Capstone Project Report: Dog Breed Classifier with CNNs

by Ali Mert Ertugrul - Nov 29, 2020

Definition

Project Overview

Dog breed identification or classification is a typical task in the image processing / computer vision domain. The purpose is to identify/classify the dog breed based on a given picture of a dog as the input. Such an application would be very useful for a large audience (e.g., children) who want to learn and investigate more about the dogs. The purpose is basically to detect the breed of a dog based on a given input of the dog image. This task has been widely studied in the literature by employing various techniques. For instance, Liu et al [1]. employed part localization technique to perform dog breed classification. In another study, Wang et al. [2] modeled the shape of dog breed as points on the Grassman manifold. Then, they performed dog breed classification on that manifold. More recently, Borwarnginn et al. [3] used Convolutional Neural Networks (CNNs) with transfer learning to classify dog breeds and compared the results with those of conventional methods. In this project, I also aim to build a dog breed classifier using CNNs and transfer learning, and evaluate this classifier on the dataset including 8351 images of 133 different breeds.

Problem Statement

The problem within the scope of this project is basically to estimate the dog breed based on a given picture of a dog as the input. This is a multi-class classification problem. Therefore, the goal is to build a machine learning pipeline to process real-world, user-supplied images for this task. In general, this pipeline will take an image as the input and estimate the breed of the dog in the picture. More specifically, the pipeline will perform the following tasks based on a given image as the input:

1. Estimate the canine's breed based on the given image of a dog with a reasonable accuracy.
2. Estimate the most resembling canine's breed if the supplied image (input) is an image of a human.

Metrics

The goal is to evaluate the model performance based on evaluation metrics. Since this is a multi-class classification task, *Accuracy* is selected as the main evaluation metric in this work. It is also convenient to compare accuracy results with the benchmark models since they provide accuracy results as well. The accuracy is calculated as follows:

$$accuracy = \#correctly\ predicted\ samples\ /\ \#total\ samples$$

Analysis

Data Exploration & Visualization

For this project, two different datasets are provided by Udacity. Both datasets include images. The first dataset, called dog-image dataset, includes images of dogs where the second dataset, called human-image dataset, includes images of humans. The more detailed information is provided for the corresponding datasets as follows:

1. *Dog-image dataset:* It includes 8351 dog images for 133 different dog breeds. The properties of the images are not consistent meaning that the image size, background and illumination may vary across the images. The numbers of the images for each breed are not uniformly distributed. For some of the breeds the dataset contains more images whereas it contains less images for other breeds. Therefore, the dog-image dataset is an imbalance dataset. Furthermore, the dataset is divided into three sets, training, validation and test, where there are 6680, 835, and 836 images for training, validation and test sets, respectively. They are used for training, validation and test in estimation of the breeds in the images.
2. *Human-image dataset:* It includes 13233 human images from 5749 different people. For some people, there are more images whereas for some of them there are less images. Therefore, the dataset is not uniformly distributed with respect to the people. All images in the dataset have the fixed image size which is 250x250 although the background and illumination may vary across these images. They are used to test the human face detected as well as estimating the resembling breed for the given human image.

Three examples from each dataset are illustrated in Figure 1.

