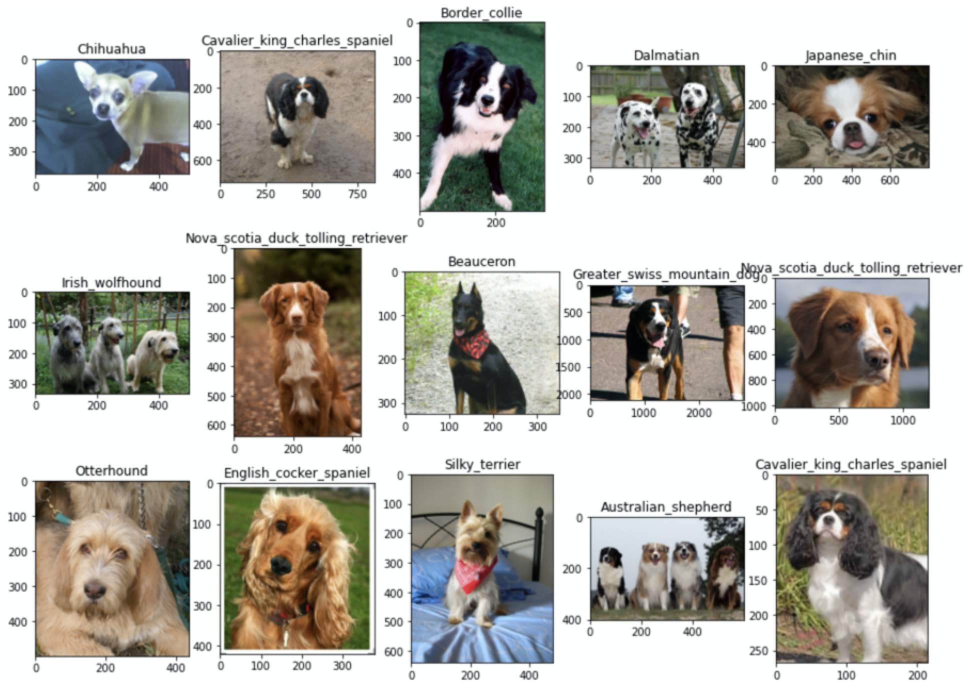
There are training/validation/testing dataset of dog images prepared for training a dog breed classifier. Since we shouldn't look at validation and testing data, so the data exploration step is focused on the training dataset. There are about 8000 to 9000 dog images with 133 breeds in the training dataset. The image below in the Exploratory Visualization section is the dog breed count distribution based on the training data.

Exploratory Visualization



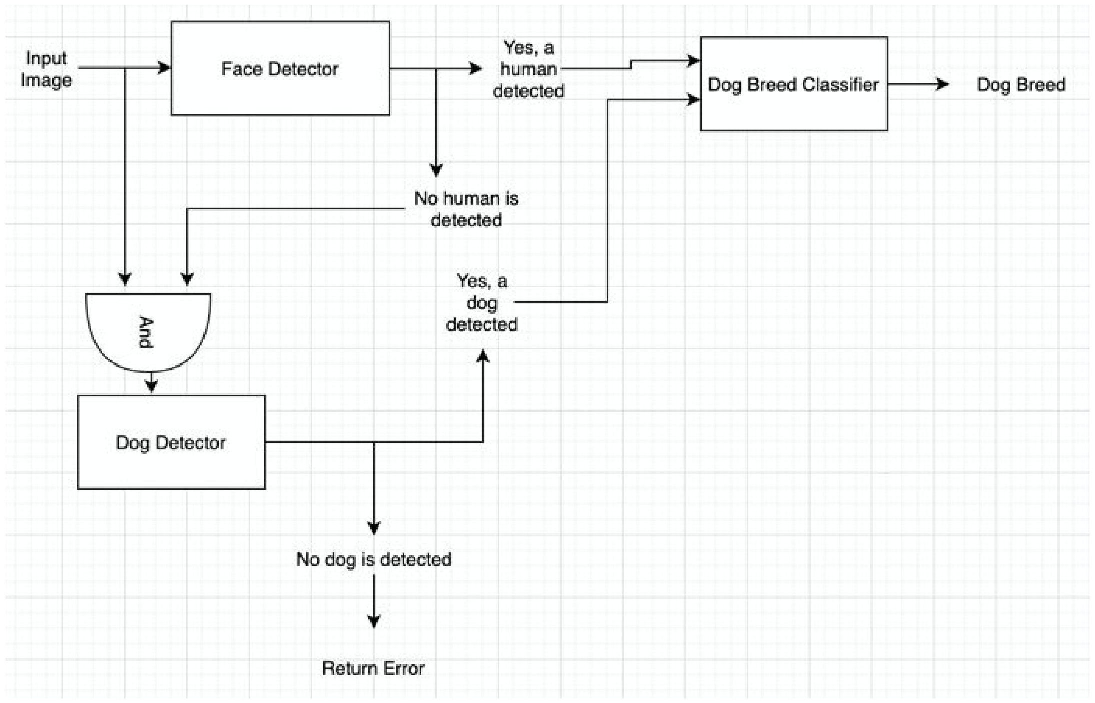
Mapping between breed index and breed name: please refer to "Create target index to target name mapping in the dog_app.ipynb"

Image sizes of training data are not the same; here is a sneak peek



Chihuahua
Cavalier_king_charles_spaniel
Border_collie
Dalmatian
Japanese_chin
Irish_wolfhound
Nova_scotia_duck_tolling_retriever
Beauceron
Greater_swiss_mountain_dog
Nova_scotia_duck_tolling_retriever
Otterhound
English_cocker_spaniel
Silky_terrier
Australian_shepherd
Cavalier_king_charles_spaniel

Algorithms and Techniques



The above picture is the overall system diagram of the project. First, an input image is fed into the face detector, which is written using an OpenCV pre-built algorithm. Then if the face detector detects there is a human face inside an image, the image will be fed to the dog breed classifier. If there is no face detected in the image, the image will be sent to the dog detector, which is written using a pretrained VGG 16 model. If no dog is detected, the system will return a message that indicates there is no dog in the image. So, the image will be fed to the dog breed classifier in both conditions whether a human or a dog is detected in the image. The dog breed classifier uses the pretrained model called Big Transfer with specific preprocessing steps given by ResNet.

