University of Riverside, California

CS 147 Project

DOG BREED CLASSIFICATION

USING TENSORFLOW AND KERAS

June 11, 2019

Tasmiyah Qazi
861218356

Melissa Santos
861213813

Project Idea/Overview

Use a convolutional neural network built with TensorFlow, Keras and Keras via transfer learning to classify dog breeds from the stanford dog dataset as well as pictures of our own dogs. Compare the training times, training accuracy, and accuracy validation between each implementation and the time it takes on CPU vs GPU.

The data set we chose does not have enough images per dog breed to train a deep neural network (120 breeds, ~200 images per breed). So the accuracy of our model is low. That's why we chose Keras, because it is a high level API used to build and train neural networks. It is also easier to implement transfer learning (Take an already pre-trained NN on a larger dataset and use it to our advantage) using Keras.

GPU Acceleration

A typical convolutional neural network has multiple convolutional layers. An image goes through a convolutional layer by a filter (kernel) shifting by a stride amount to extract features from the image. At each operation, matrix multiply of the kernel and the current region of input is calculated. This is where GPUs come in!

If a tensor operation has a GPU implementation, the GPU will be given priority when that operation is being assigned a device to run on. Both TensorFlow and Keras will try to automatically assign the best device for each operation without the user having to specify.

To make your code specifically run on a certain device:

```
with tf.device("/GPU:0"):
    # insert code you want run on GPU
with tf.device("/CPU:0"):
    # insert code you want run on CPU
```

You can see what operations are running on what device by setting log_device_placement to True. An example is given below

```
a = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], shape=[2, 3], name='a')
b = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], shape=[3, 2], name='b')
c = tf.matmul(a, b)
sess = tf.Session(config=tf.ConfigProto(log_device_placement=True))
print(sess.run(c))
```

The device assignments will be logged to the output console.

Implementation Details

Convolutional Neural Network (CNN)

- **Convolution:** A filter that is slid over the image that performs a certain function.
- **Rectified Linear Units (ReLU):** Activation function f(x) = max(x, 0). Eliminates negative values.
- **Pooling:** Downsampling layer that reduces the number of parameters. Helps reduce computation.
- **Fully Connected Layer:** Feed-forward multi-layer
- **Dropout Layer:** Layer that prevents from overfitting
- **Flatten Layer:** Turns 2D array into 1D array by appending the second tuple to the first tuple and so on and so forth

The CNN architecture we used:

- Convolution 32
  - Max Pooling
  - ReLU
- Convolution 64
  - Max Pooling
  - ReLU
- Convolution 128
  - Max Pooling
  - ReLU
  - DropOut
- Flatten
- Fully Connected Layer of 500 Nodes
  - ReLU
  - DropOut
- Fully Connected Layer with N dog breeds (n = number of dog breeds)

We use ReLU as our activation function and max pooling to down-sample. These are standard in a convolutional neural network. We also use dropout to in our 3 convolution layer to prevent overfitting. The last fully connected layer has N nodes with N being the number of dog breeds we want to classify.

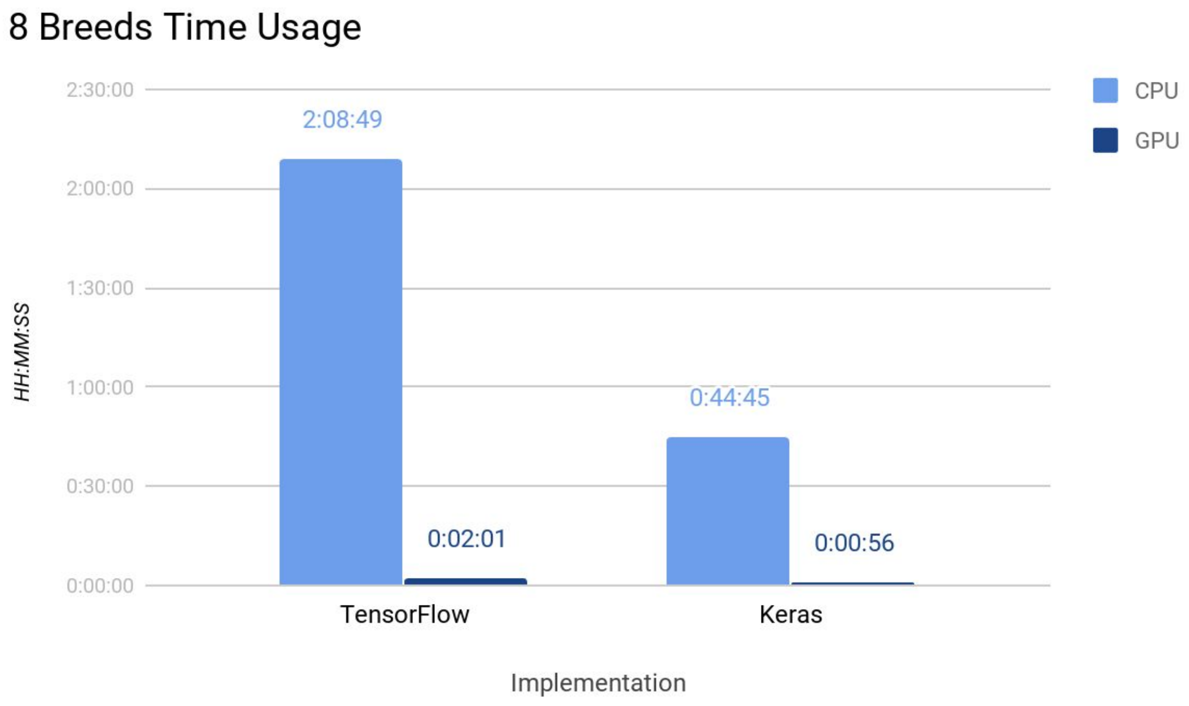
How To Run

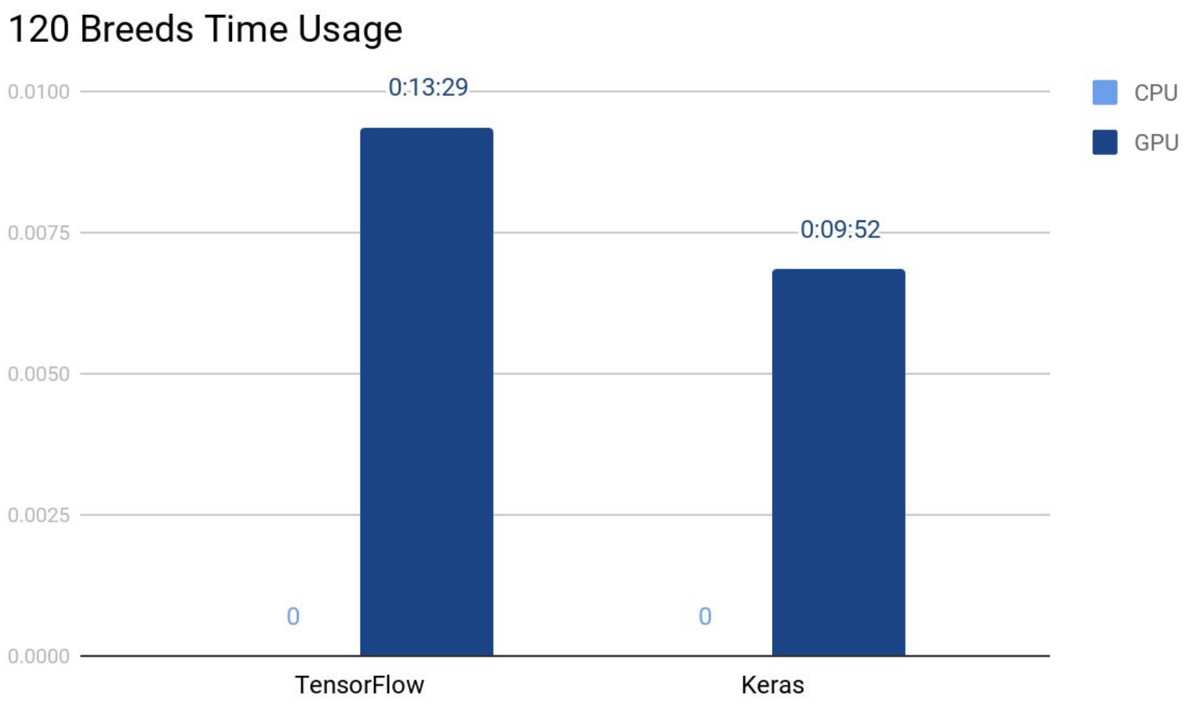
Our code and data is saved in a team drive CS147 - Cucumbers. There are Google Colab notebooks for Tensorflow, Keras and Keras with transfer learning. We have included you in our team drive and a link to it so you can access all of the notebooks and test data we used. Select a notebook and run each cell in sequential order to ensure the code executes correctly.