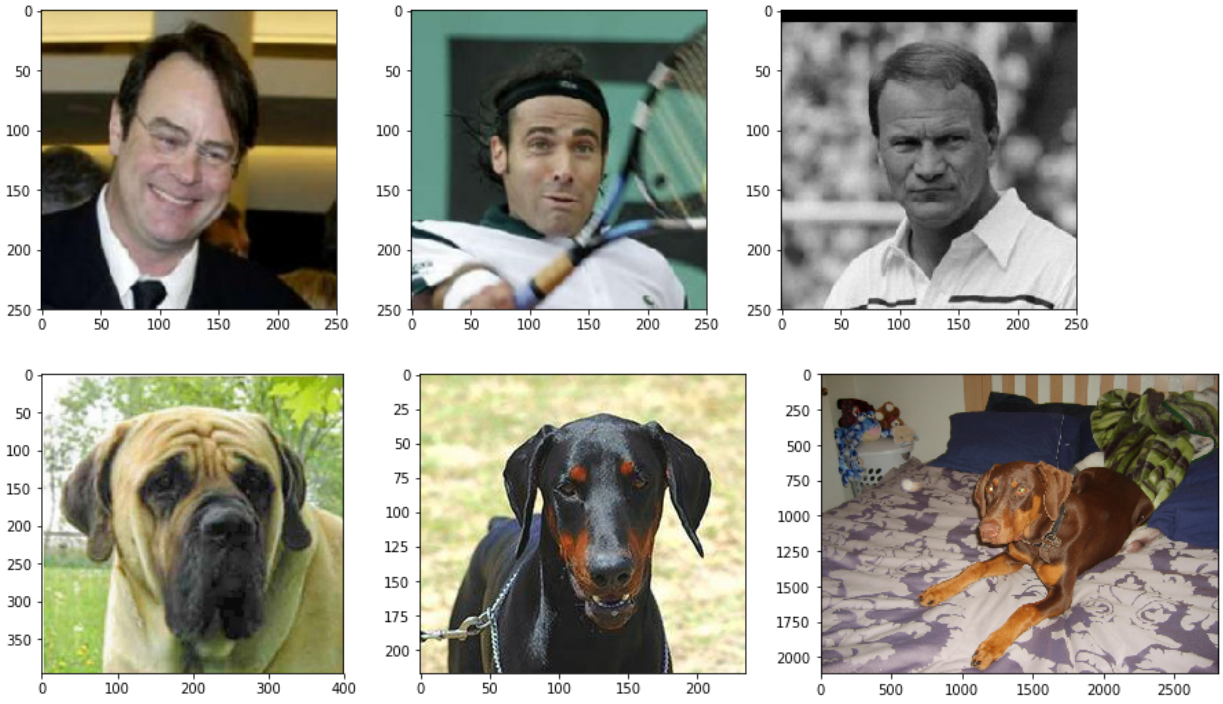

Figure 1 - Example images from human- and dog-image datasets.

The distribution of samples from the first 50 dog breed classes (in order to provide a clear illustration) in the training set is given in Figure 2. We clearly observe that this is an imbalance multi-class classification tas