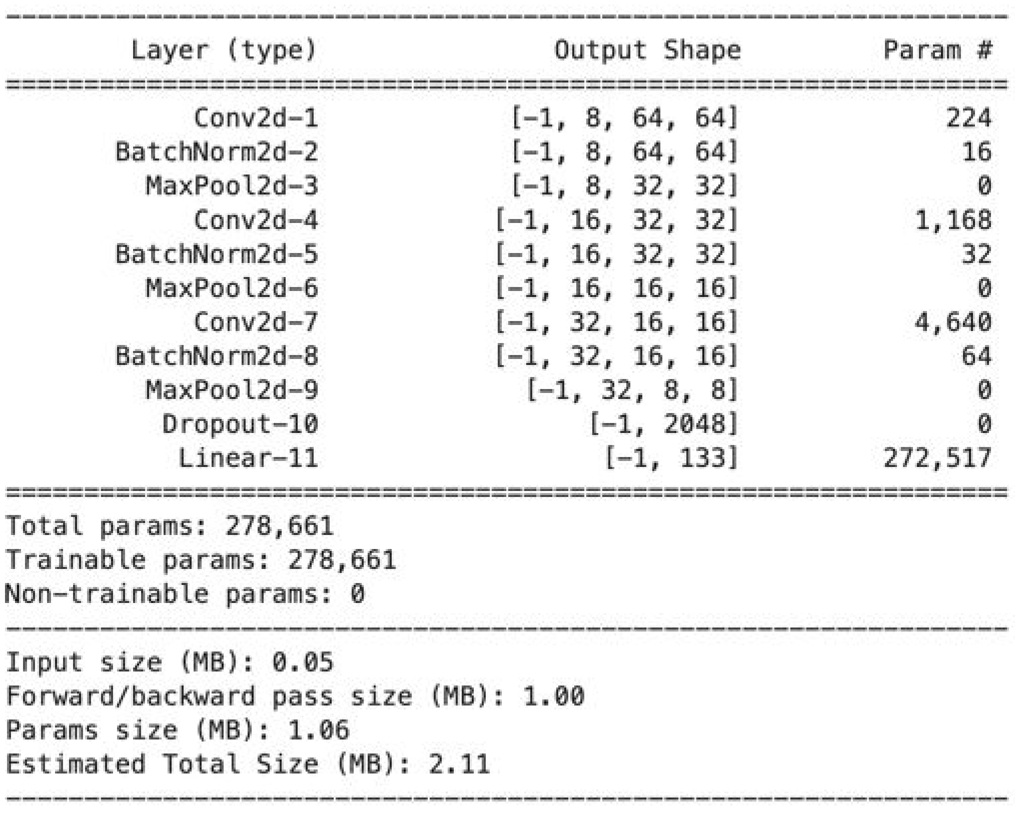
Big Transfer(BiT) Model Architecture

There are multiple pretrained model architectures released by Google. I finally chose BiT-M-R101x1, which is trained using ImageNet-21k. The reason I chose this architecture is to reach the most performant result given the constraint of the GPU memory(16GB). I trained the model using one Nvidia P100 GPU. The detailed architecture is mentioned in the reference. The model reaches 86% accuracy for classifying dog breeds.

Benchmark

The baseline model is very simple, and it looks similar to the classical LeNet architecture with several convolutional layers following pooling layers, then a fully connected layer at the top. The image below is the baseline model architecture and it reaches 12% accuracy for classifying dog breeds.

Model Architecture

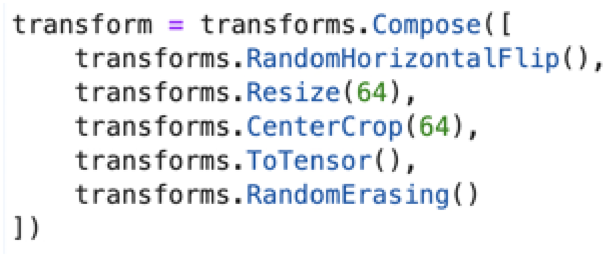


```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1            [-1, 8, 64, 64]             224
       BatchNorm2d-2            [-1, 8, 64, 64]              16
         MaxPool2d-3            [-1, 8, 32, 32]               0
            Conv2d-4           [-1, 16, 32, 32]           1,168
       BatchNorm2d-5           [-1, 16, 32, 32]              32
         MaxPool2d-6           [-1, 16, 16, 16]               0
            Conv2d-7           [-1, 32, 16, 16]           4,640
       BatchNorm2d-8           [-1, 32, 16, 16]              64
         MaxPool2d-9             [-1, 32, 8, 8]               0
          Dropout-10                 [-1, 2048]               0
           Linear-11                  [-1, 133]         272,517
================================================================
Total params: 278,661
Trainable params: 278,661
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 1.00
Params size (MB): 1.06
Estimated Total Size (MB): 2.11
----------------------------------------------------------------
```

