# Dog Breed Classifier

# Group members:

Deepkumar Jigneshbhai Patel (dp29)

Jay Ashwinkumar Sorathiya (js2575)

Twinkle Deepakkumar Jariwala (tj257)

# Contribution:

Each member of the group has equally contributed to the coding of the project. Each member has contributed to the report by adding the information of their part of coding.

# Problem Statement

It is rather simple for us, humans, to distinguish one dog breed from another. If you're talking about 10 to 20 common dog breeds, that is pretty easy. It's a different story when we're talking about more than 100 different breeds of dog. A different strategy than sheer memorization is required for a person to classify many breeds correctly and consistently. We need to start identifying features associated with distinct breeds, such as fur color, ear shape, facial form, tail length, and so forth. Even then, we must memorize which breeds have certain characteristics, which is neither simple nor enjoyable.

Humans having trouble identifying a huge number of species is an excellent example of why we need to solve this problem using Deep Learning. We'll show you how to create and train a convolutional neural network (CNN) that can classify 133 different dog breeds in this project. Given an image of a dog, our algorithm will identify an estimate of the dog’s breed. If supplied an image of a human, the code will identify the resembling dog breed. Main objective of this project will be to perform a systematic evaluation of various methods for detecting humans and dogs in images. Provide improved methodology for the face detector and dog detector functions.A picture containing dog, looking, mammal

Description automatically generated

# Data set and feature description

The dataset used in our project can be downloaded with this link:

Dog Dataset: <https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/dogImages.zip>

Human Dataset: <https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/lfw.zip>

The dataset contains 8,351 dog images, 133 dog breeds, that were split into 6,680 images for training, 836 images for testing, and 835 images for validation.

Different images have different sized but the code resize the input into 224\*224 pixel image and it is then loaded to memory as numpy series.

# Model and hyperparameters

We have normalized our data to eliminate units of measurements. It also helps our model to better compare data to different scales.

## Model Architecture

We are creating CNN with Keras. Our model architecture has an

* Input layer with a shape of 224\*224\*3

The inputs are then given to two convolutional layers, followed by

* Max-pooling layer (For downsampling)