Team Drive: https://drive.google.com/drive/folders/0ACxC3fKgAszdUk9PVA

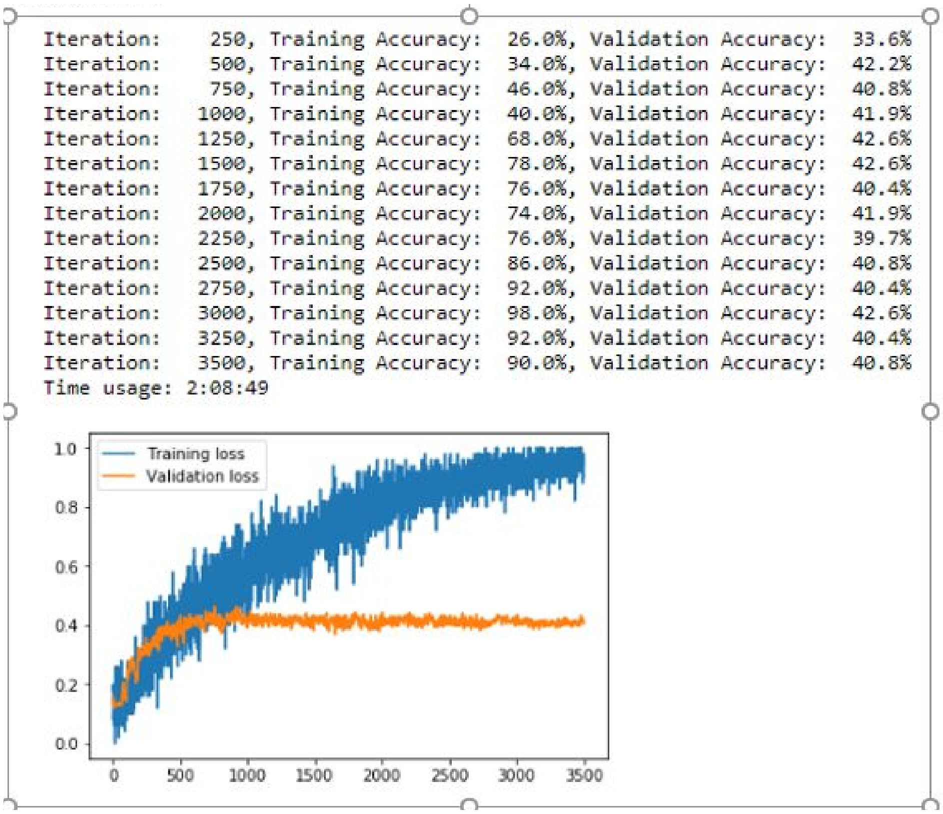
To enable the GPU go to **Runtime > Change Runtime** and under **Hardware accelerator** click on the dropdown and select **GPU**.

Evaluation/Results



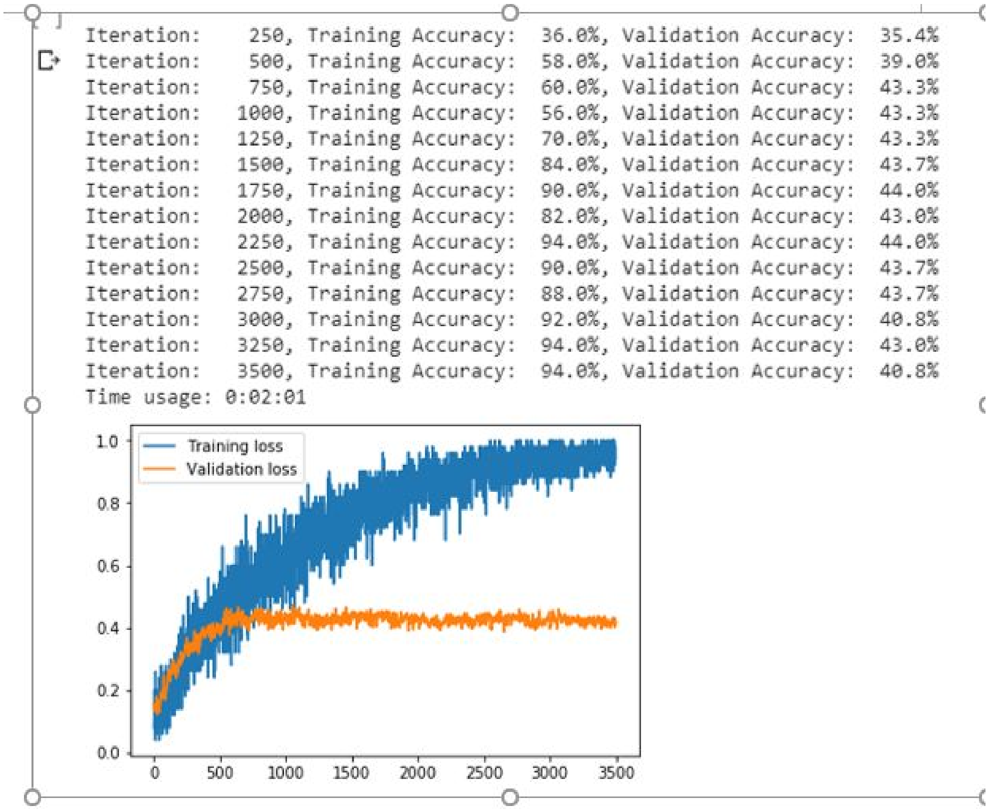


Tensorflow with 8 breeds - CPU:



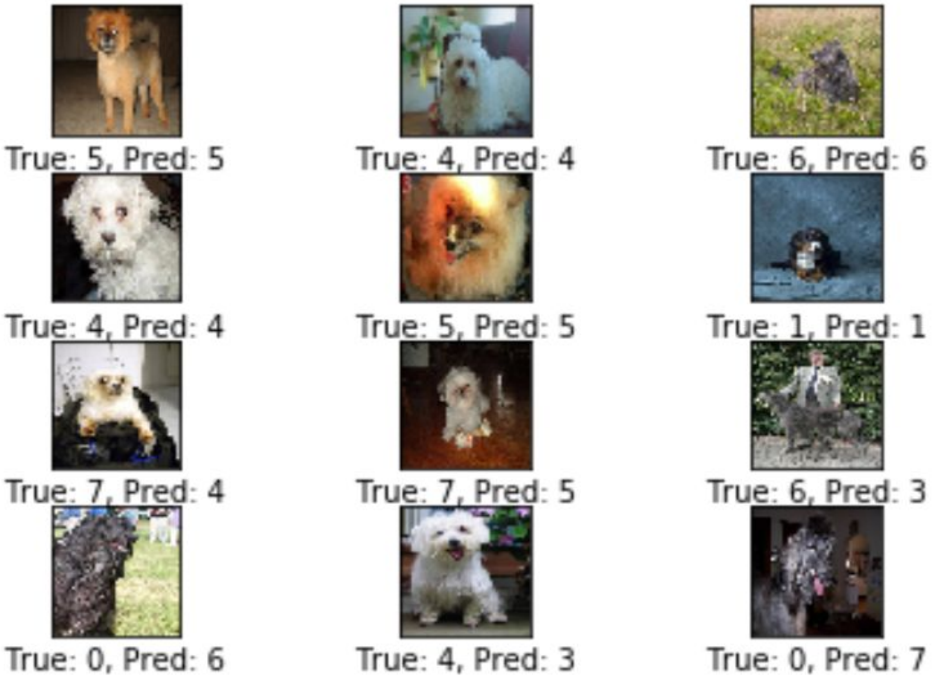
```
Iteration:    250, Training Accuracy:  26.0%, Validation Accuracy:  33.6%
Iteration:    500, Training Accuracy:  34.0%, Validation Accuracy:  42.2%
Iteration:    750, Training Accuracy:  46.0%, Validation Accuracy:  40.8%
Iteration:   1000, Training Accuracy:  40.0%, Validation Accuracy:  41.9%
Iteration:   1250, Training Accuracy:  68.0%, Validation Accuracy:  42.6%
Iteration:   1500, Training Accuracy:  78.0%, Validation Accuracy:  42.6%
Iteration:   1750, Training Accuracy:  76.0%, Validation Accuracy:  40.4%
Iteration:   2000, Training Accuracy:  74.0%, Validation Accuracy:  41.9%
Iteration:   2250, Training Accuracy:  76.0%, Validation Accuracy:  39.7%
Iteration:   2500, Training Accuracy:  86.0%, Validation Accuracy:  40.8%
Iteration:   2750, Training Accuracy:  92.0%, Validation Accuracy:  40.4%
Iteration:   3000, Training Accuracy:  98.0%, Validation Accuracy:  42.6%
Iteration:   3250, Training Accuracy:  92.0%, Validation Accuracy:  40.4%
Iteration:   3500, Training Accuracy:  90.0%, Validation Accuracy:  40.8%
Time usage: 2:08:49
```

Tensorflow with 8 breeds - GPU:

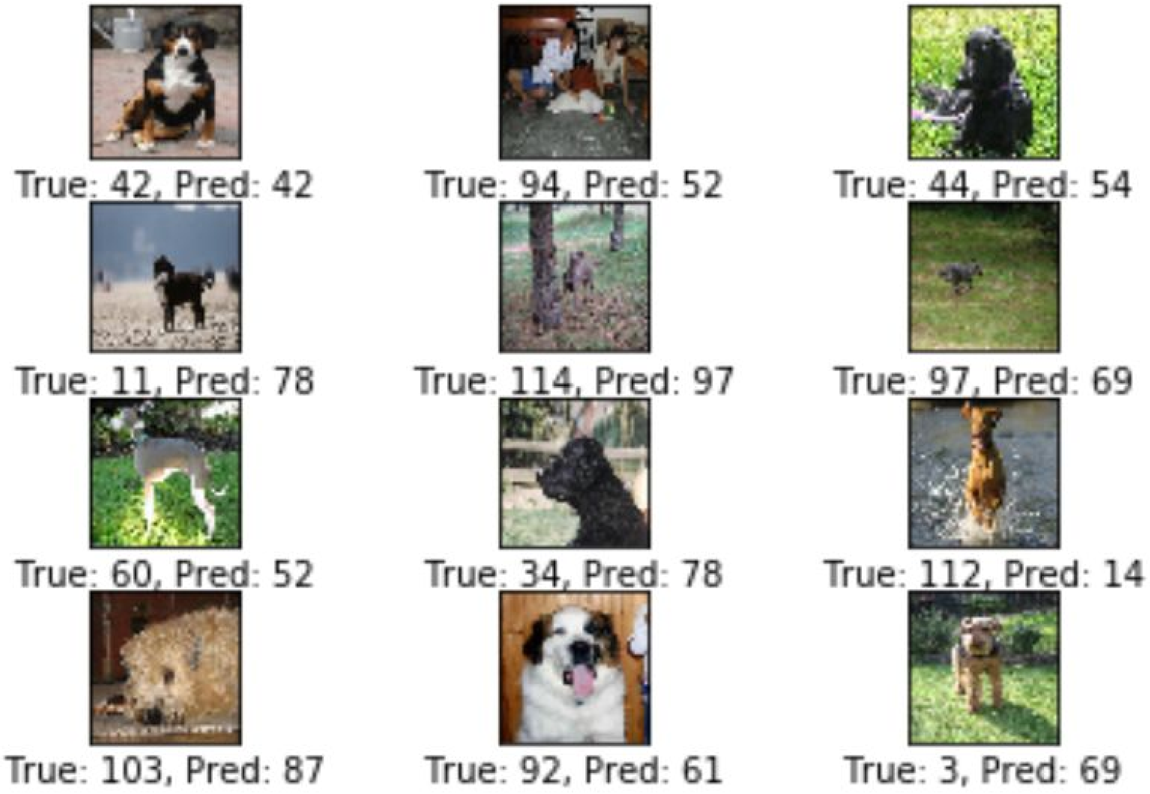


```
Iteration:    250, Training Accuracy:  36.0%, Validation Accuracy:  35.4%
Iteration:    500, Training Accuracy:  58.0%, Validation Accuracy:  39.0%
Iteration:    750, Training Accuracy:  60.0%, Validation Accuracy:  43.3%
Iteration:   1000, Training Accuracy:  56.0%, Validation Accuracy:  43.3%
Iteration:   1250, Training Accuracy:  70.0%, Validation Accuracy:  43.3%
Iteration:   1500, Training Accuracy:  84.0%, Validation Accuracy:  43.7%
Iteration:   1750, Training Accuracy:  90.0%, Validation Accuracy:  44.0%
Iteration:   2000, Training Accuracy:  82.0%, Validation Accuracy:  43.0%
Iteration:   2250, Training Accuracy:  94.0%, Validation Accuracy:  44.0%
Iteration:   2500, Training Accuracy:  90.0%, Validation Accuracy:  43.7%
Iteration:   2750, Training Accuracy:  88.0%, Validation Accuracy:  43.7%
Iteration:   3000, Training Accuracy:  92.0%, Validation Accuracy:  40.8%
Iteration:   3250, Training Accuracy:  94.0%, Validation Accuracy:  43.0%
Iteration:   3500, Training Accuracy:  94.0%, Validation Accuracy:  40.8%
Time usage: 0:02:01
```

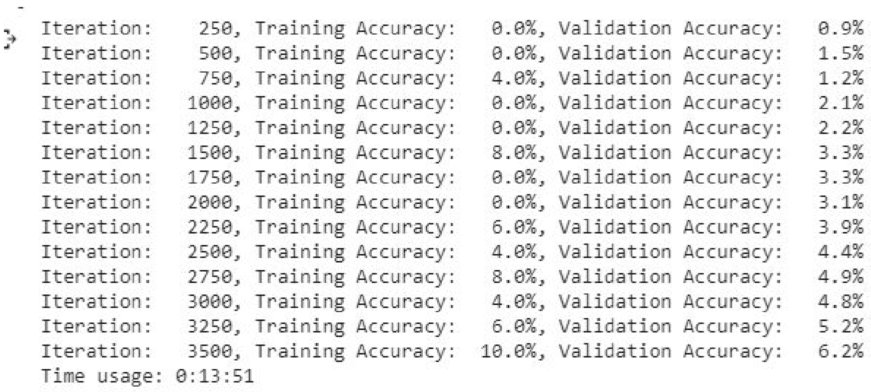
Tensorflow with 8 breeds (Stanford Dog Dataset):



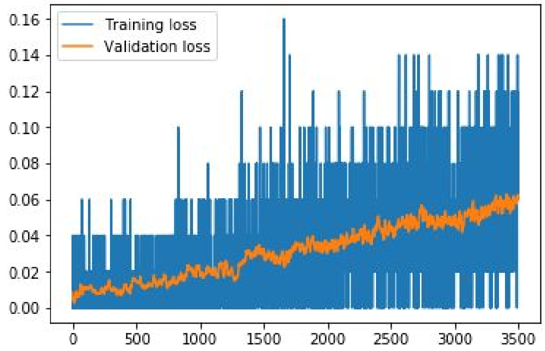
Tensorflow with 120 breeds (Stanford Dog Dataset):