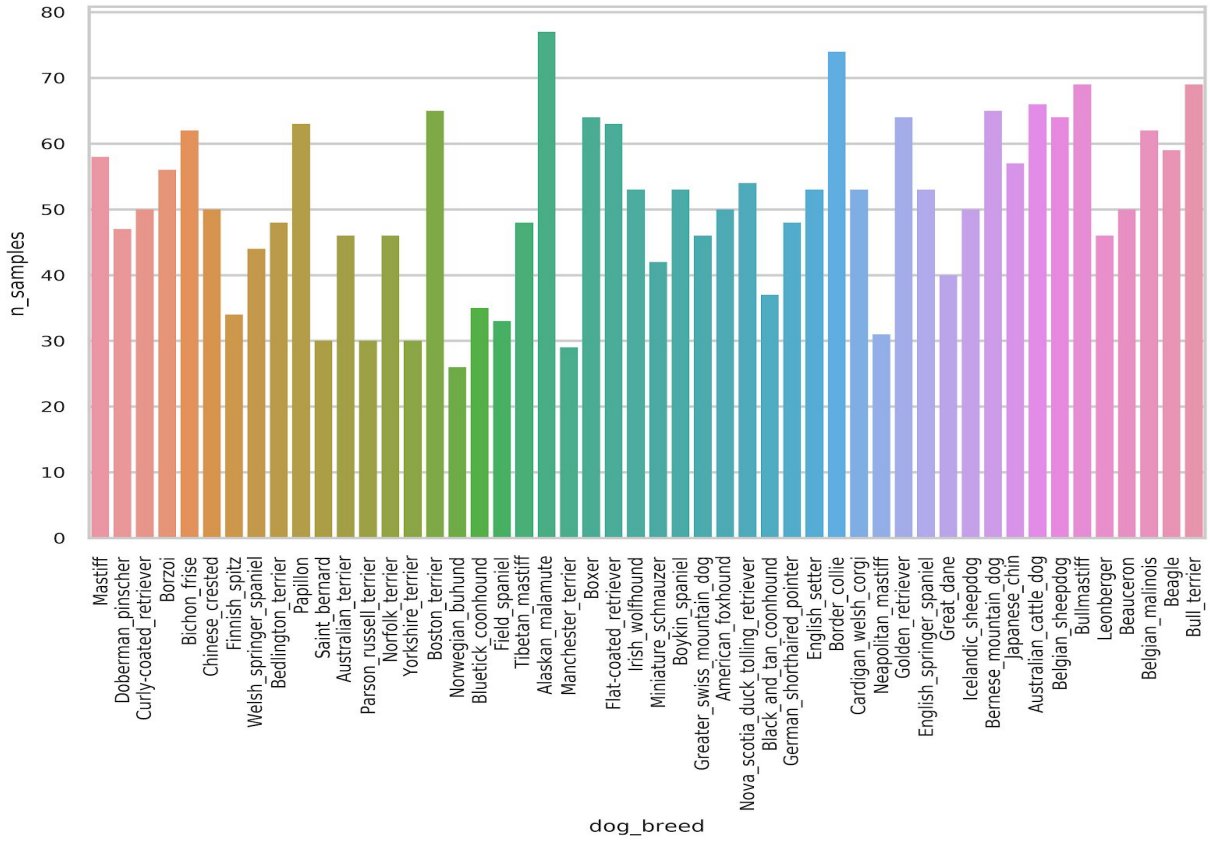

Figure 2 - Distribution of number of samples for the first 50 dog breed classes in the training set.

Algorithms & Techniques

In this work, we will employ Convolutional Neural Networks (CNNs) to perform dog breed classification. Our algorithm will (1) predict the breed of the dog based on a given input image of a dog, (2) estimate the most resembling dog breed based on a given input image of a human. In our algorithm, we will follow the several steps as follows:

1. Detect whether a given input image contains a human. To do that, we use one of the Haar feature-based cascade classifiers of OpenCV[1].
2. Detect whether a given input image contains a dog. For this purpose, we use a pre-trained VGG16 model which is a CNN-based architecture trained on ImageNet [4].
3. Develop a dog breed classifier from the scratch using CNNs. The architecture contains three convolutional layers and 2 linear layers to detect a dog breed from the given input image. The architecture is constructed by investigating the literature in object detection/recognition.

[1] https://opencv.org/

4. Develop a dog breed classifier using transfer learning. We use a pre-trained VGG16 model, and finetune it on our dog-image datasets to detect dog breed from the given input image.

Benchmark

I consider two different benchmark models for this task from the literature. It is important for benchmark model selection to obtain studies, methods that perform the same task on the same dataset. I select the published studies [1] and [2] which perform dog breed classification on the same dataset. They achieved 67%, 96.5% accuracy, respectively. I will compare the results of the proposed solution to those of the benchmark models. Another basic model can be a random classifier which assigns a random dog breed to a given input image. Since there are 133 dog breed classes in the dataset, the random guess accuracy will be less than 1%. As a result, I consider three different models as the benchmark models in this work.

Methodology

Data Preprocessing

I follow different data preprocessing approaches for images in the training set, the images in validation and test sets. For the training set, all images are first scale to (224, 224) since the input of my model takes an input tensor in the same size (3, 224, 224). Then, I follow several augmentation approaches for the images. I randomly flip the images horizontally. Then, I randomly rotate the images between -20 degrees to 20 degrees. By these augmentation operations, my aim is to have a more robust and more accurate model on the validation and the test sets. Finally, I normalize the images by the mean and the standard deviation of the ImageNet, which was used to train the VGG16 model. On the other hand, for the images in the validation and the test sets, I scale them to (224, 224) due to the same reason. I do not perform any data augmentation operation. But, I still normalize these images with the same mean and standard deviation of ImageNet since the model is trained in the same way.

Implementation

Five modules are implemented in this work. The implementation details of each model are given below.

1. *Human-face detector:* I use one of the Haar feature-based cascade classifiers of the OpenCV software, which is already provided by Udacity.
2. *Dog detector:* I use a pre-trained VGG16 model to detect whether a given input image contains a dog. I use PyTorch[2] framework to implement this module. ImageNet consists of 118 dog categories. If the pre-trained VGG16 model returns any of these categories, the module returns true.