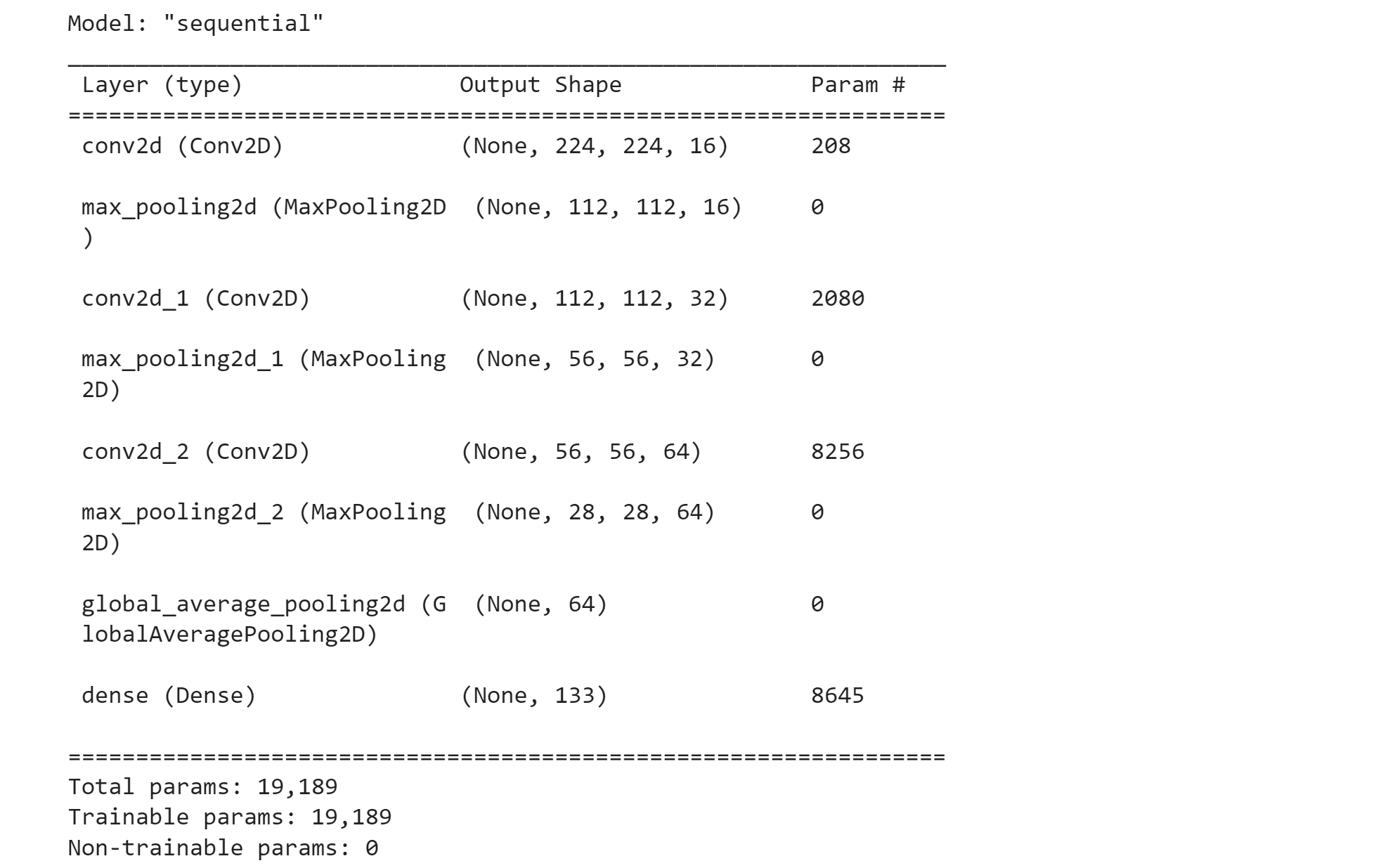
The output is then given to

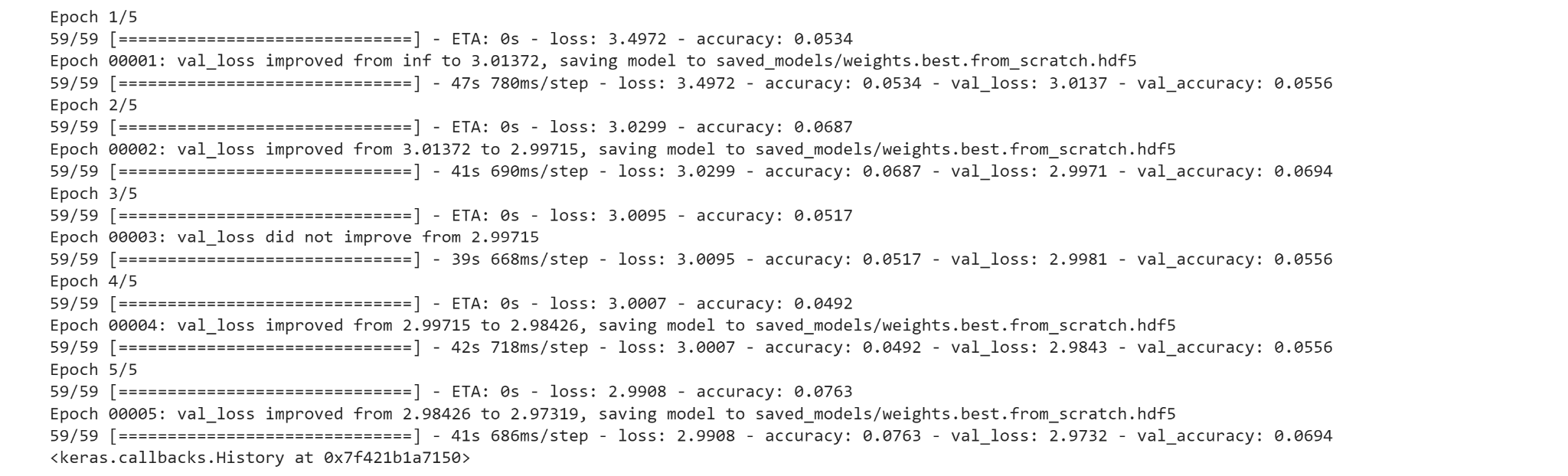
* Global Average pooling layer (to minimize overfitting)

The output layer returns 133 probabilities for dog breeds.