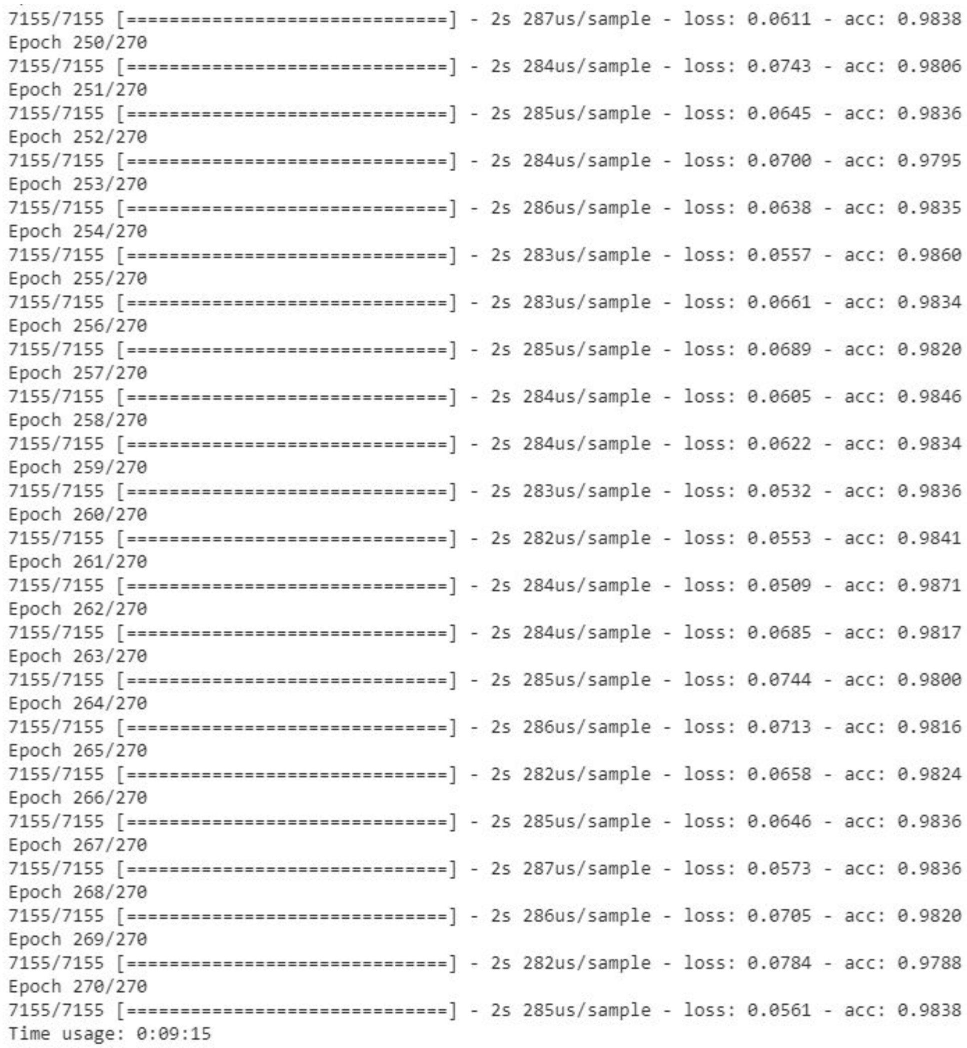
Tensorflow with 120 Breeds - GPU:



```
Iteration:    250, Training Accuracy:   0.0%, Validation Accuracy:   0.9%
Iteration:    500, Training Accuracy:   0.0%, Validation Accuracy:   1.5%
Iteration:    750, Training Accuracy:   4.0%, Validation Accuracy:   1.2%
Iteration:   1000, Training Accuracy:   0.0%, Validation Accuracy:   2.1%
Iteration:   1250, Training Accuracy:   0.0%, Validation Accuracy:   2.2%
Iteration:   1500, Training Accuracy:   8.0%, Validation Accuracy:   3.3%
Iteration:   1750, Training Accuracy:   0.0%, Validation Accuracy:   3.3%
Iteration:   2000, Training Accuracy:   0.0%, Validation Accuracy:   3.1%
Iteration:   2250, Training Accuracy:   6.0%, Validation Accuracy:   3.9%
Iteration:   2500, Training Accuracy:   4.0%, Validation Accuracy:   4.4%
Iteration:   2750, Training Accuracy:   8.0%, Validation Accuracy:   4.9%
Iteration:   3000, Training Accuracy:   4.0%, Validation Accuracy:   4.8%
Iteration:   3250, Training Accuracy:   6.0%, Validation Accuracy:   5.2%
Iteration:   3500, Training Accuracy:  10.0%, Validation Accuracy:   6.2%
Time usage: 0:13:51
```

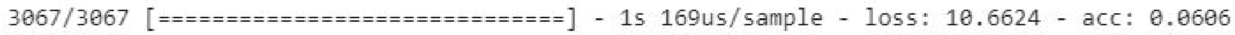


Keras with 120 breeds - GPU:



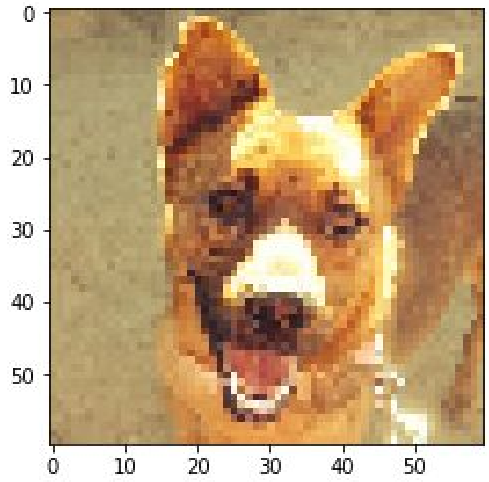
```
7155/7155 [==============================] - 2s 287us/sample - loss: 0.0611 - acc: 0.9838
Epoch 250/270
7155/7155 [==============================] - 2s 284us/sample - loss: 0.0743 - acc: 0.9806
Epoch 251/270
7155/7155 [==============================] - 2s 285us/sample - loss: 0.0645 - acc: 0.9836
Epoch 252/270
7155/7155 [==============================] - 2s 284us/sample - loss: 0.0700 - acc: 0.9795
Epoch 253/270
7155/7155 [==============================] - 2s 286us/sample - loss: 0.0638 - acc: 0.9835
Epoch 254/270
7155/7155 [==============================] - 2s 283us/sample - loss: 0.0557 - acc: 0.9860
Epoch 255/270
7155/7155 [==============================] - 2s 283us/sample - loss: 0.0661 - acc: 0.9834
Epoch 256/270
7155/7155 [==============================] - 2s 285us/sample - loss: 0.0689 - acc: 0.9820
Epoch 257/270
7155/7155 [==============================] - 2s 284us/sample - loss: 0.0605 - acc: 0.9846
Epoch 258/270
7155/7155 [==============================] - 2s 284us/sample - loss: 0.0622 - acc: 0.9834
Epoch 259/270
7155/7155 [==============================] - 2s 283us/sample - loss: 0.0532 - acc: 0.9836
Epoch 260/270
7155/7155 [==============================] - 2s 282us/sample - loss: 0.0553 - acc: 0.9841
Epoch 261/270
7155/7155 [==============================] - 2s 284us/sample - loss: 0.0509 - acc: 0.9871
Epoch 262/270
7155/7155 [==============================] - 2s 284us/sample - loss: 0.0685 - acc: 0.9817
Epoch 263/270
7155/7155 [==============================] - 2s 285us/sample - loss: 0.0744 - acc: 0.9800
Epoch 264/270
7155/7155 [==============================] - 2s 286us/sample - loss: 0.0713 - acc: 0.9816
Epoch 265/270
7155/7155 [==============================] - 2s 282us/sample - loss: 0.0658 - acc: 0.9824
Epoch 266/270
7155/7155 [==============================] - 2s 285us/sample - loss: 0.0646 - acc: 0.9836
Epoch 267/270
7155/7155 [==============================] - 2s 287us/sample - loss: 0.0573 - acc: 0.9836
Epoch 268/270
7155/7155 [==============================] - 2s 286us/sample - loss: 0.0705 - acc: 0.9820
Epoch 269/270
7155/7155 [==============================] - 2s 282us/sample - loss: 0.0784 - acc: 0.9788
Epoch 270/270
7155/7155 [==============================] - 2s 285us/sample - loss: 0.0561 - acc: 0.9838
Time usage: 0:09:15
```