III. Methodology

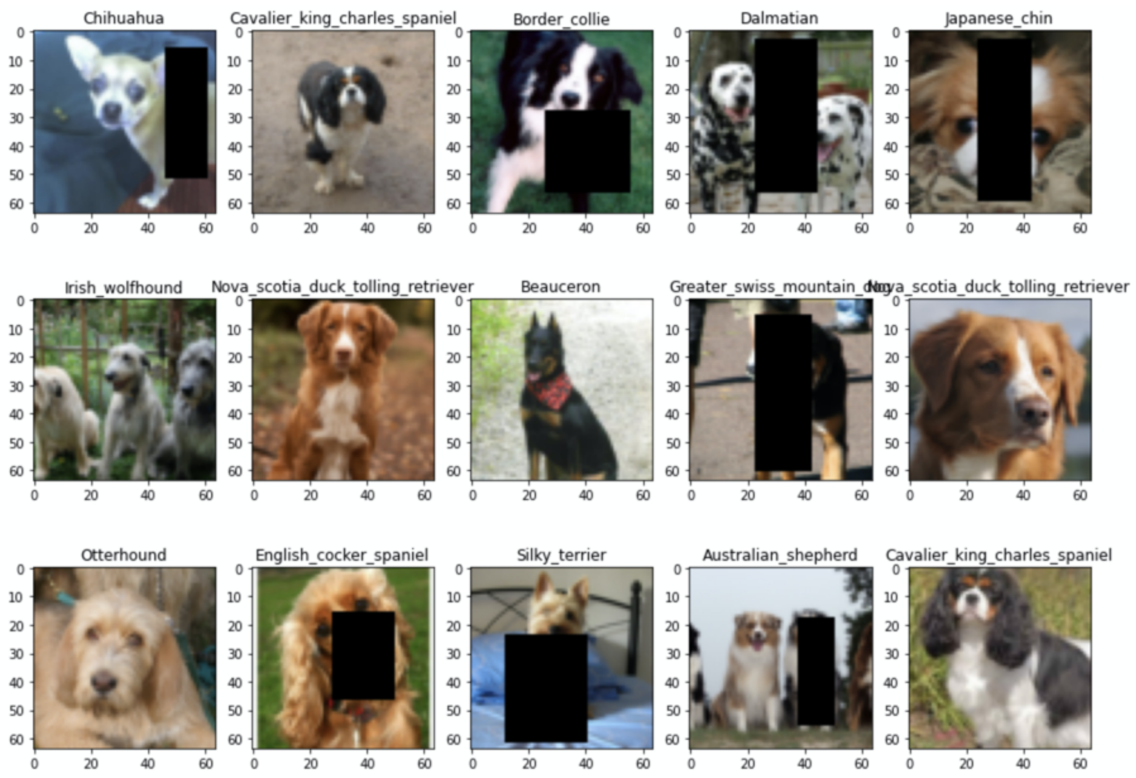
Data Preprocessing

Data preprocessing for the baseline model

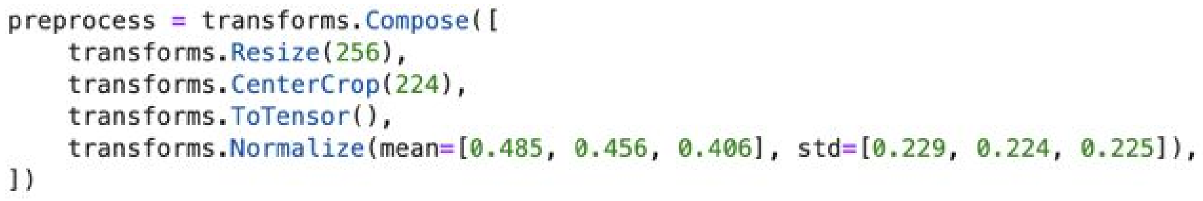


```
transform = transforms.Compose([
    transforms.RandomHorizontalFlip(),
    transforms.Resize(64),
    transforms.CenterCrop(64),
    transforms.ToTensor(),
    transforms.RandomErasing()
])
```

There are several steps in the preprocessing: random horizontal flips, resizing images to 64x64, center crop to 64x64, then the random erasing algorithm. The result of the preprocessing is as follows: (We can clearly see that the effect of random erasing algorithm is it would randomly selects a rectangle region in an image and erases its pixels with random values.)


Chihuahua
Cavalier_king_charles_spaniel
Border_collie
Dalmatian
Japanese_chin
Irish_wolfhound
Nova_scotia_duck_tolling_retriever
Beauceron
Greater_swiss_mountain_dog
Nova_scotia_duck_tolling_retriever
Otterhound
English_cocker_spaniel
Silky_terrier
Australian_shepherd
Cavalier_king_charles_spaniel

Data preprocessing for the proposed model



```
preprocess = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
])
```

Images are resized to 256x256 pixels, then center cropped to 224x224 pixels. Finally we normalize them with mean and standard deviation appeared above in the image. This preprocessing steps are according to the ResNet.

Implementation

- Data exploration
    - Check number of training data, validation data in order to make sure there is enough training data and the ratio among two datasets.
    - Draw target(dog breed) number distribution based on training data to check if they are severely unbalanced. This will affect how we design the performance metric or if we need to add more training data.
- Preprocessing
    - Make sure the image size would be the same for all input data to neural network models: use resizing techniques
    - Apply normalization to image data for neural network models: use normalization techniques
    - Try to apply the same preprosessing step as the pretrained model does
    - Try data augmentation if possible to enhance model stability(generalization ability)

