1. What is TensorFlow? Which company is the leading contributor to TensorFlow?

   Taken straight from the Tensorflow website, “Tensorflow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML-powered applications”. The company that is the leading contributor to TensorFlow is Google and their Brain Team.

2. What is TensorRT? How is it different from TensorFlow?

   TensorRT is like the 5G of TensorFlow. It is an SDK for high-performance deep learning inference. It includes a deep learning inference optimizer and runtime that delivers low-latency and high throughput for deep learning inference applications. TensorRT was developed by NVIDIA and was developed using CUDA. It allows the user to import trained models from every deep learning network into it, and after applying optimizations, select platform-specific kernels to use that will maximize performance on GPUs allowing the user to focus more on the application creation and less on the model optimization.

3. What is ImageNet? How many images does it contain? How many classes?

   ImageNet is a giant database of more than fourteen million images that have been looked at by humans and annotated to say which objects are in which pictures and according to Wikipedia, there are at least one million of the images that also contain bounding boxes. There are 20,000 categories and 200 classes.

4. Please research and explain the differences between MobileNet and GoogleNet (Inception) architectures.

   MobileNet and GoogleNet architectures are types of convolutional neural networks, which are designed to recognize visual patterns directly from pixel images with minimal preprocessing. GoogleNet, aka Inception V1, came first and is a CNN that is 22-layers deep that reduced the number of parameters from 60 million to 4 million. It was the winner of the ILSVRC (ImageNet Large Scale Visual Recognition Challenge) competition in 2014 and managed to achieve an error rate of only 6.67%, which was very close to the human-level performance. The network uses a CNN inspired by the 1998 LeNet which was built to recognize digits on bank checks but implemented something referred to as an inception module, which uses batch normalization, image distortions, and

RMSprop. The inception module is based on very small convolutions done to reduce the number of parameters on a drastic scale.

MobileNet also uses a CNN architecture model for Image Classification but is built for Mobile Vision. The architecture is based on a streamlined version that uses depthwise separable convolutions to build lightweight deep neural networks; this is useful for mobile applications. The architecture introduces two simple global hyperparameters that efficiently trade off between latency and accuracy. The width and resolution can also be tuned to trade off between latency and accuracy. Finally, the architecture uses depthwise separable convolution filters as its network structure. DSC filters apply a single filter to each input channel. A pointwise convolution is then performed; this convolution applies a 1x11x1 convolution to combine the outputs of the DSC.

5. In your own words, what is a bottleneck?

   A bottleneck is a point in a process that does not allow pieces in the process to continue flowing at the same rate as the other points. For example if there are five steps in a pb&j assembly line, but the step add peanut butter takes 30 seconds, compared to all other steps which only take 10 seconds, there will be a lot of sandwiches waiting to be peanut buttered. In our case, a bottleneck would be a step in our process that does not compute or process as fast as the steps that feed into it.

