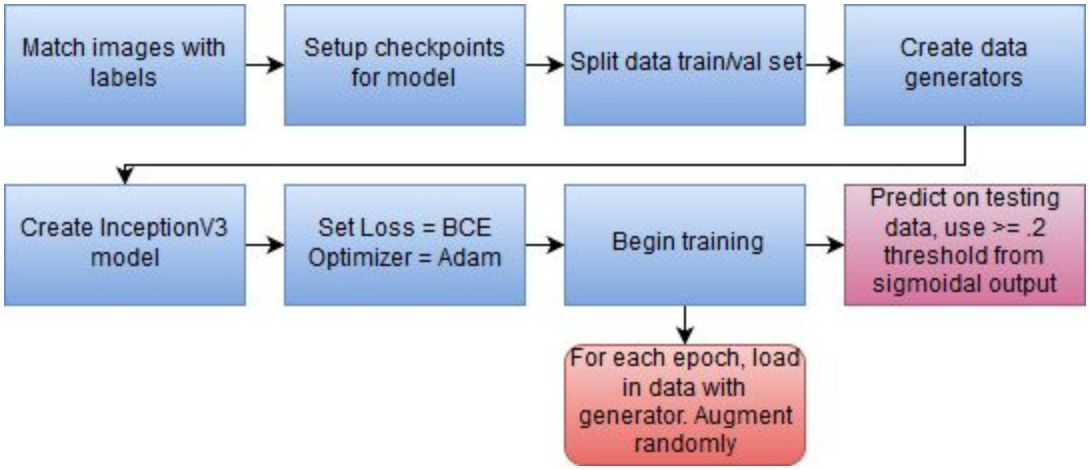
Anthony Galczak
Tristin Glunt
CS529 Project 4

We did the human protein atlas image classification Kaggle competition.

Flow Diagram



First, we load in the Kaggle data for the the 4 different samples. From there, they are stacked on top of each other into one image.

From this point, we do data augmentation on the stacked images. We do 0, 90, 180, 270 affines, horizontal flips, and vertical flips. Through experimentation, gaussian blurring, image cropping and contrast normalization did not increase accuracy.

Then, we load InceptionV3 pre-trained weights that were trained on imagenet. This pre-trained model gives a pretty solid accuracy already, but we then add 2 more convolutional layers. We did this for a few reasons. First, imagenet is trained on totally different images than proteins. Also, imagenet isn't tuned to the higher image sizes (512) that we are now using.

We finish the network with a fully connected layer that uses a sigmoid activation layer. This is important because we are doing multi-classification and thus, we don't want a softmax to "choose a winner". We need the outputs for all the classifications intact.

From this point we needed to train for a number of epochs to train the network.
We tried a few different numbers of image sizes and epochs.

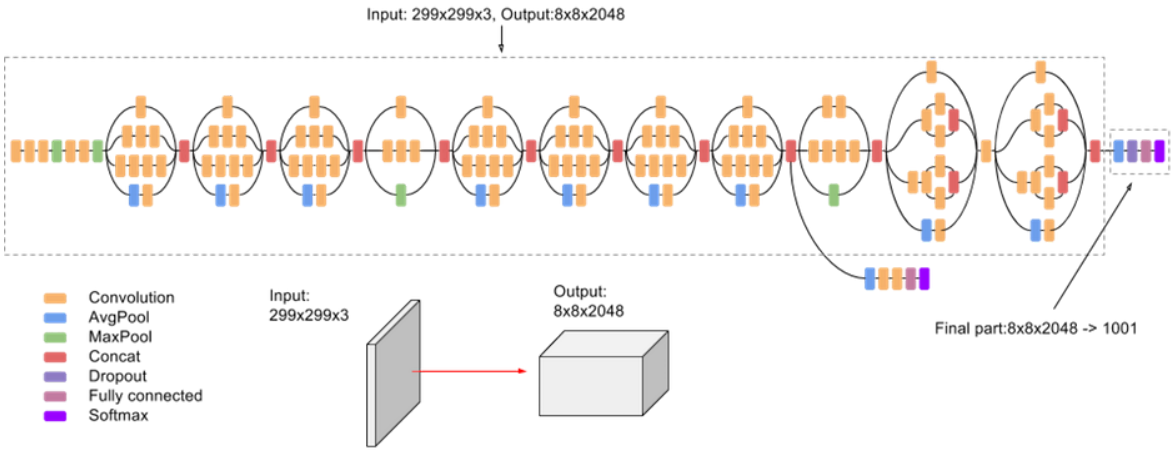
We trained these on Kaggle's Kernel runs. This was limited to 9hrs of run-time.
Image size 299, just InceptionV3, 10 epochs - .382 accuracy
Image size 299, just InceptionV3, 25 epochs - .423 accuracy
Image size 299, just InceptionV3, 45 epochs - .437 accuracy
Image size 299, InceptionV3 + 2 conv layers, 22 epochs - .437 accuracy

At this point, we needed more run-time, so we switched over to a p2.xlarge on AWS. It has the same GPU hardware (K80), but we just needed to run for longer than 9 hours.

Image size 512, InceptionV3 + 2 conv layers, 20 epochs - .457 (~20 hrs runtime)
Image size 512, InceptionV3 + 2 conv layers, 25 epochs - .463 (re-started from 20 epochs, 6hrs)
Tried various other epoch numbers and a few other static thresholds and none of these increased our accuracy.

Since we are doing multi-classification, we needed to pick a threshold for when a classification was predicted. This number is currently set to >= 0.2. We tried a few other values than 0.2, but this value worked the best. This thresholding is extremely important for increasing accuracy according to a number of discussions and kernels. Most of the top submissions do a search for the thresholds per classification.

InceptionV3 Architecture



A final comment about pre-trained kernels and accuracy.
The kernel that we based our work on got a leaderboard accuracy of .379. We almost immediately surpassed this accuracy with some changes to the script. Thiμμs script was a kernel that we found understandable and able to be worked on. There was a kernel available that used ResNet34 with a modified 4-channel RGBY that got a .460 leaderboard accuracy. We didn't feel good about using this kernel as we both have a poor understanding of FastAI as well as some of the finer details of this kernel. Thus, we sacrificed accuracy for explainability.

References

InceptionV3 Architecture: https://cloud.google.com/tpu/docs/inception-v3-advanced
Pre-trained InceptionV3 Kernel:
https://www.kaggle.com/mathormad/inceptionv3-baseline-lb-0-379
"FastAI" ResNet34 Kernel:
https://www.kaggle.com/iafoss/pretrained-resnet34-with-rgby-0-460-public-lb