- Training and testing the three models
    - Face detector(using one pretrained CV algorithm)
        - Input: image
        - Output: a boolean that indicates if there is a human face inside the image
    - Dog detector(using pretrained VGG-16 model): There is no fine-tuning here
        - Input: image
        - Output: a boolean that indicates if there is a dog inside the image
    - Dog breed classifier(using pretrained Big Transfer(BiT) model): I fine-tune the model here
        - Input: image
        - Output: one of 133 possible dog breeds
- Refinement
    - Apply transfer learning using a pretrained Big Transfer model, then fine-tune the model hyperparameters(batchsize, optimizer, learning rate, model architecture) to suit the project's need
    - I use the learning rate finder and one-cycle policy to determine the learning rate of the optimizer.
- Performance evaluation
    - Use dog breed classification accuracy as the performance metric since dog breeds are not severely imbalanced

Refinement

At first without fine-tuning hyperparameters of the pretrained model, the model reached only 68% accuracy, which I think it's still too low. I try hard and tune hyperparameters for a day to finally reach 86% accuracy. I tried batch sizes(16, 32, 64), different optimizer(SGD, Adam) with different learning rate, etc., different pretrained model architecture(ResNet-M-50x1, ResNet-S(/M)-50(/101)x1(/3)) Basically it's kind of like a grid search process.

Parameters used for the final result

Batch Size	Model Architecture	Optimizer	Learning Rate
16	ResNet-M-101x1	SGD with momentum 0.9	Using one-cycle policy with maximum 1e-3

IV. Results

Model Evaluation and Validation

	Proposed Model	Benchmark Model
Epochs spent	6	89
Training loss	0.056811	2.779065
Validation loss	0.372113	3.909540
Testing loss	0.412053	3.924075
Testing accuracy	86% (727/836)	12% (102/836)

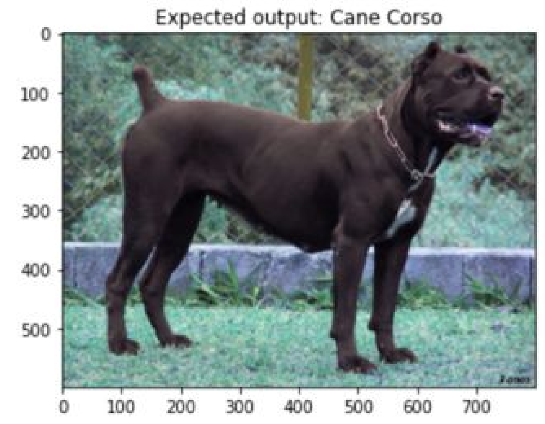
As we can see, the performance of the proposed model outperforms the benchmark model in every aspect. Especially we can see one important characteristic of transfer learning here is that the proposed model only spent 6 epochs to reach 86% accuracy. However, I found that it’s easy that the model would be overfitting, so I might need to add more training data or regularize the model more in order to mitigate the gap between training and validation loss.

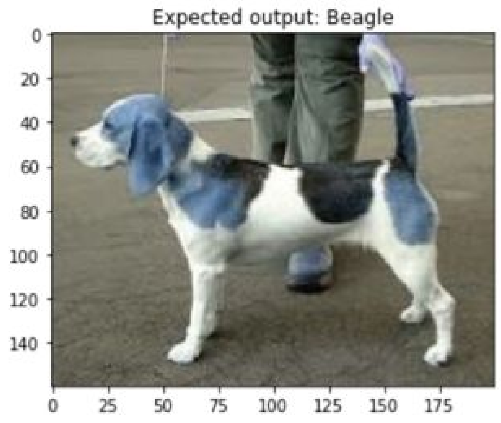
Model Justification

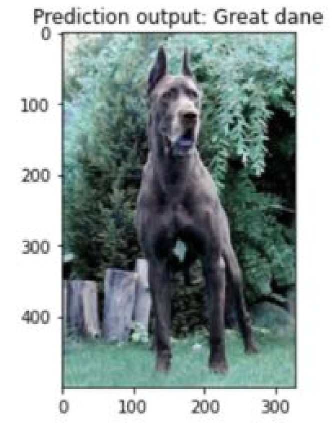
I randomly chose 8 dog images from the Internet, and got 75% prediction accuracy. It turned out that all dog breeds in 8 dog images are in the scope of the given 133 dog breeds. There are two dog breeds that are not predicted correct, they are shown as the following:

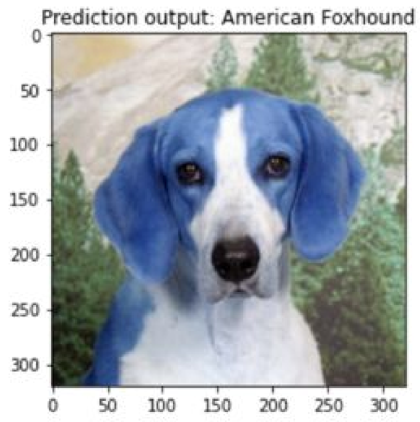
	Img 1	Img 2	Img 3	Img 4	Img 5	Img 6	Img 7	Img 8
Target	Bulldog	Labrador retriever	Cane corso	Beagle	Pointer	German shepherd dog	Dalmatian	Poodle
Prediction	Bulldog	Labrador retriever	Great dane	American fox hound	Pointer	German shepherd dog	Dalmatian	Poodle

Images that the proposed model predicted wrong results are shown below
It turned out that for dogs that really look similar to each other is hard for the model to predict right. I propose to give more dog images that are hard to be differentiated to the model in order to improve the model performance. Or maybe we can print out top-3 or top-5 prediction results for users to look further!