Validation accuracy:

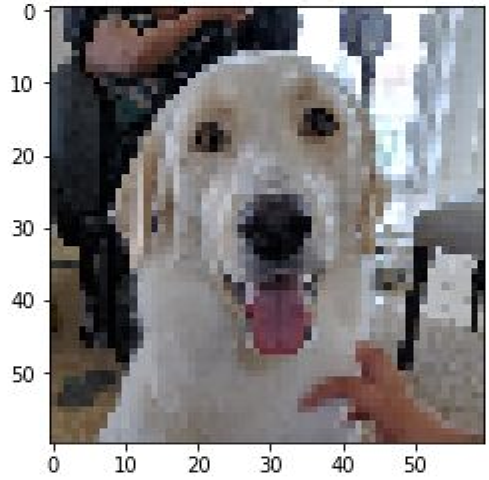


```
3067/3067 [==============================] - 1s 169us/sample - loss: 10.6624 - acc: 0.0606
```

Prediction (Our own dog pics):

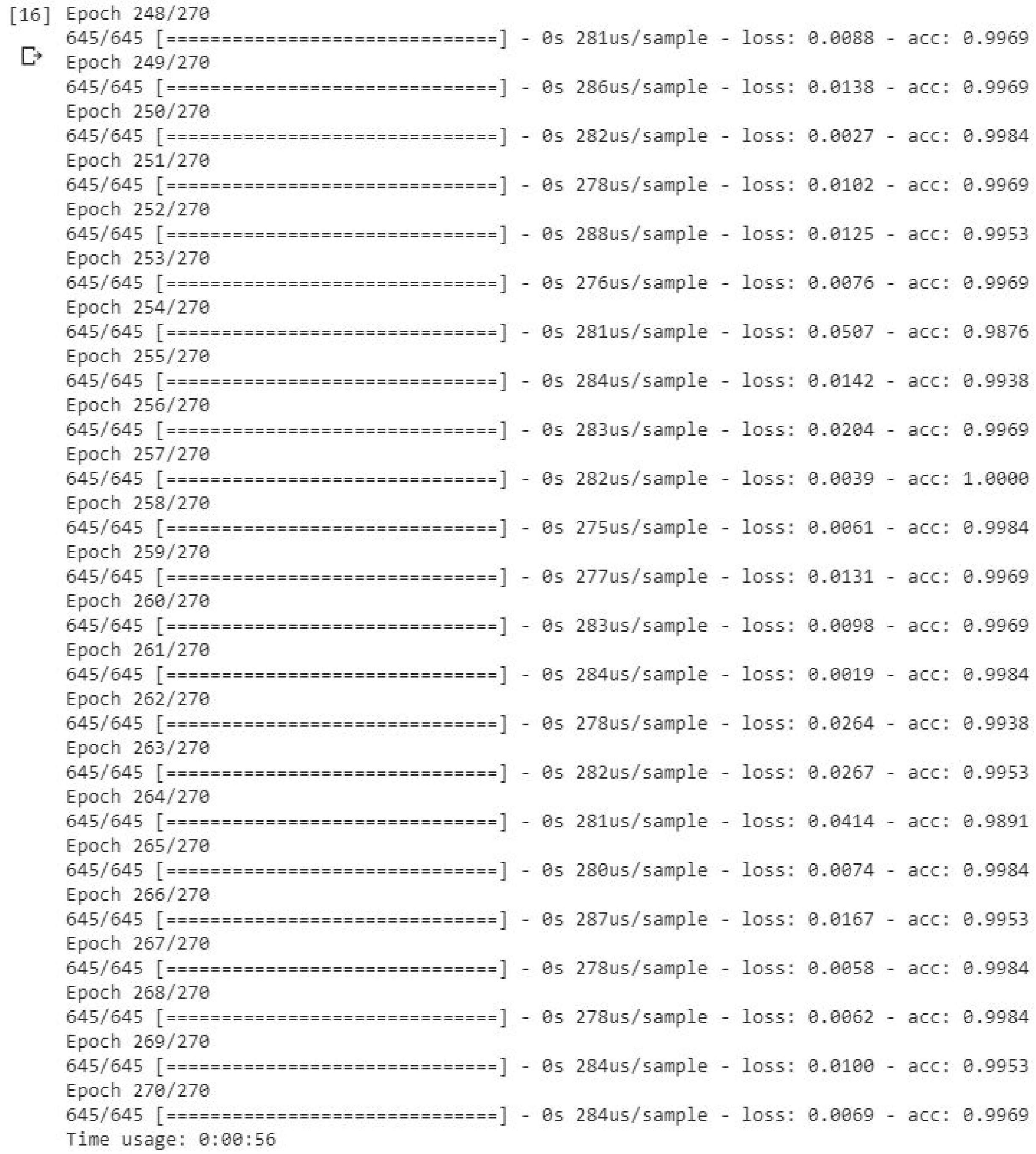


```
dandie_dinmont
```



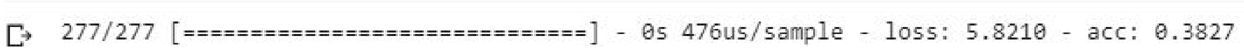
```
giant_schnauzer
```

Keras with 8 breeds - GPU:



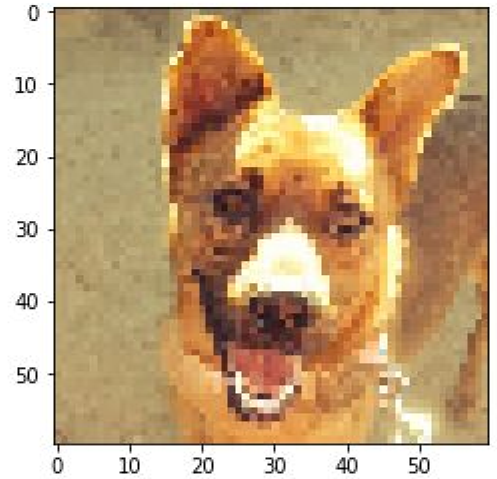
```
[16] Epoch 248/270
     645/645 [==============================] - 0s 281us/sample - loss: 0.0088 - acc: 0.9969
     Epoch 249/270
     645/645 [==============================] - 0s 286us/sample - loss: 0.0138 - acc: 0.9969
     Epoch 250/270
     645/645 [==============================] - 0s 282us/sample - loss: 0.0027 - acc: 0.9984
     Epoch 251/270
     645/645 [==============================] - 0s 278us/sample - loss: 0.0102 - acc: 0.9969
     Epoch 252/270
     645/645 [==============================] - 0s 288us/sample - loss: 0.0125 - acc: 0.9953
     Epoch 253/270
     645/645 [==============================] - 0s 276us/sample - loss: 0.0076 - acc: 0.9969
     Epoch 254/270
     645/645 [==============================] - 0s 281us/sample - loss: 0.0507 - acc: 0.9876
     Epoch 255/270
     645/645 [==============================] - 0s 284us/sample - loss: 0.0142 - acc: 0.9938
     Epoch 256/270
     645/645 [==============================] - 0s 283us/sample - loss: 0.0204 - acc: 0.9969
     Epoch 257/270
     645/645 [==============================] - 0s 282us/sample - loss: 0.0039 - acc: 1.0000
     Epoch 258/270
     645/645 [==============================] - 0s 275us/sample - loss: 0.0061 - acc: 0.9984
     Epoch 259/270
     645/645 [==============================] - 0s 277us/sample - loss: 0.0131 - acc: 0.9969
     Epoch 260/270
     645/645 [==============================] - 0s 283us/sample - loss: 0.0098 - acc: 0.9969
     Epoch 261/270
     645/645 [==============================] - 0s 284us/sample - loss: 0.0019 - acc: 0.9984
     Epoch 262/270
     645/645 [==============================] - 0s 278us/sample - loss: 0.0264 - acc: 0.9938
     Epoch 263/270
     645/645 [==============================] - 0s 282us/sample - loss: 0.0267 - acc: 0.9953
     Epoch 264/270
     645/645 [==============================] - 0s 281us/sample - loss: 0.0414 - acc: 0.9891
     Epoch 265/270
     645/645 [==============================] - 0s 280us/sample - loss: 0.0074 - acc: 0.9984
     Epoch 266/270
     645/645 [==============================] - 0s 287us/sample - loss: 0.0167 - acc: 0.9953
     Epoch 267/270
     645/645 [==============================] - 0s 278us/sample - loss: 0.0058 - acc: 0.9984
     Epoch 268/270
     645/645 [==============================] - 0s 278us/sample - loss: 0.0062 - acc: 0.9984
     Epoch 269/270
     645/645 [==============================] - 0s 284us/sample - loss: 0.0100 - acc: 0.9953
     Epoch 270/270
     645/645 [==============================] - 0s 284us/sample - loss: 0.0069 - acc: 0.9969
     Time usage: 0:00:56
```