6. How is a bottleneck different from the concept of layer freezing?

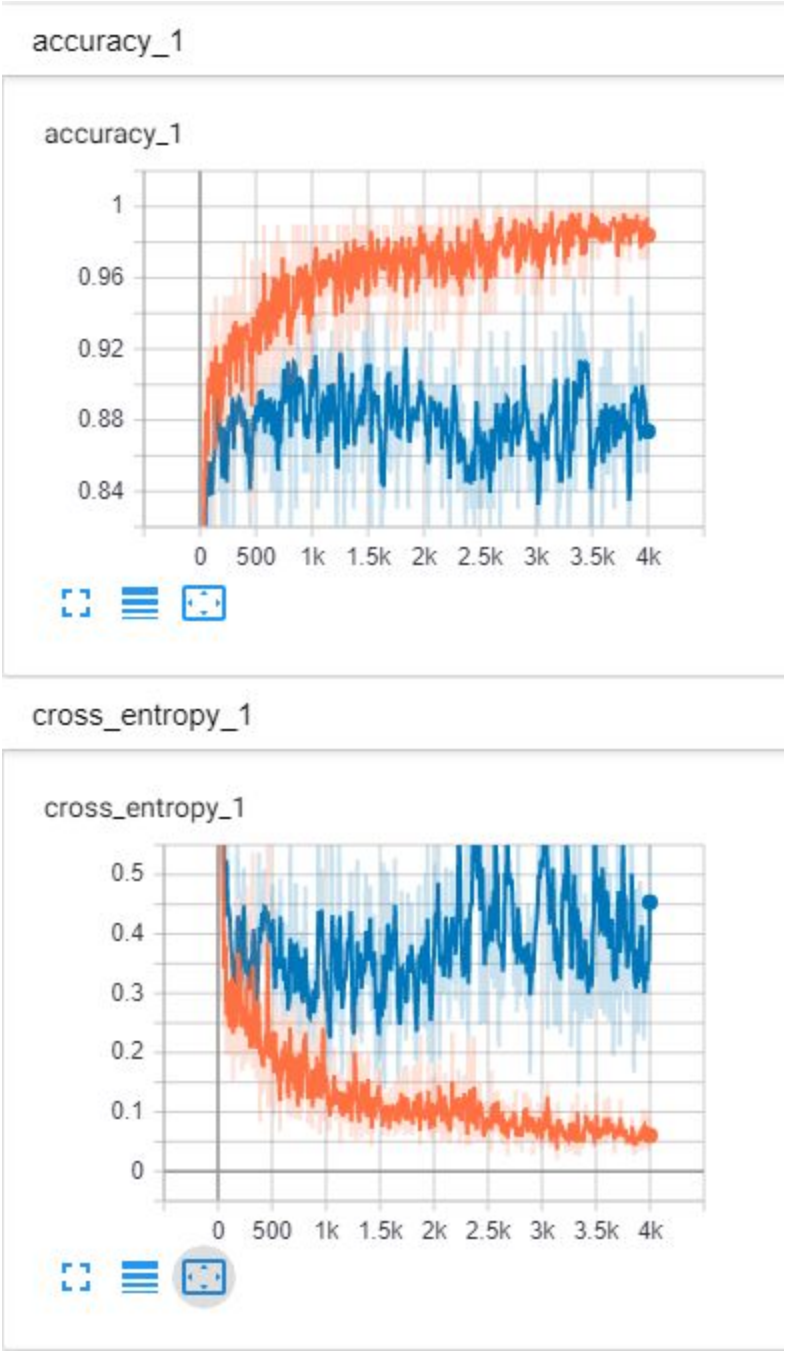
   Layer freezing is a technique used in neural networks that controls the way the weights are updated. When a layer is frozen, it means that its weights cannot be modified any further. The technique is done in order to cut down on computation time without giving up too much on the side of accuracy. A bottleneck is different in that it does not occur by choice, whereas layer freezing is.

7. In the TF1 lab, you trained the last layer (all the previous layers retain their already-trained state). Explain how the lab used the previous layers (where did they come from? how were they used in the process?)

   The previous layers were already built into the pre-trained model which were specified as parameters when we kicked off the model training session. These pre-trained layers are already very good at finding and summarizing the information they receive into image classification. At each step, 10 images are selected from the training dataset, their bottleneck cache files are found and fed