V. Reference

Big Transfer(BiT) Model Architecture(BiT-M-R101x1)

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
         StdConv2d-1         [-1, 64, 112, 112]           9,408
    ConstantPad2d-2         [-1, 64, 114, 114]               0
         MaxPool2d-3           [-1, 64, 56, 56]               0
         GroupNorm-4           [-1, 64, 56, 56]             128
              ReLU-5           [-1, 64, 56, 56]               0
         StdConv2d-6          [-1, 256, 56, 56]          16,384
         StdConv2d-7           [-1, 64, 56, 56]           4,096
         GroupNorm-8           [-1, 64, 56, 56]             128
              ReLU-9           [-1, 64, 56, 56]               0
        StdConv2d-10           [-1, 64, 56, 56]          36,864
        GroupNorm-11           [-1, 64, 56, 56]             128
             ReLU-12           [-1, 64, 56, 56]               0
        StdConv2d-13          [-1, 256, 56, 56]          16,384
PreActBottleneck-14          [-1, 256, 56, 56]               0
        GroupNorm-15          [-1, 256, 56, 56]             512
             ReLU-16          [-1, 256, 56, 56]               0
        StdConv2d-17           [-1, 64, 56, 56]          16,384
        GroupNorm-18           [-1, 64, 56, 56]             128
             ReLU-19           [-1, 64, 56, 56]               0
        StdConv2d-20           [-1, 64, 56, 56]          36,864
        GroupNorm-21           [-1, 64, 56, 56]             128
             ReLU-22           [-1, 64, 56, 56]               0
        StdConv2d-23          [-1, 256, 56, 56]          16,384
PreActBottleneck-24          [-1, 256, 56, 56]               0
        GroupNorm-25          [-1, 256, 56, 56]             512
             ReLU-26          [-1, 256, 56, 56]               0
        StdConv2d-27           [-1, 64, 56, 56]          16,384
        GroupNorm-28           [-1, 64, 56, 56]             128
             ReLU-29           [-1, 64, 56, 56]               0
        StdConv2d-30           [-1, 64, 56, 56]          36,864
        GroupNorm-31           [-1, 64, 56, 56]             128
             ReLU-32           [-1, 64, 56, 56]               0
        StdConv2d-33          [-1, 256, 56, 56]          16,384
PreActBottleneck-34          [-1, 256, 56, 56]               0
        GroupNorm-35          [-1, 256, 56, 56]             512
             ReLU-36          [-1, 256, 56, 56]               0
        StdConv2d-37          [-1, 512, 28, 28]         131,072
        StdConv2d-38          [-1, 128, 56, 56]          32,768
        GroupNorm-39          [-1, 128, 56, 56]             256
             ReLU-40          [-1, 128, 56, 56]               0
        StdConv2d-41          [-1, 128, 28, 28]         147,456
```

```
      GroupNorm-42          [-1, 128, 28, 28]             256
           ReLU-43          [-1, 128, 28, 28]               0
      StdConv2d-44          [-1, 512, 28, 28]          65,536
PreActBottleneck-45          [-1, 512, 28, 28]               0
      GroupNorm-46          [-1, 512, 28, 28]           1,024
           ReLU-47          [-1, 512, 28, 28]               0
      StdConv2d-48          [-1, 128, 28, 28]          65,536
      GroupNorm-49          [-1, 128, 28, 28]             256
           ReLU-50          [-1, 128, 28, 28]               0
      StdConv2d-51          [-1, 128, 28, 28]         147,456
      GroupNorm-52          [-1, 128, 28, 28]             256
           ReLU-53          [-1, 128, 28, 28]               0
      StdConv2d-54          [-1, 512, 28, 28]          65,536
PreActBottleneck-55          [-1, 512, 28, 28]               0
      GroupNorm-56          [-1, 512, 28, 28]           1,024
           ReLU-57          [-1, 512, 28, 28]               0
      StdConv2d-58          [-1, 128, 28, 28]          65,536
      GroupNorm-59          [-1, 128, 28, 28]             256
           ReLU-60          [-1, 128, 28, 28]               0
      StdConv2d-61          [-1, 128, 28, 28]         147,456
      GroupNorm-62          [-1, 128, 28, 28]             256
           ReLU-63          [-1, 128, 28, 28]               0
      StdConv2d-64          [-1, 512, 28, 28]          65,536
PreActBottleneck-65          [-1, 512, 28, 28]               0
      GroupNorm-66          [-1, 512, 28, 28]           1,024
           ReLU-67          [-1, 512, 28, 28]               0
      StdConv2d-68          [-1, 128, 28, 28]          65,536
      GroupNorm-69          [-1, 128, 28, 28]             256
           ReLU-70          [-1, 128, 28, 28]               0
      StdConv2d-71          [-1, 128, 28, 28]         147,456
      GroupNorm-72          [-1, 128, 28, 28]             256
           ReLU-73          [-1, 128, 28, 28]               0
      StdConv2d-74          [-1, 512, 28, 28]          65,536
PreActBottleneck-75          [-1, 512, 28, 28]               0
      GroupNorm-76          [-1, 512, 28, 28]           1,024
           ReLU-77          [-1, 512, 28, 28]               0
      StdConv2d-78         [-1, 1024, 14, 14]         524,288
      StdConv2d-79          [-1, 256, 28, 28]         131,072
      GroupNorm-80          [-1, 256, 28, 28]             512
           ReLU-81          [-1, 256, 28, 28]               0
      StdConv2d-82          [-1, 256, 14, 14]         589,824
      GroupNorm-83          [-1, 256, 14, 14]             512
           ReLU-84          [-1, 256, 14, 14]               0
      StdConv2d-85         [-1, 1024, 14, 14]         262,144
PreActBottleneck-86         [-1, 1024, 14, 14]               0
      GroupNorm-87         [-1, 1024, 14, 14]           2,048
           ReLU-88         [-1, 1024, 14, 14]               0
```