[2] https://pytorch.org/

3. *Dog breed classifier from the scratch:* I implement a CNN model for dog breed classification from the scratch. I also use the PyTorch framework to implement this module. Based on my investigation on object detection/classification literature in the image processing domain as well as inspired by the structure of VGG16, I decided to use three convolutional layers, followed by two linear layers (fully connected layers) in my model. Each convolutional layer consists of a convolution operation, a 2d batch normalization operation, a ReLU operation, a max pooling operation and a dropout with 0.2. The reason why I use ReLU operation is to bring non-linearity to the model. The reason behind why I use 2d batch normalization is that it helps stabilize the learning process and reduce the number of training epochs required. Max pooling will find the locations with higher activation. In the first convolutional operation, I preferred to use 'kernel_size=5' and 'stride=2', for the rest of them I use 'kernel_size=5' and 'stride=1'. For each max pooling operation I preferred to use 'kernel_size=2' and 'stride=2' to reduce output size. Through the convolutional layers, the number of channels/kernels increases (i.e., 3, 64, 128, 128) based on the best practices. By using dropout in each convolutional layer, I aim to prevent overfitting and generalize better on the test set. After the convolutional layers, my model has two linear layers. In the first linear layer, I also provided a ReLU operation for non-linearity. In the second and final linear layer, the output size is equal to the number of classes of dog breeds, which is 133. The structure of this CNN model is given in Figure 3.

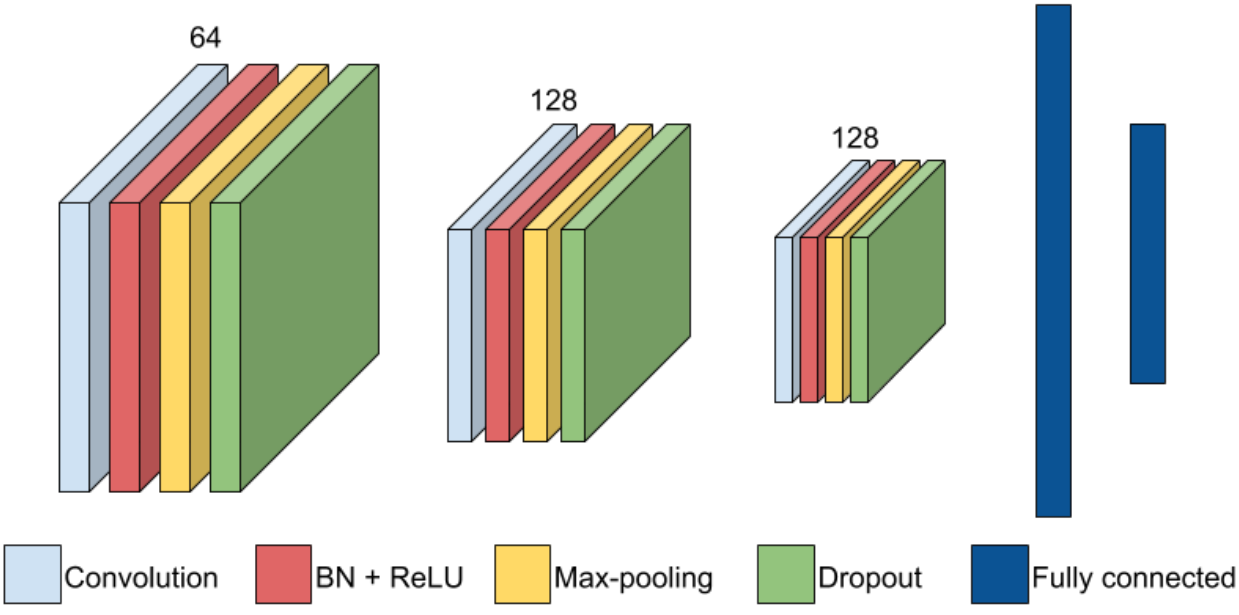

Figure 3 - CNN model from the scratch for dog breed classification.