Validation Accuracy:

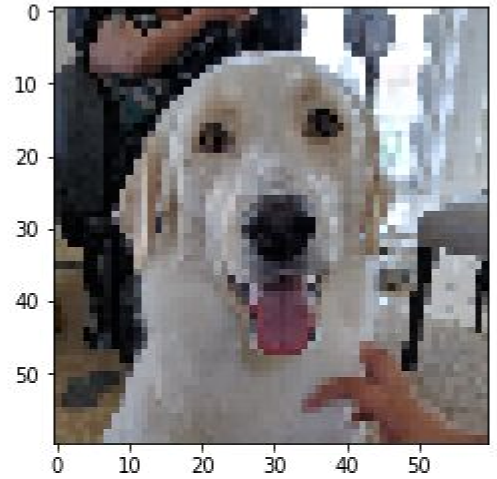


```
277/277 [==============================] - 0s 476us/sample - loss: 5.8210 - acc: 0.3827
```

Prediction (Our own dog pics):

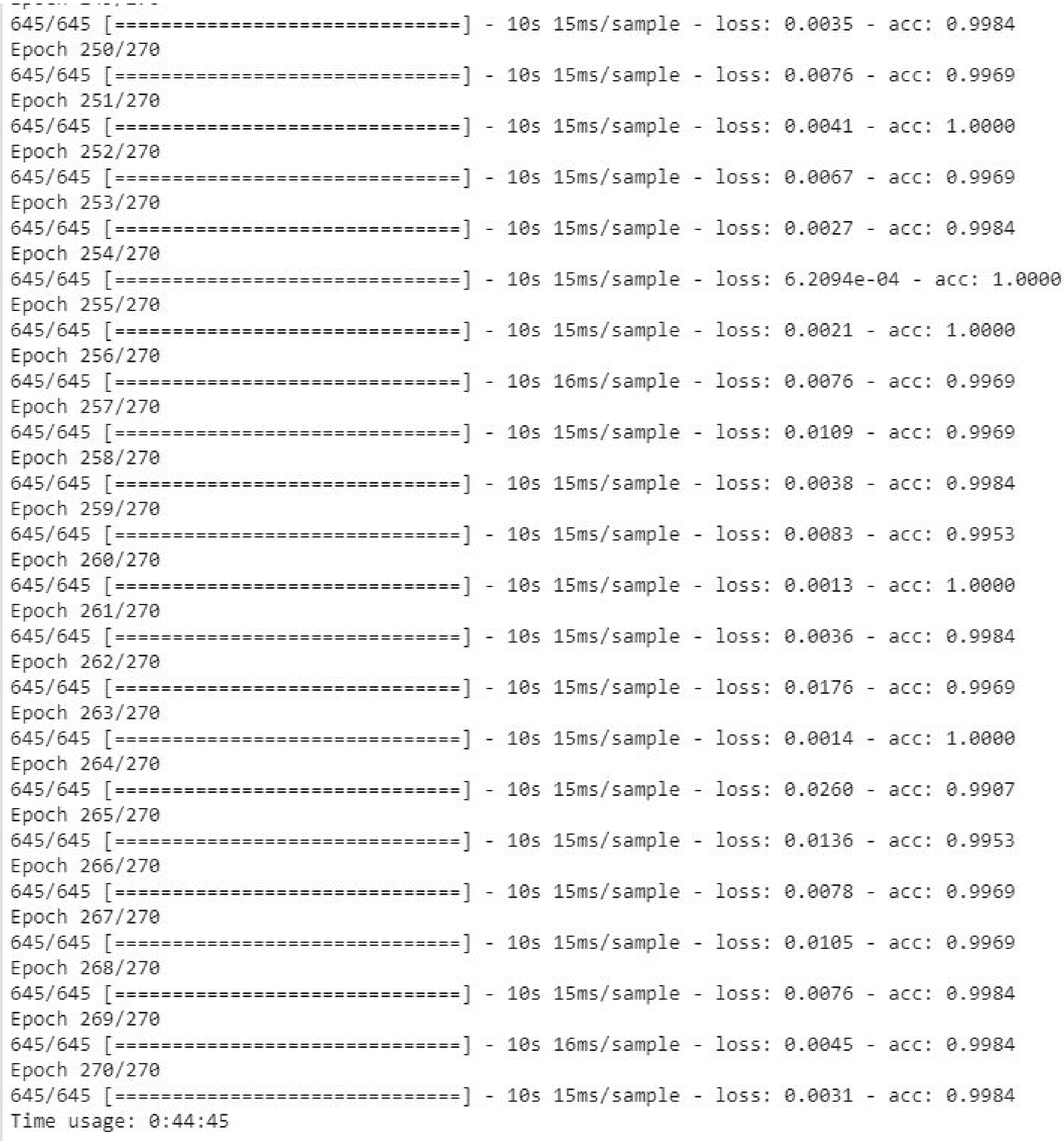


```
shih-tzu
```



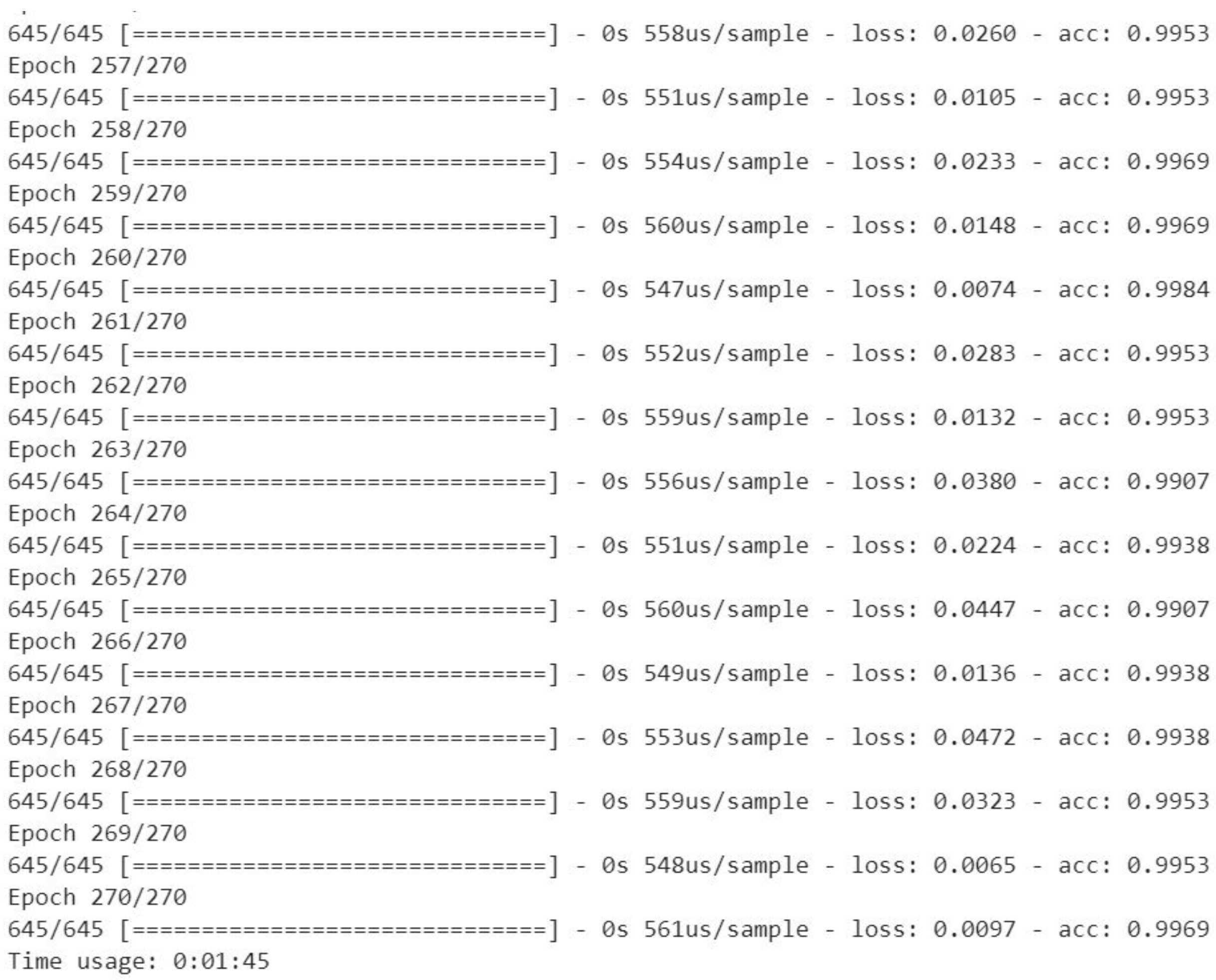
```
scottish_deerhound
```

Keras with 8 breeds - CPU:



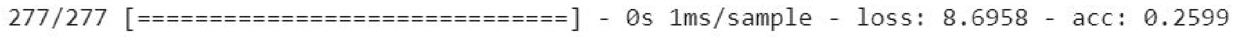
```
Epoch 249/270
645/645 [==============================] - 10s 15ms/sample - loss: 0.0035 - acc: 0.9984
Epoch 250/270
645/645 [==============================] - 10s 15ms/sample - loss: 0.0076 - acc: 0.9969
Epoch 251/270
645/645 [==============================] - 10s 15ms/sample - loss: 0.0041 - acc: 1.0000
Epoch 252/270
645/645 [==============================] - 10s 15ms/sample - loss: 0.0067 - acc: 0.9969
Epoch 253/270
645/645 [==============================] - 10s 15ms/sample - loss: 0.0027 - acc: 0.9984
Epoch 254/270
645/645 [==============================] - 10s 15ms/sample - loss: 6.2094e-04 - acc: 1.0000
Epoch 255/270
645/645 [==============================] - 10s 15ms/sample - loss: 0.0021 - acc: 1.0000
Epoch 256/270
645/645 [==============================] - 10s 16ms/sample - loss: 0.0076 - acc: 0.9969
Epoch 257/270
645/645 [==============================] - 10s 15ms/sample - loss: 0.0109 - acc: 0.9969
Epoch 258/270
645/645 [==============================] - 10s 15ms/sample - loss: 0.0038 - acc: 0.9984
Epoch 259/270
645/645 [==============================] - 10s 15ms/sample - loss: 0.0083 - acc: 0.9953
Epoch 260/270
645/645 [==============================] - 10s 15ms/sample - loss: 0.0013 - acc: 1.0000
Epoch 261/270
645/645 [==============================] - 10s 15ms/sample - loss: 0.0036 - acc: 0.9984
Epoch 262/270
645/645 [==============================] - 10s 15ms/sample - loss: 0.0176 - acc: 0.9969
Epoch 263/270
645/645 [==============================] - 10s 15ms/sample - loss: 0.0014 - acc: 1.0000
Epoch 264/270
645/645 [==============================] - 10s 15ms/sample - loss: 0.0260 - acc: 0.9907
Epoch 265/270
645/645 [==============================] - 10s 15ms/sample - loss: 0.0136 - acc: 0.9953
Epoch 266/270