```
        StdConv2d-89         [-1, 256, 14, 14]         262,144
        GroupNorm-90         [-1, 256, 14, 14]             512
             ReLU-91         [-1, 256, 14, 14]               0
        StdConv2d-92         [-1, 256, 14, 14]         589,824
        GroupNorm-93         [-1, 256, 14, 14]             512
             ReLU-94         [-1, 256, 14, 14]               0
        StdConv2d-95        [-1, 1024, 14, 14]         262,144
 PreActBottleneck-96        [-1, 1024, 14, 14]               0
        GroupNorm-97        [-1, 1024, 14, 14]           2,048
             ReLU-98        [-1, 1024, 14, 14]               0
        StdConv2d-99         [-1, 256, 14, 14]         262,144
       GroupNorm-100         [-1, 256, 14, 14]             512
            ReLU-101         [-1, 256, 14, 14]               0
       StdConv2d-102         [-1, 256, 14, 14]         589,824
       GroupNorm-103         [-1, 256, 14, 14]             512
            ReLU-104         [-1, 256, 14, 14]               0
       StdConv2d-105        [-1, 1024, 14, 14]         262,144
PreActBottleneck-106        [-1, 1024, 14, 14]               0
       GroupNorm-107        [-1, 1024, 14, 14]           2,048
            ReLU-108        [-1, 1024, 14, 14]               0
       StdConv2d-109         [-1, 256, 14, 14]         262,144
       GroupNorm-110         [-1, 256, 14, 14]             512
            ReLU-111         [-1, 256, 14, 14]               0
       StdConv2d-112         [-1, 256, 14, 14]         589,824
       GroupNorm-113         [-1, 256, 14, 14]             512
            ReLU-114         [-1, 256, 14, 14]               0
       StdConv2d-115        [-1, 1024, 14, 14]         262,144
PreActBottleneck-116        [-1, 1024, 14, 14]               0
       GroupNorm-117        [-1, 1024, 14, 14]           2,048
            ReLU-118        [-1, 1024, 14, 14]               0
       StdConv2d-119         [-1, 256, 14, 14]         262,144
       GroupNorm-120         [-1, 256, 14, 14]             512
            ReLU-121         [-1, 256, 14, 14]               0
       StdConv2d-122         [-1, 256, 14, 14]         589,824
       GroupNorm-123         [-1, 256, 14, 14]             512
            ReLU-124         [-1, 256, 14, 14]               0
       StdConv2d-125        [-1, 1024, 14, 14]         262,144
PreActBottleneck-126        [-1, 1024, 14, 14]               0
       GroupNorm-127        [-1, 1024, 14, 14]           2,048
            ReLU-128        [-1, 1024, 14, 14]               0
       StdConv2d-129         [-1, 256, 14, 14]         262,144
       GroupNorm-130         [-1, 256, 14, 14]             512
            ReLU-131         [-1, 256, 14, 14]               0
       StdConv2d-132         [-1, 256, 14, 14]         589,824
       GroupNorm-133         [-1, 256, 14, 14]             512
            ReLU-134         [-1, 256, 14, 14]               0
       StdConv2d-135        [-1, 1024, 14, 14]         262,144
```

```
PreActBottleneck-136     [-1, 1024, 14, 14]     0
    GroupNorm-137     [-1, 1024, 14, 14]     2,048
        ReLU-138     [-1, 1024, 14, 14]     0
    StdConv2d-139     [-1, 256, 14, 14]     262,144
    GroupNorm-140     [-1, 256, 14, 14]     512
        ReLU-141     [-1, 256, 14, 14]     0
    StdConv2d-142     [-1, 256, 14, 14]     589,824
    GroupNorm-143     [-1, 256, 14, 14]     512
        ReLU-144     [-1, 256, 14, 14]     0
    StdConv2d-145     [-1, 1024, 14, 14]     262,144
PreActBottleneck-146     [-1, 1024, 14, 14]     0
    GroupNorm-147     [-1, 1024, 14, 14]     2,048
        ReLU-148     [-1, 1024, 14, 14]     0
    StdConv2d-149     [-1, 256, 14, 14]     262,144
    GroupNorm-150     [-1, 256, 14, 14]     512
        ReLU-151     [-1, 256, 14, 14]     0
    StdConv2d-152     [-1, 256, 14, 14]     589,824
    GroupNorm-153     [-1, 256, 14, 14]     512
        ReLU-154     [-1, 256, 14, 14]     0
    StdConv2d-155     [-1, 1024, 14, 14]     262,144
PreActBottleneck-156     [-1, 1024, 14, 14]     0
    GroupNorm-157     [-1, 1024, 14, 14]     2,048
        ReLU-158     [-1, 1024, 14, 14]     0
    StdConv2d-159     [-1, 256, 14, 14]     262,144
    GroupNorm-160     [-1, 256, 14, 14]     512
        ReLU-161     [-1, 256, 14, 14]     0
    StdConv2d-162     [-1, 256, 14, 14]     589,824
    GroupNorm-163     [-1, 256, 14, 14]     512
        ReLU-164     [-1, 256, 14, 14]     0
    StdConv2d-165     [-1, 1024, 14, 14]     262,144
PreActBottleneck-166     [-1, 1024, 14, 14]     0
    GroupNorm-167     [-1, 1024, 14, 14]     2,048
        ReLU-168     [-1, 1024, 14, 14]     0
    StdConv2d-169     [-1, 256, 14, 14]     262,144
    GroupNorm-170     [-1, 256, 14, 14]     512
        ReLU-171     [-1, 256, 14, 14]     0
    StdConv2d-172     [-1, 256, 14, 14]     589,824
    GroupNorm-173     [-1, 256, 14, 14]     512
        ReLU-174     [-1, 256, 14, 14]     0
    StdConv2d-175     [-1, 1024, 14, 14]     262,144
PreActBottleneck-176     [-1, 1024, 14, 14]     0
    GroupNorm-177     [-1, 1024, 14, 14]     2,048
        ReLU-178     [-1, 1024, 14, 14]     0
    StdConv2d-179     [-1, 256, 14, 14]     262,144
    GroupNorm-180     [-1, 256, 14, 14]     512
        ReLU-181     [-1, 256, 14, 14]     0
    StdConv2d-182     [-1, 256, 14, 14]     589,824
```