Model Summary:

| **Layers** | **Output Shape** | **No of Parameters** |
| --- | --- | --- |
| Conv2D | (None, 224, 224, 16) | 208 |
| MaxPooling2 | (None, 112, 112, 16) | 0 |
| Conv2D | (None, 112, 112, 32) | 2080 |
| MaxPooling2 | (None, 56, 56, 32) | 0 |
| Conv2D | (None, 56, 56, 64) | 8256 |
| MaxPooling2 | (None, 28, 28, 64) | 0 |
| MaxPooling2 (Global average pooling) | (None, 64) | 0 |
| Dense | (None, 133) | 8645 |
| Total Parameters: 19,189  Trainable Parameters: 19,189 |  |  |



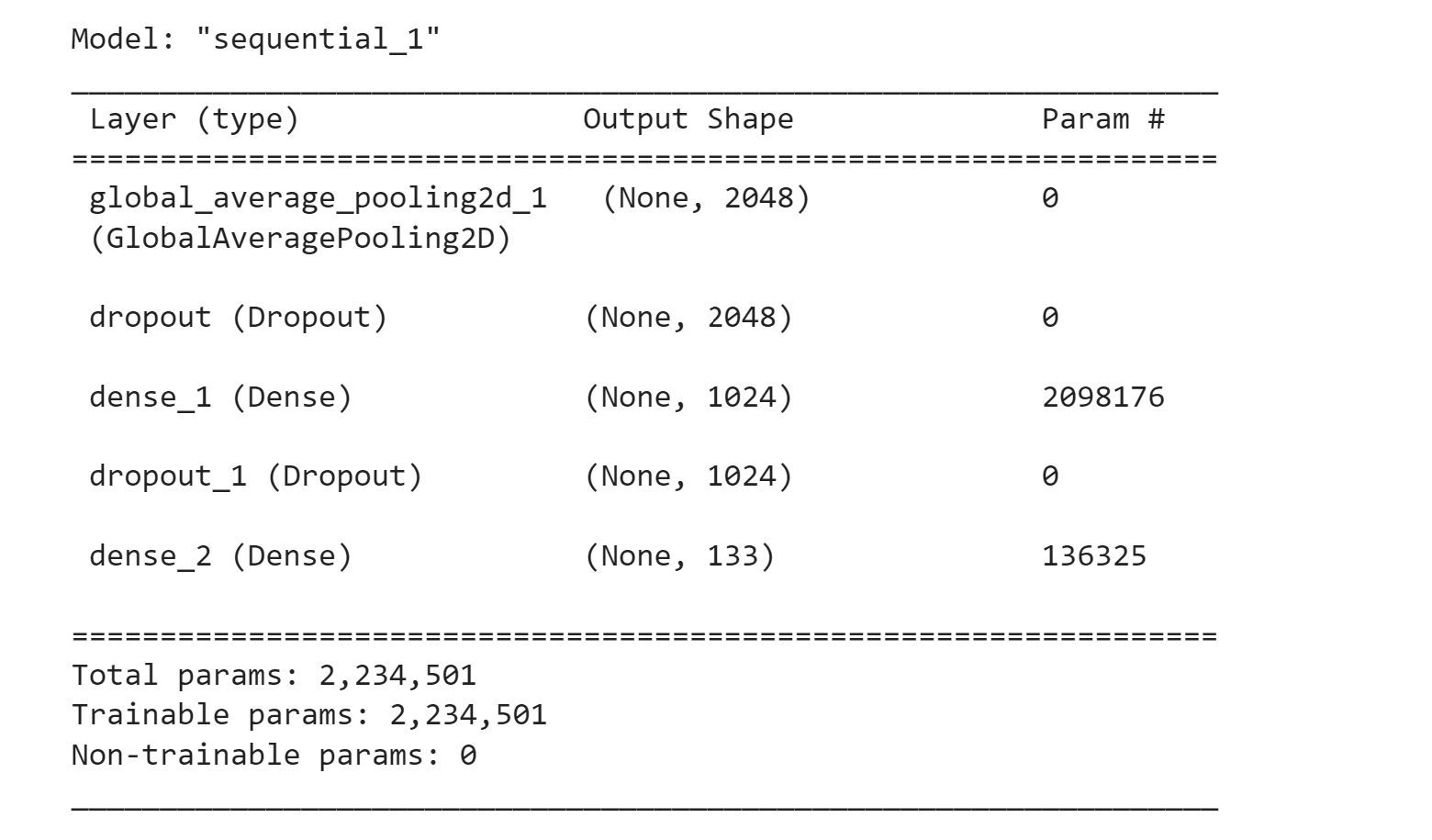


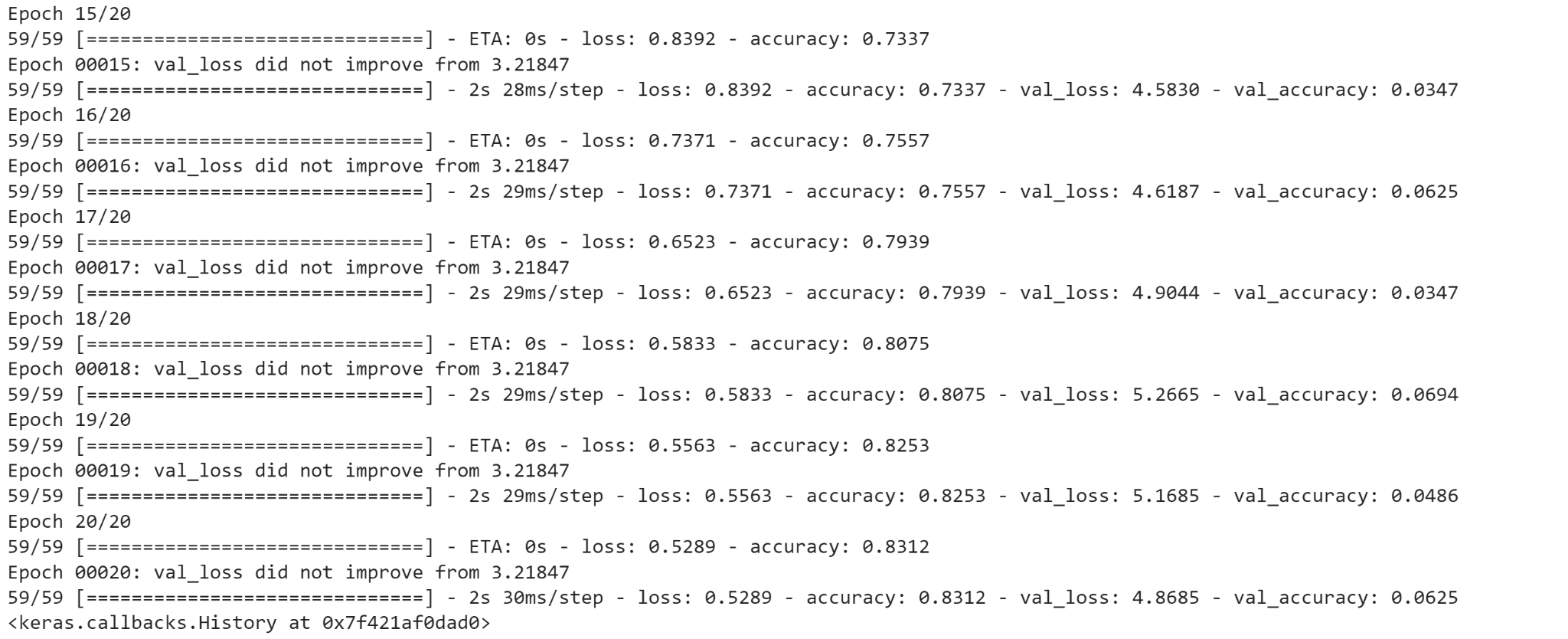
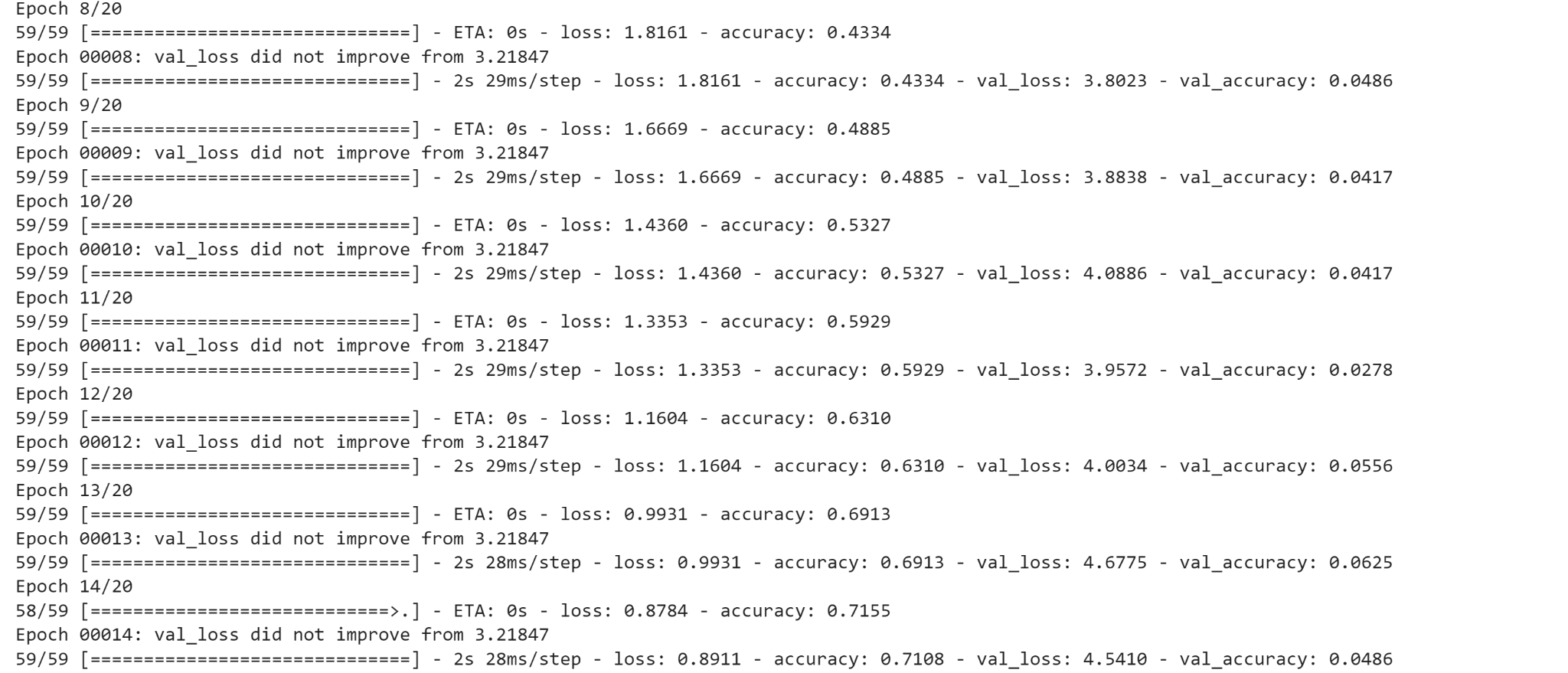
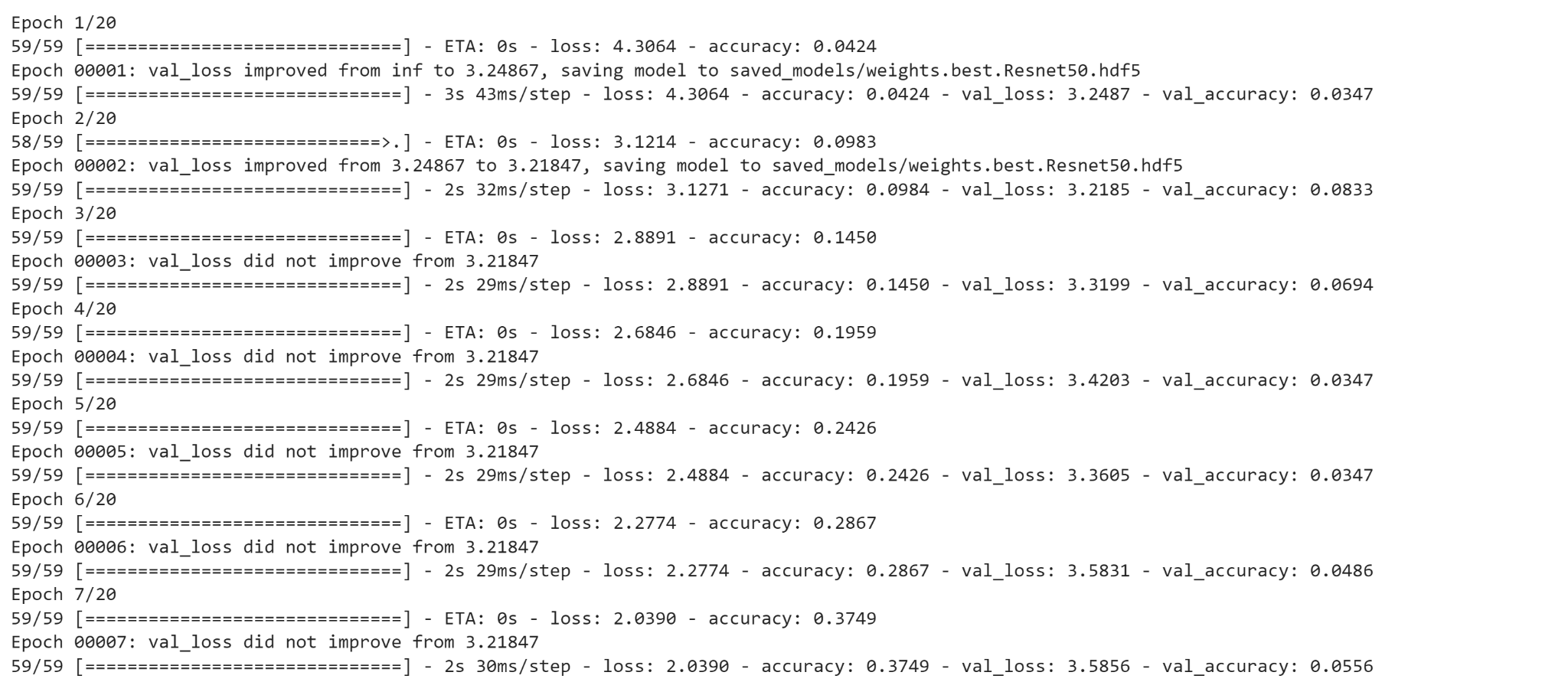
With five epochs, it takes about eight minutes with an accuracy of 7.63%. This is not useful in reality. So, we used transfer learning to reduce training time without sacrificing accuracy.

We used ResNet-50 offered by Keras to obtain the bottle neck feature in transfer learning. With the pre-trained ResNet-50 model, we created our fully connected layers and also added two drop out layers to prevent overfitting.

Updated Model:

| **Layers** | **Output Shape** | **No of Parameters** |
| --- | --- | --- |
| MaxPooling2 (Global average pooling) | (None, 2048) | 0 |
| Dropout | (None, 2048) | 0 |
| Dense | (None, 1024) | 2098176 |
| Dropout | (None, 1024) | 0 |
| Dense | (None, 133) | 136325 |
| Total Parameters: 2,234,501  Trainable Parameters: 2,234,501  Non- Trainable Parameters: 0 | | |





Using transfer learning (ResNet-50), with twenty epochs and less than two minutes, we have achieved a test accuracy of 83.12%.