645/645 [==============================] - 10s 15ms/sample - loss: 0.0078 - acc: 0.9969
Epoch 267/270
645/645 [==============================] - 10s 15ms/sample - loss: 0.0105 - acc: 0.9969
Epoch 268/270
645/645 [==============================] - 10s 15ms/sample - loss: 0.0076 - acc: 0.9984
Epoch 269/270
645/645 [==============================] - 10s 16ms/sample - loss: 0.0045 - acc: 0.9984
Epoch 270/270
645/645 [==============================] - 10s 15ms/sample - loss: 0.0031 - acc: 0.9984
Time usage: 0:44:45
```

Keras with 120 breeds using larger filter sizes (10, 15, 20) - GPU:



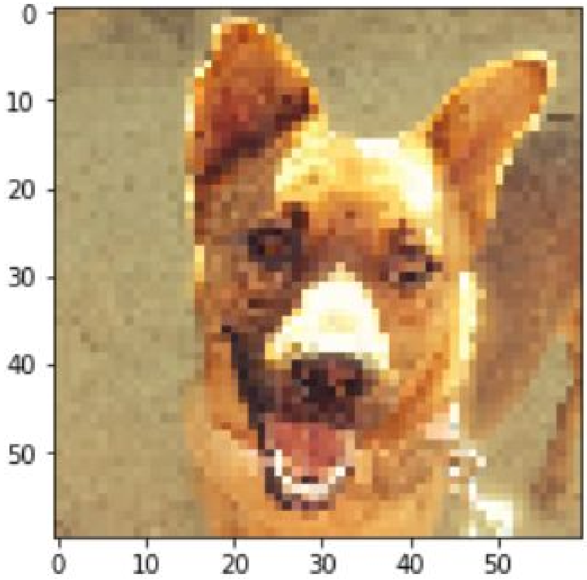
```
645/645 [==============================] - 0s 558us/sample - loss: 0.0260 - acc: 0.9953
Epoch 257/270
645/645 [==============================] - 0s 551us/sample - loss: 0.0105 - acc: 0.9953
Epoch 258/270
645/645 [==============================] - 0s 554us/sample - loss: 0.0233 - acc: 0.9969
Epoch 259/270
645/645 [==============================] - 0s 560us/sample - loss: 0.0148 - acc: 0.9969
Epoch 260/270
645/645 [==============================] - 0s 547us/sample - loss: 0.0074 - acc: 0.9984
Epoch 261/270
645/645 [==============================] - 0s 552us/sample - loss: 0.0283 - acc: 0.9953
Epoch 262/270
645/645 [==============================] - 0s 559us/sample - loss: 0.0132 - acc: 0.9953
Epoch 263/270
645/645 [==============================] - 0s 556us/sample - loss: 0.0380 - acc: 0.9907
Epoch 264/270
645/645 [==============================] - 0s 551us/sample - loss: 0.0224 - acc: 0.9938
Epoch 265/270
645/645 [==============================] - 0s 560us/sample - loss: 0.0447 - acc: 0.9907
Epoch 266/270
645/645 [==============================] - 0s 549us/sample - loss: 0.0136 - acc: 0.9938
Epoch 267/270
645/645 [==============================] - 0s 553us/sample - loss: 0.0472 - acc: 0.9938
Epoch 268/270
645/645 [==============================] - 0s 559us/sample - loss: 0.0323 - acc: 0.9953
Epoch 269/270
645/645 [==============================] - 0s 548us/sample - loss: 0.0065 - acc: 0.9953
Epoch 270/270
645/645 [==============================] - 0s 561us/sample - loss: 0.0097 - acc: 0.9969
Time usage: 0:01:45
```