```
        GroupNorm-183          [-1, 256, 14, 14]             512
             ReLU-184          [-1, 256, 14, 14]               0
        StdConv2d-185         [-1, 1024, 14, 14]         262,144
PreActBottleneck-186         [-1, 1024, 14, 14]               0
        GroupNorm-187         [-1, 1024, 14, 14]           2,048
             ReLU-188         [-1, 1024, 14, 14]               0
        StdConv2d-189          [-1, 256, 14, 14]         262,144
        GroupNorm-190          [-1, 256, 14, 14]             512
             ReLU-191          [-1, 256, 14, 14]               0
        StdConv2d-192          [-1, 256, 14, 14]         589,824
        GroupNorm-193          [-1, 256, 14, 14]             512
             ReLU-194          [-1, 256, 14, 14]               0
        StdConv2d-195         [-1, 1024, 14, 14]         262,144
PreActBottleneck-196         [-1, 1024, 14, 14]               0
        GroupNorm-197         [-1, 1024, 14, 14]           2,048
             ReLU-198         [-1, 1024, 14, 14]               0
        StdConv2d-199          [-1, 256, 14, 14]         262,144
        GroupNorm-200          [-1, 256, 14, 14]             512
             ReLU-201          [-1, 256, 14, 14]               0
        StdConv2d-202          [-1, 256, 14, 14]         589,824
        GroupNorm-203          [-1, 256, 14, 14]             512
             ReLU-204          [-1, 256, 14, 14]               0
        StdConv2d-205         [-1, 1024, 14, 14]         262,144
PreActBottleneck-206         [-1, 1024, 14, 14]               0
        GroupNorm-207         [-1, 1024, 14, 14]           2,048
             ReLU-208         [-1, 1024, 14, 14]               0
        StdConv2d-209          [-1, 256, 14, 14]         262,144
        GroupNorm-210          [-1, 256, 14, 14]             512
             ReLU-211          [-1, 256, 14, 14]               0
        StdConv2d-212          [-1, 256, 14, 14]         589,824
        GroupNorm-213          [-1, 256, 14, 14]             512
             ReLU-214          [-1, 256, 14, 14]               0
        StdConv2d-215         [-1, 1024, 14, 14]         262,144
PreActBottleneck-216         [-1, 1024, 14, 14]               0
        GroupNorm-217         [-1, 1024, 14, 14]           2,048
             ReLU-218         [-1, 1024, 14, 14]               0
        StdConv2d-219          [-1, 256, 14, 14]         262,144
        GroupNorm-220          [-1, 256, 14, 14]             512
             ReLU-221          [-1, 256, 14, 14]               0
        StdConv2d-222          [-1, 256, 14, 14]         589,824
        GroupNorm-223          [-1, 256, 14, 14]             512
             ReLU-224          [-1, 256, 14, 14]               0
        StdConv2d-225         [-1, 1024, 14, 14]         262,144
PreActBottleneck-226         [-1, 1024, 14, 14]               0
        GroupNorm-227         [-1, 1024, 14, 14]           2,048
             ReLU-228         [-1, 1024, 14, 14]               0
        StdConv2d-229          [-1, 256, 14, 14]         262,144
```

```
       GroupNorm-230          [-1, 256, 14, 14]             512
            ReLU-231          [-1, 256, 14, 14]               0
       StdConv2d-232          [-1, 256, 14, 14]         589,824
       GroupNorm-233          [-1, 256, 14, 14]             512
            ReLU-234          [-1, 256, 14, 14]               0
       StdConv2d-235         [-1, 1024, 14, 14]         262,144
PreActBottleneck-236         [-1, 1024, 14, 14]               0
       GroupNorm-237         [-1, 1024, 14, 14]           2,048
            ReLU-238         [-1, 1024, 14, 14]               0
       StdConv2d-239          [-1, 256, 14, 14]         262,144
       GroupNorm-240          [-1, 256, 14, 14]             512
            ReLU-241          [-1, 256, 14, 14]               0
       StdConv2d-242          [-1, 256, 14, 14]         589,824
       GroupNorm-243          [-1, 256, 14, 14]             512
            ReLU-244          [-1, 256, 14, 14]               0
       StdConv2d-245         [-1, 1024, 14, 14]         262,144
PreActBottleneck-246         [-1, 1024, 14, 14]               0
       GroupNorm-247         [-1, 1024, 14, 14]           2,048
            ReLU-248         [-1, 1024, 14, 14]               0
       StdConv2d-249          [-1, 256, 14, 14]         262,144
       GroupNorm-250          [-1, 256, 14, 14]             512
            ReLU-251          [-1, 256, 14, 14]               0
       StdConv2d-252          [-1, 256, 14, 14]         589,824
       GroupNorm-253          [-1, 256, 14, 14]             512
            ReLU-254          [-1, 256, 14, 14]               0
       StdConv2d-255         [-1, 1024, 14, 14]         262,144
PreActBottleneck-256         [-1, 1024, 14, 14]               0
       GroupNorm-257         [-1, 1024, 14, 14]           2,048
            ReLU-258         [-1, 1024, 14, 14]               0
       StdConv2d-259          [-1, 256, 14, 14]         262,144
       GroupNorm-260          [-1, 256, 14, 14]             512
            ReLU-261          [-1, 256, 14, 14]               0
       StdConv2d-262          [-1, 256, 14, 14]         589,824
       GroupNorm-263          [-1, 256, 14, 14]             512
            ReLU-264          [-1, 256, 14, 14]               0
       StdConv2d-265         [-1, 1024, 14, 14]         262,144
PreActBottleneck-266         [-1, 1024, 14, 14]               0
       GroupNorm-267         [-1, 1024, 14, 14]           2,048
            ReLU-268         [-1, 1024, 14, 14]               0
       StdConv2d-269          [-1, 256, 14, 14]         262,144
       GroupNorm-270          [-1, 256, 14, 14]             512
            ReLU-271          [-1, 256, 14, 14]               0
       StdConv2d-272          [-1, 256, 14, 14]         589,824
       GroupNorm-273          [-1, 256, 14, 14]             512
            ReLU-274          [-1, 256, 14, 14]               0
       StdConv2d-275         [-1, 1024, 14, 14]         262,144
PreActBottleneck-276         [-1, 1024, 14, 14]               0
```