4. *Dog breed classifier using transfer learning:* I also develop dog breed classifier using transfer learning. Similarly, I utilize PyTorch framework to develop this module. I select a pre-trained VGG16 model as the base, and then I finetune it on the dog-image training

dataset. I only change the last linear layer (fully connected layer) of the pre-trained VGG16 model to make it compatible with our dataset. Since it was pre-trained on ImageNet with 1000 classes, the input and output size of this last layer is 4096 and 1000, respectively. First, I freeze all the layers in the original architecture to prevent backpropagation in all these layers. Then, I change the final linear layer with a new one where the input size and output size are 4096 and 133, respectively. Finally, I trained the model on the dog-image training set. The backpropagation is applied only on the final linear layer.

5. *Run_app:* This module is a combination of the previous models that aims to find either the dog breed of a given dog image or the most resembling dog breed for a given human image as the input. If the given input image does not include either a dog or a human, the module prints an error message.

Refinement

First, I try the dog breed classifier from scratch. It performed 22% accuracy on the test set. Without data augmentation (random flipping and random rotation), it performed worse and yielded 16% accuracy. Next, I go with the dog breed classifier using transfer learning. I use the pre-trained VGG16 model as the base in the model, and finetune it on the dog-image training set. It results in 82% accuracy on the test set after just 4 epochs. It is a huge improvement compared to the CNN model trained from the scratch. This indicates the importance of using transfer learning for such tasks. I also tried the pre-trained ResNet50 model and finetune it in a similar way that the one I follow for the VGG16 model. It yielded slightly better performance, which is 84% accuracy.

Results

Model Evaluation and Validation

This section presents the evaluation and validation of each developed module throughout this work as follows:

1. *Human-face detector:* This module was evaluated on the first 100 images from human-image dataset and dog-image dataset. This module performed 98% accuracy on the short human-image dataset whereas it performed 83% accuracy (misclassified 17% of the images) on the short dog-image dataset.
2. *Dog detector:* This module was evaluated on the first 100 images from human-image dataset and dog-image dataset as well. This module resulted in 100% accuracy on the short dog-image dataset and it also performed 100% accuracy (misclassified 0% of the images) on the short human-image dataset.
3. *Dog breed classifier from scratch:* This module was evaluated on all test images of the dog-image dataset. I use batch-size as 32, number of epochs to train as 40. After the epoch 28 the log loss on the validation set didn't decrease. I use Adam optimizer and 0.001 learning rate, which is the default value in PyTorch framework for the training. As a result, this module produced 22% accuracy. It means that this model correctly classified 184 images of the 836 images in the test set. The line chart for training loss and validation

loss over the epochs is given in Figure 4. I observe that the training loss decreases in each epoch whereas the validation loss decreases until some point and then becomes stable or fluctuates a little.