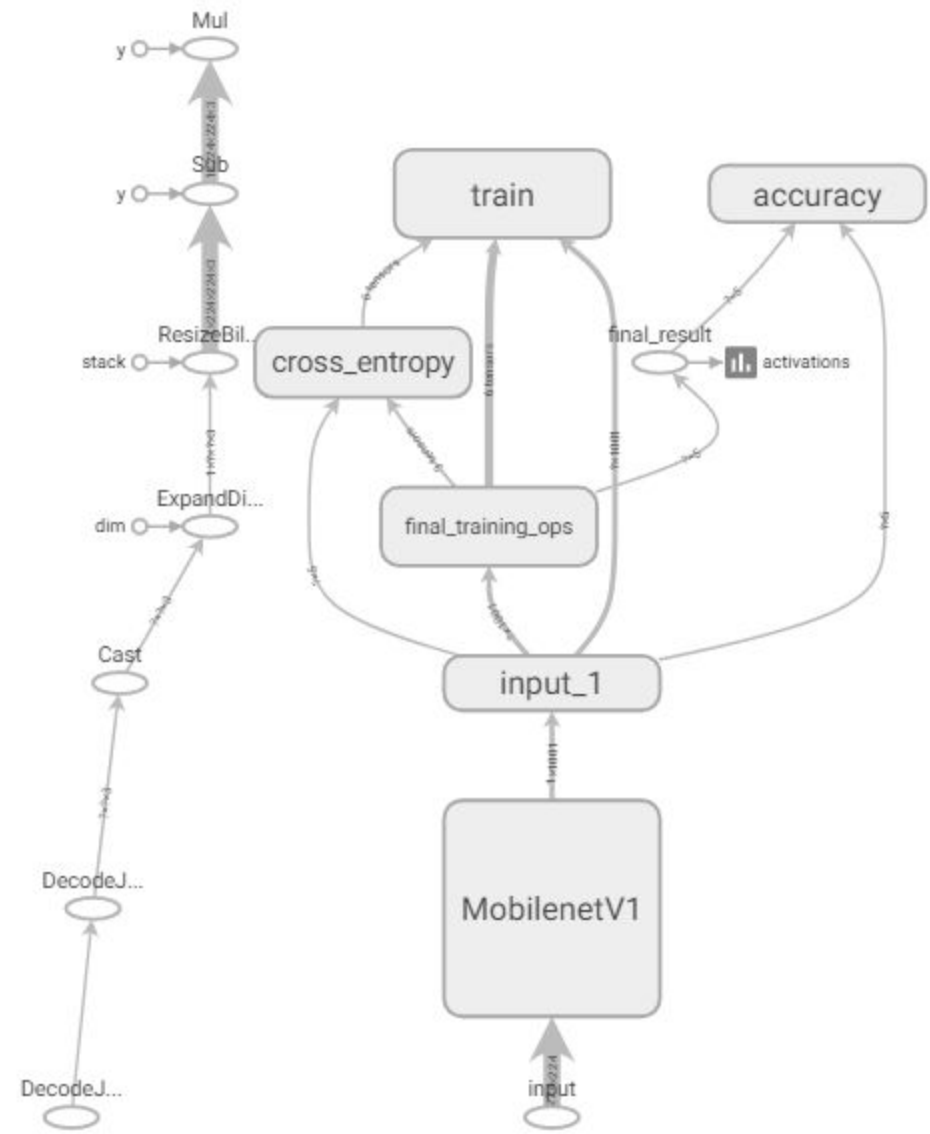
into the final layer to get a prediction. The predictions are compared to the actuals and the final layer's weights are updated using backpropagation.

Graphs after running 4000 steps (for personal reference):