```
      GroupNorm-277       [-1, 1024, 14, 14]          2,048
           ReLU-278       [-1, 1024, 14, 14]              0
      StdConv2d-279        [-1, 256, 14, 14]        262,144
      GroupNorm-280        [-1, 256, 14, 14]            512
           ReLU-281        [-1, 256, 14, 14]              0
      StdConv2d-282        [-1, 256, 14, 14]        589,824
      GroupNorm-283        [-1, 256, 14, 14]            512
           ReLU-284        [-1, 256, 14, 14]              0
      StdConv2d-285       [-1, 1024, 14, 14]        262,144
PreActBottleneck-286       [-1, 1024, 14, 14]              0
      GroupNorm-287       [-1, 1024, 14, 14]          2,048
           ReLU-288       [-1, 1024, 14, 14]              0
      StdConv2d-289        [-1, 256, 14, 14]        262,144
      GroupNorm-290        [-1, 256, 14, 14]            512
           ReLU-291        [-1, 256, 14, 14]              0
      StdConv2d-292        [-1, 256, 14, 14]        589,824
      GroupNorm-293        [-1, 256, 14, 14]            512
           ReLU-294        [-1, 256, 14, 14]              0
      StdConv2d-295       [-1, 1024, 14, 14]        262,144
PreActBottleneck-296       [-1, 1024, 14, 14]              0
      GroupNorm-297       [-1, 1024, 14, 14]          2,048
           ReLU-298       [-1, 1024, 14, 14]              0
      StdConv2d-299        [-1, 256, 14, 14]        262,144
      GroupNorm-300        [-1, 256, 14, 14]            512
           ReLU-301        [-1, 256, 14, 14]              0
      StdConv2d-302        [-1, 256, 14, 14]        589,824
      GroupNorm-303        [-1, 256, 14, 14]            512
           ReLU-304        [-1, 256, 14, 14]              0
      StdConv2d-305       [-1, 1024, 14, 14]        262,144
PreActBottleneck-306       [-1, 1024, 14, 14]              0
      GroupNorm-307       [-1, 1024, 14, 14]          2,048
           ReLU-308       [-1, 1024, 14, 14]              0
      StdConv2d-309         [-1, 2048, 7, 7]      2,097,152
      StdConv2d-310        [-1, 512, 14, 14]        524,288
      GroupNorm-311        [-1, 512, 14, 14]          1,024
           ReLU-312        [-1, 512, 14, 14]              0
      StdConv2d-313          [-1, 512, 7, 7]      2,359,296
      GroupNorm-314          [-1, 512, 7, 7]          1,024
           ReLU-315          [-1, 512, 7, 7]              0
      StdConv2d-316         [-1, 2048, 7, 7]      1,048,576
PreActBottleneck-317         [-1, 2048, 7, 7]              0
      GroupNorm-318         [-1, 2048, 7, 7]          4,096
           ReLU-319         [-1, 2048, 7, 7]              0
      StdConv2d-320          [-1, 512, 7, 7]      1,048,576
      GroupNorm-321          [-1, 512, 7, 7]          1,024
           ReLU-322          [-1, 512, 7, 7]              0
      StdConv2d-323          [-1, 512, 7, 7]      2,359,296
      GroupNorm-324           [-1, 512, 7, 7]           1,024
           ReLU-325           [-1, 512, 7, 7]               0
      StdConv2d-326          [-1, 2048, 7, 7]       1,048,576
PreActBottleneck-327          [-1, 2048, 7, 7]               0
      GroupNorm-328          [-1, 2048, 7, 7]           4,096
           ReLU-329          [-1, 2048, 7, 7]               0
      StdConv2d-330           [-1, 512, 7, 7]       1,048,576
      GroupNorm-331           [-1, 512, 7, 7]           1,024
           ReLU-332           [-1, 512, 7, 7]               0
      StdConv2d-333           [-1, 512, 7, 7]       2,359,296
      GroupNorm-334           [-1, 512, 7, 7]           1,024
           ReLU-335           [-1, 512, 7, 7]               0
      StdConv2d-336          [-1, 2048, 7, 7]       1,048,576
PreActBottleneck-337          [-1, 2048, 7, 7]               0
      GroupNorm-338          [-1, 2048, 7, 7]           4,096
           ReLU-339          [-1, 2048, 7, 7]               0
AdaptiveAvgPool2d-340          [-1, 2048, 1, 1]               0
         Conv2d-341           [-1, 133, 1, 1]         272,517
================================================================
Total params: 42,764,997
Trainable params: 42,764,997
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 415.40
Params size (MB): 163.14
Estimated Total Size (MB): 579.11
----------------------------------------------------------------
```