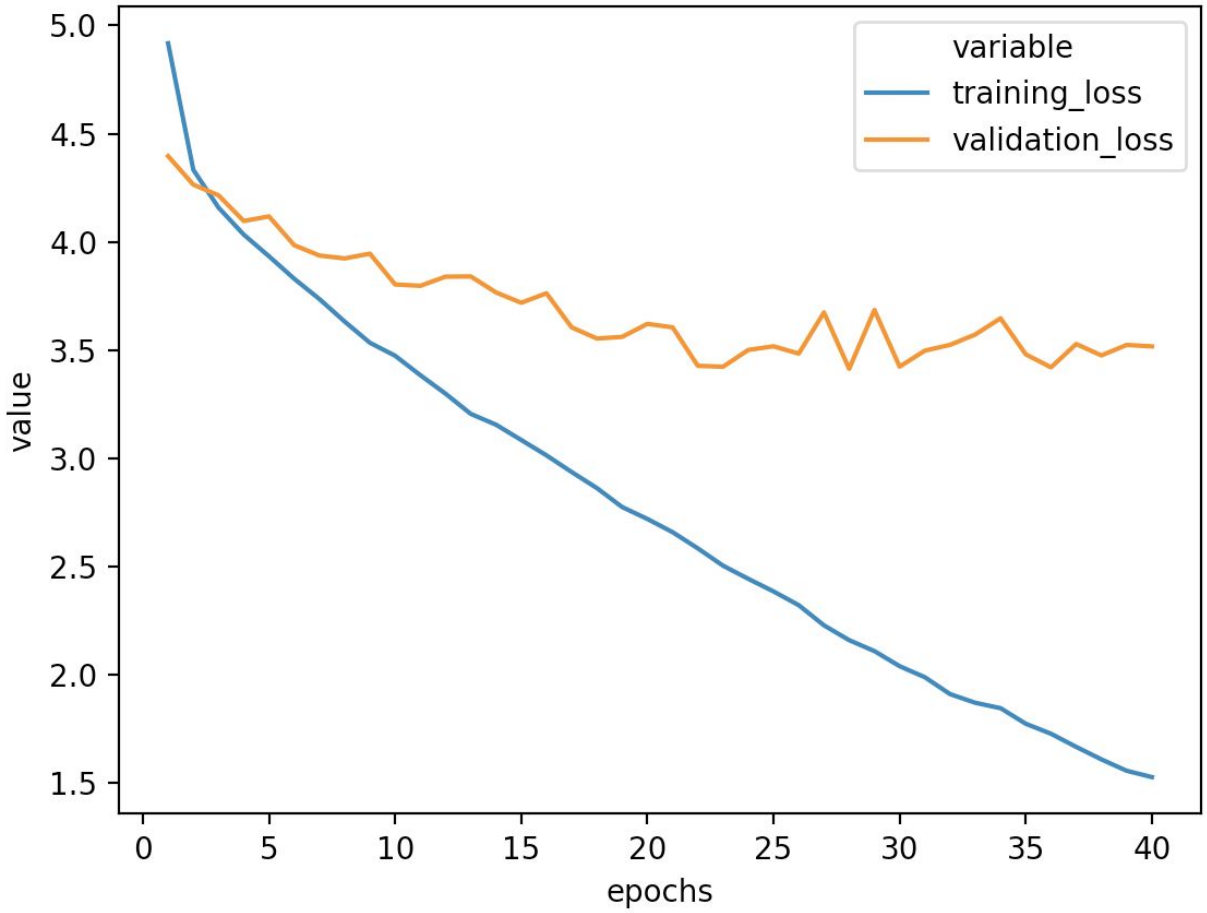

Figure 4 - Training and validation loss over the epochs for Dog breed classifier from scratch.

4. *Dog breed classifier using transfer learning:* This module was evaluated on all test images of the dog-image dataset. I use batch-size as 32, number of epochs to train as 20. After the epoch 4 the log loss on the validation set didn't decrease. I use Adam optimizer and 0.001 learning rate, which is the default value in PyTorch framework for the training. As a result, this module produced 82% accuracy. It means that this model correctly classified 689 images of the 836 images in the test set. This indicates that this model with transfer learning significantly outperforms the dog breed classifier from scratch. The line chart for training loss and validation loss over the epochs is given in Figure 5. I observe that the training loss decreases in each epoch (except for the last 5 epochs) whereas the validation loss decreases until some point (until 4th epoch) and then starts to increase and fluctuates over the epochs while increasing.