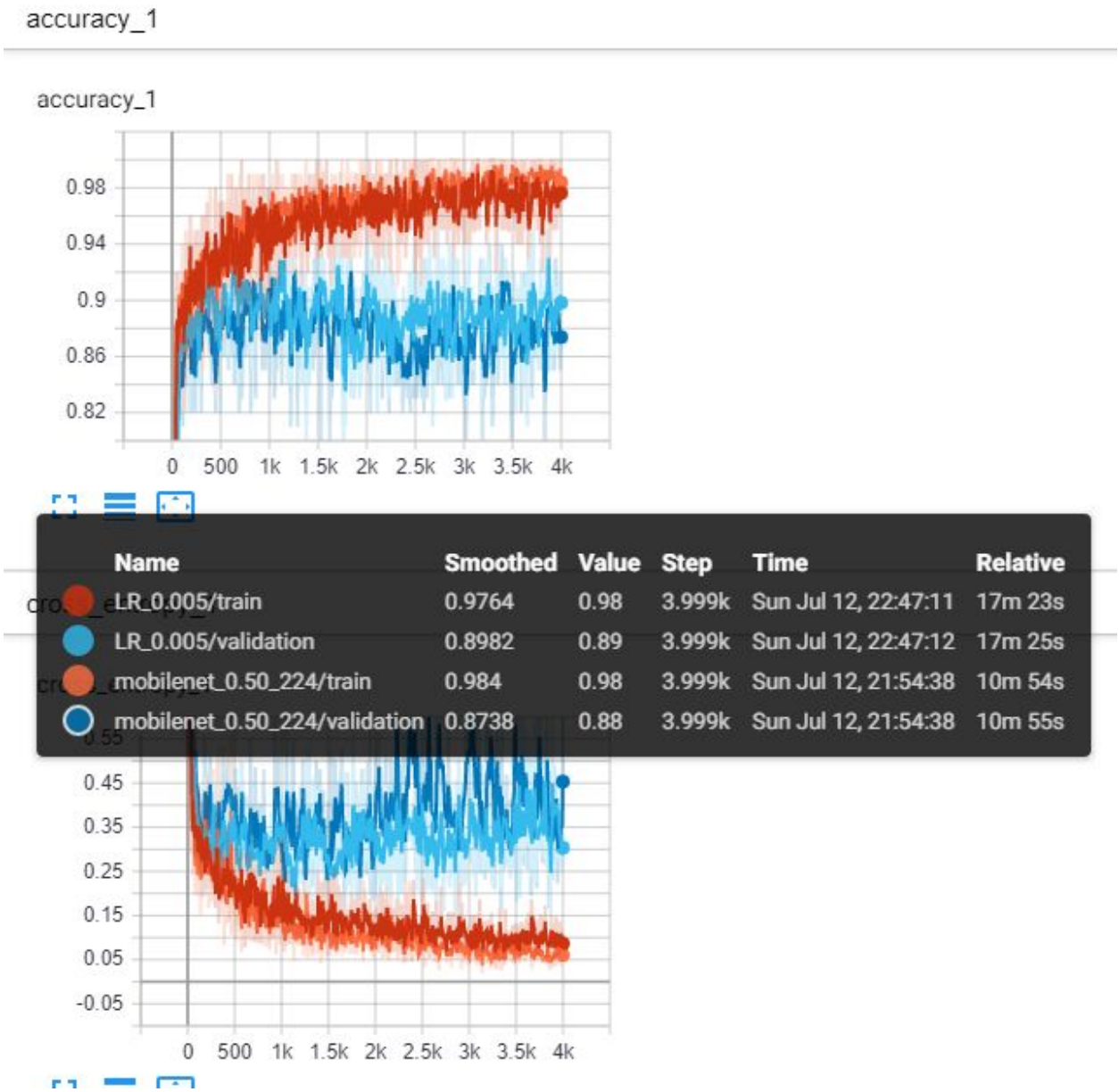


Accuracy was 0.8738 and cross_entropy was 0.4534 for the training set.