Validation accuracy:

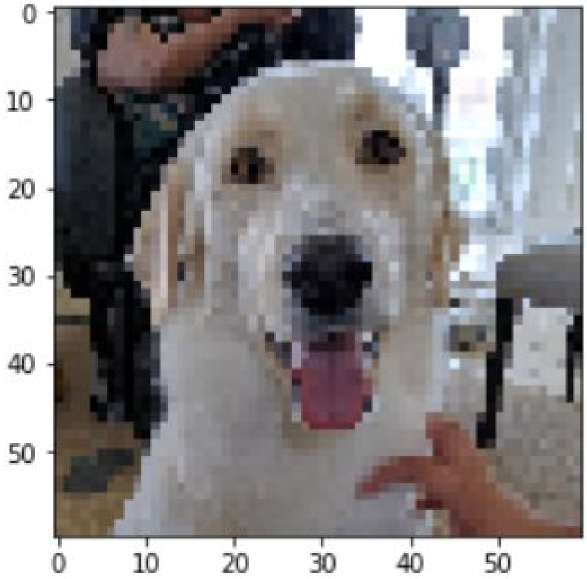


```
277/277 [==============================] - 0s 1ms/sample - loss: 8.6958 - acc: 0.2599
```

Predictions (Our own dog pics):

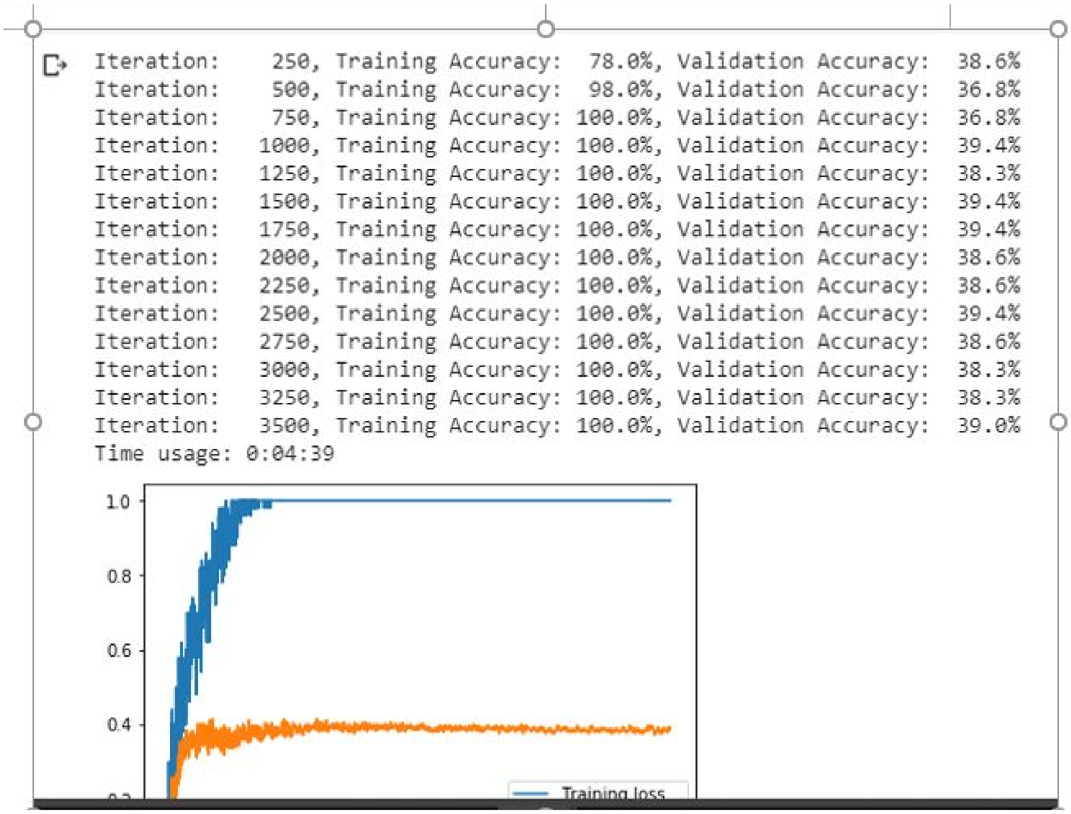


```
afghan_hound
```



```
bernese_mountain_dog
```

Tensorflow with 8 dog breeds using no dropout layer - GPU:



```
Iteration:    250, Training Accuracy:  78.0%, Validation Accuracy:  38.6%
Iteration:    500, Training Accuracy:  98.0%, Validation Accuracy:  36.8%
Iteration:    750, Training Accuracy: 100.0%, Validation Accuracy:  36.8%
Iteration:   1000, Training Accuracy: 100.0%, Validation Accuracy:  39.4%
Iteration:   1250, Training Accuracy: 100.0%, Validation Accuracy:  38.3%
Iteration:   1500, Training Accuracy: 100.0%, Validation Accuracy:  39.4%
Iteration:   1750, Training Accuracy: 100.0%, Validation Accuracy:  39.4%
Iteration:   2000, Training Accuracy: 100.0%, Validation Accuracy:  38.6%
Iteration:   2250, Training Accuracy: 100.0%, Validation Accuracy:  38.6%
Iteration:   2500, Training Accuracy: 100.0%, Validation Accuracy:  39.4%
Iteration:   2750, Training Accuracy: 100.0%, Validation Accuracy:  38.6%
Iteration:   3000, Training Accuracy: 100.0%, Validation Accuracy:  38.3%
Iteration:   3250, Training Accuracy: 100.0%, Validation Accuracy:  38.3%
Iteration:   3500, Training Accuracy: 100.0%, Validation Accuracy:  39.0%
Time usage: 0:04:39
```

Keras with Transfer Learning with 120 breeds - GPU:

We only got code for Keras transfer learning to work once. It would not compile again after that. The Accuracy was around 95% and it took around the same time as Keras with 120 breeds.

Summary of Results:

- Using GPU is faster than CPU
- Both training and validation accuracy drops when more dog breeds are added while using Tensorflow (due to limited pictures per breed)
- Both training and validation accuracy drops when more dog breeds are added while using Keras (due to limited pictures per breed)
- Using larger filter sizes increases the accuracy of prediction while using Keras
- Using no dropout layer while using Tensorflow overfits the data
- Using transfer learning with Keras had a much better accuracy

Challenges Faced

- One of our issues was figuring out how to preprocess the data for the using Keras implementation so that we could leverage Keras' functionality.
- We also had difficulties figuring out how to use personal pictures of our own dogs to test our model.
- Implementing a CNN from scratch using only Tensorflow was challenging.

Task Breakdown

Task	Breakdown	
Tensorflow	Taz: 50%	Melissa: 50%
Keras	Taz: 50%	Melissa: 50%
Keras Transfer Learning	Taz: 50%	Melissa: 50%
Report	Taz: 50%	Melissa: 50%
Presentation	Taz: 50%	Melissa: 50%

References

Transfer Learning -
https://towardsdatascience.com/dog-breed-classification-using-cnns-f042fbe0f333
https://towardsdatascience.com/transfer-learning-for-image-classification-using-keras-c47ccf09c8c8

TensorFlow Dog Breed Classification Tutorial -
https://www.kaggle.com/kaggleslayer/simple-convolutional-n-network-with-tensorflow

Keras Documentation - https://keras.io/