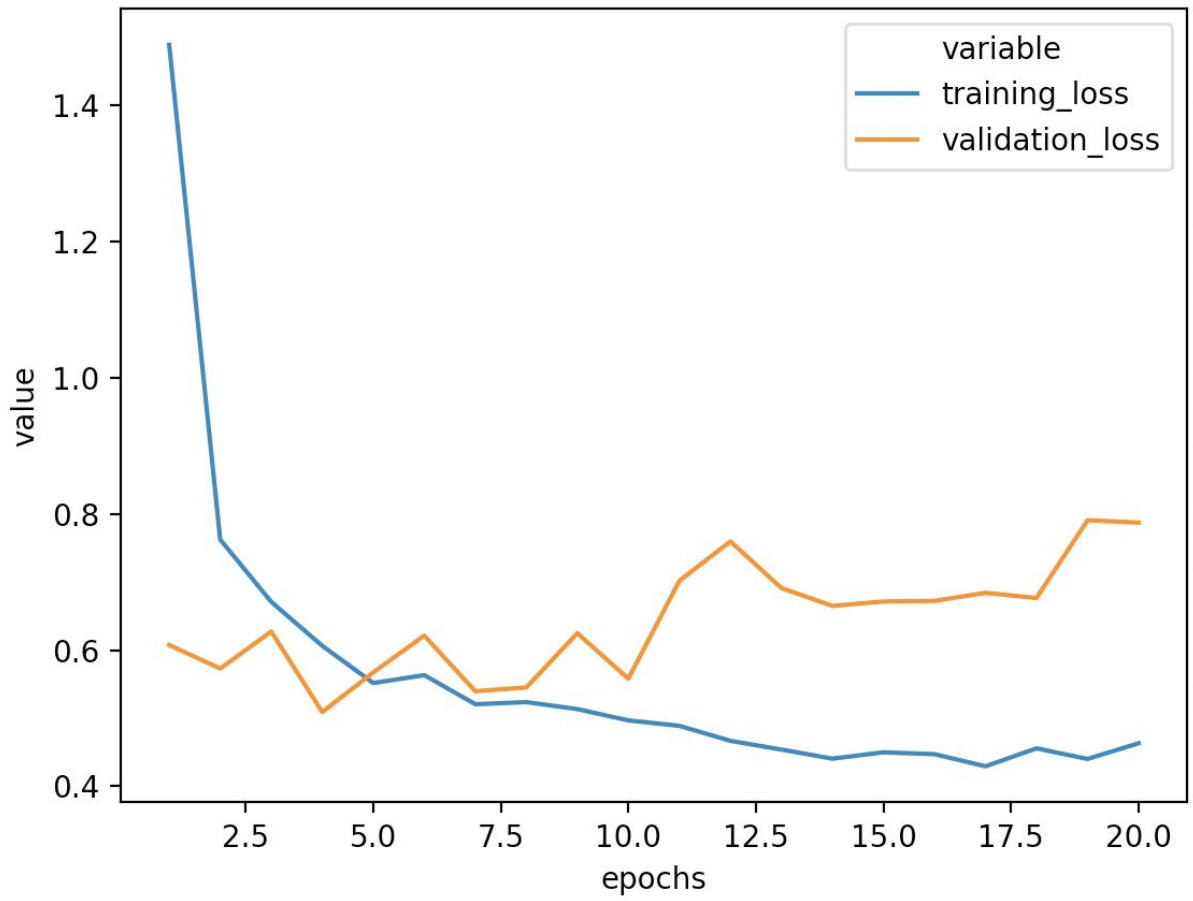

Figure 5 - Training and validation loss over the epochs for Dog breed classifier using transfer learning.

Justification

I think the model performance is better than I expected especially for the *dog breed classifier using transfer learning*. This model yields significantly better accuracy (82%) than the *dog breed classifier from scratch* (22%). These two models satisfied the conditions in the project requirements where the expectation for *dog breed classifier from scratch* is 10% and *classifier using transfer learning* is 60% accuracy. Furthermore, I compare the *accuracy* measures of these models with the benchmark models defined in the "Benchmark" section. The Table 1 indicates the performance of my models as well as the benchmark models on the dog-image dataset.

Table 1 - Performance comparison.

	Dog breed classifier from scratch	Dog breed classifier using transfer learning	[1]	[2]	Baseline (Random guess)
Accuracy	22%	82%	67%	96.5%	< 1%

According to Table 1, *dog breed classifier using transfer learning* outperformed the published study [1], but was beaten by another published study [2]. Also, my both models are much better than the baseline (random guess).

References

[1] Liu, Jiongxin, et al. "Dog breed classification using part localization." European conference on computer vision. Springer, Berlin, Heidelberg, 2012.
[2] Wang, Xiaolong, et al. "Dog breed classification via landmarks." 2014 IEEE International Conference on Image Processing (ICIP). IEEE, 2014.
[3] Borwarnginn, Punyanuch, et al. "Breakthrough Conventional Based Approach for Dog Breed Classification Using CNN with Transfer Learning." 2019 11th International Conference on Information Technology and Electrical Engineering (ICITEE). IEEE, 2019.
[4] Deng, Jia, et al. "Imagenet: A large-scale hierarchical image database." 2009 IEEE conference on computer vision and pattern recognition. IEEE, 2009.