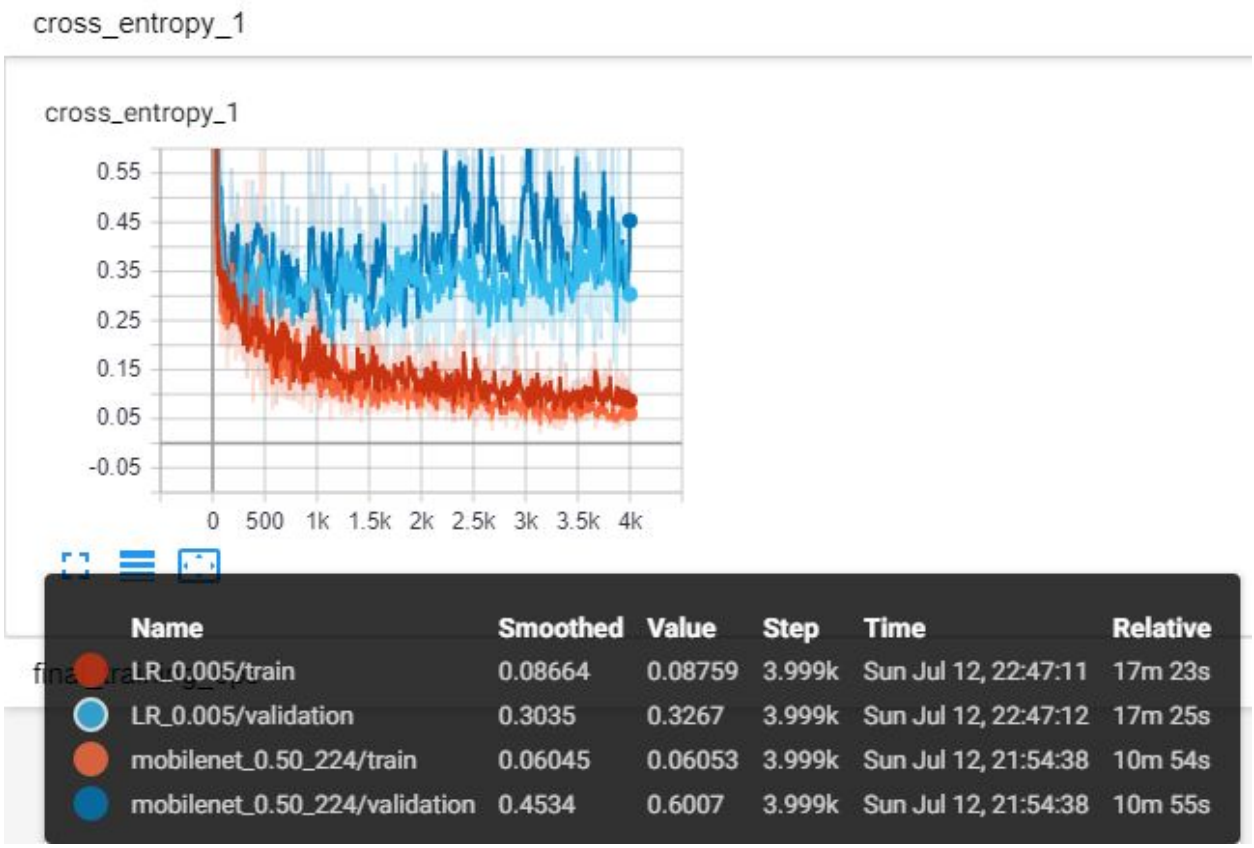

DISTRIBUTIONS
HISTOGRAMS
INACTIVE
activations
mobilenet_0.50_224/train
activations
mobilenet_0.50_224/validation
final_training_ops
final_training_ops/Wx_plus_b/pre_activations
mobilenet_0.50_224/train
final_training_ops/Wx_plus_b/pre_activations
mobilenet_0.50_224/validation
final_training_ops/biases/summaries/histogram
mobilenet_0.50_224/train
final_training_ops/biases/summaries/histogram
mobilenet_0.50_224/validation
final_training_ops/weights/summaries/histogram
mobilenet_0.50_224/train
final_training_ops/weights/summaries/histogram
mobilenet_0.50_224/validation



8. How does a low `--learning_rate` (step 7 of TF1) value (like 0.005) affect the precision? How much longer does training take?

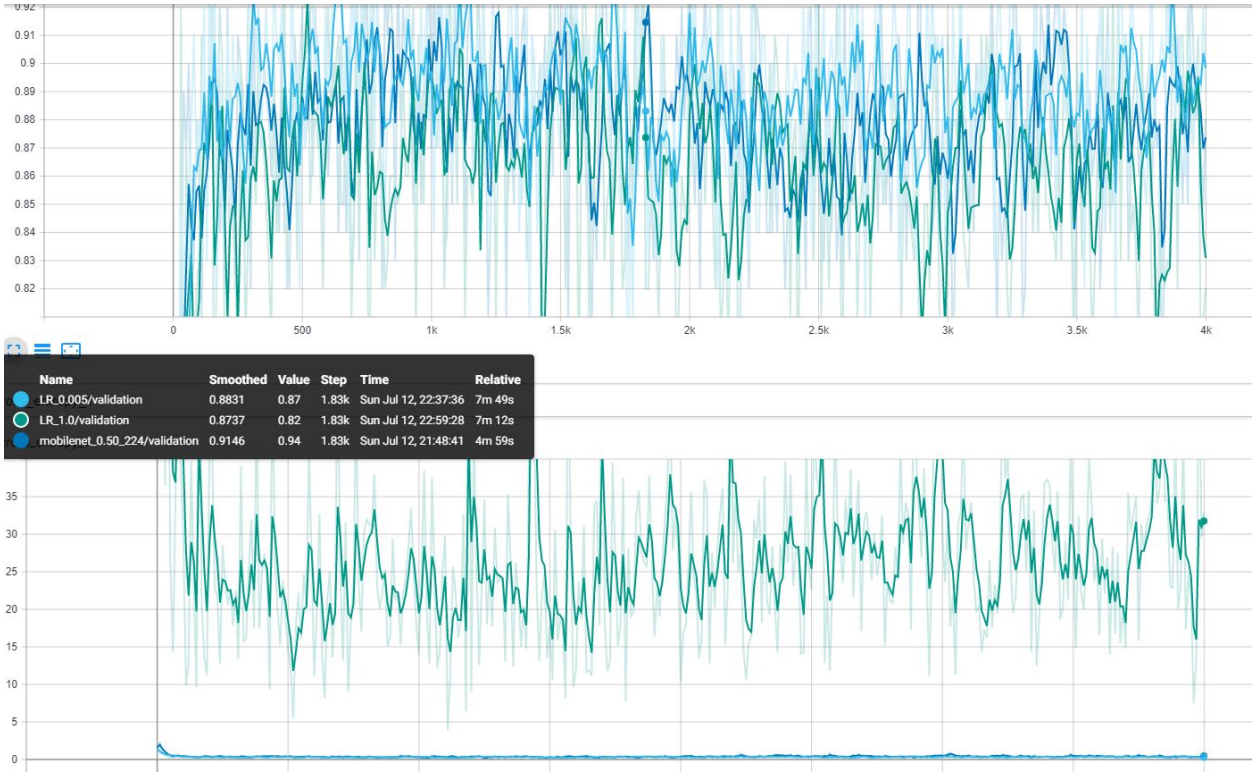


Accuracy improved about 0.02% and it took almost twice as long to train with the lowered learning rate. Cross_entropy differences are much better though - a difference of about 0.15:

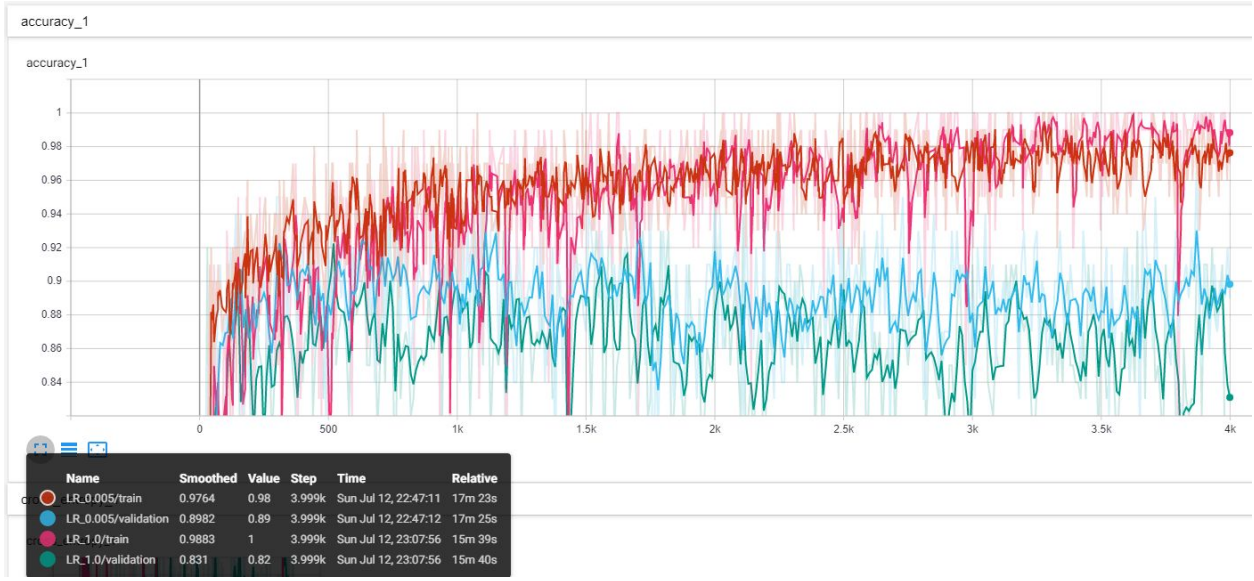


9. How about a `--learning_rate` (step 7 of TF1) of 1.0? Is the precision still good enough to produce a usable graph?

   The precision on the accuracy looks okay, but when you look at the cross_entropy, the number is way too high:



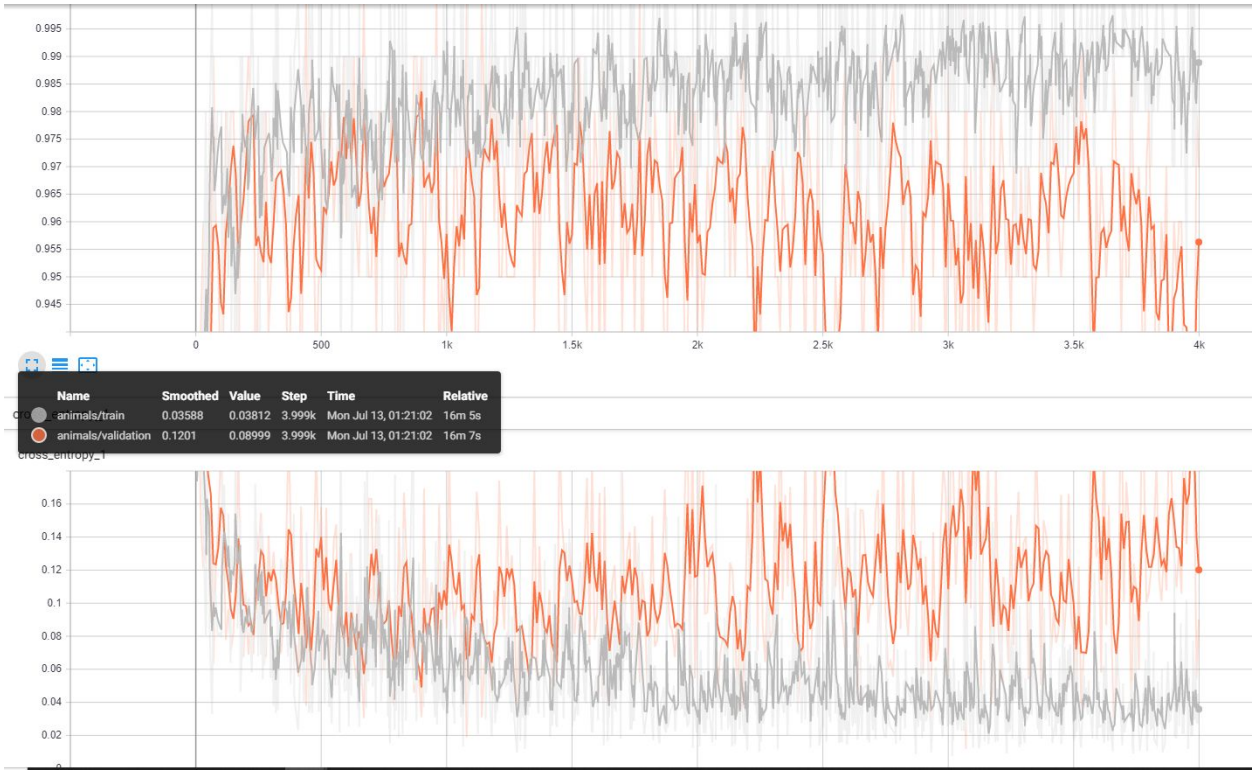
The above is comparing all three tests that were run on only the validation. The below is the accuracy for just the most recent run of learning rate = 1.0 vs LR=0.005.



Though it's a bit hard to tell from the like colors, but the graphs show the accuracy to be pretty close. However, the fact there is quite a discrepancy between the train and validation for LR=1.0; ~0.99 for the train, but only 0.83 for the validation set.

10. For step 8, you can use any images you like. Pictures of food, people, or animals work well. You can even use ImageNet images. How accurate was your model? Were you able to train it using a few images, or did you need a lot?

    Using animal image dataset found here: https://github.com/imamun93/animal-image-classifications/tree/master/data/train which is a subset of the Animal-10 dataset from a Kaggle competition. The dataset was quite large, though I suspect that the model would need a decent amount of images in order to capture all the intricacies that make up each image. Results from my training are here:



    The accuracy of the validation set was 0.9563 which is pretty good, but the cross_entropy was only 0.1201; could use some improvement. Because of the large dataset, it took about 15 minutes to create the bottlenecks and then another 15 minutes to train. I had to tweak the code to allow the gpu to grow - got a segmentation error the first time through after about 1260 steps.

11. Run the TF1 script on the CPU (see instructions above) How does the training time compare to the default network training (section 4)? Why?

    Getting this error even after attempting to pkill -f tensorboard: docker run --rm -p 6006:6006 -ti tensorflow bash

bash: docker: command not found

root@a247999b6cae:/tmp/tensorflow-for-poets-2# TensorBoard caught SIGTERM; exiting...

Assume that running the script on the CPU vs the GPU would make the script take longer, but without running through the exercise, am not actually able to compare to see if my thinking is correct.

12. Try the training again, but this time do `export ARCHITECTURE="inception_v3"` Are CPU and GPU training times different?

    See error from #11

13. Given the hints under the notes section, if we trained Inception_v3, what do we need to pass to replace ??? below to the label_image script? Can we also glean the answer from examining TensorBoard?

    I would think that we would need to change the size from where the script says =299 and =299 to =224/=224 - can see in the attributes in each step of the graph that there are some question marks on the input sizing, but the output shows the 1x224x224x3.