Sprint Report: Team NSTC

#3: 17 - 31 Mar

Sprint Outcome Assessment

After the last sprint, we identified that the lack of data would prove to be large roadblock in the ability for our system to learn efficiently. Because of this, we focused mainly on aggregating extra sets of data, as well as manipulating our current data into more data sets. This includes running each patient through a series of functions that flip the images along the X,Y and Z axes using native Numpy function, increasing our data by a factor of 8, then running each of those through an scipy library that applies a maximum blur filter to the images. Overall this increased our data by a factor of 16; a sufficient amount for our Machine Learning model.

Through the implementation of the above code, we also ran into a problem that all the different datasets found online are in very different formats which we needed to standardize. In order to accomplish this we created a Bash script that unzips and reformats the file in a way that we can then run through the above system.

Finally, we identified that system memory constraints would likely bottleneck training were batching to occur at file system speeds. To overcome this, we modified code to store datasets in the HDF5 format, which enables I/O of numpy arrays at the C level. Prior to this, a linear operations sequence of [read batch 0; train on batch 0; write the results of batch 0; read batch 1; repeat] would have seriously bottlenecked the system. Combined with TF Learn, HDF5 allows the file system to handle read/write operations in the background and manage memory and batch size accordingly. With HDF5, the GPUs will not have to wait for the filesystem.

Project Assessment

Since we corrected the data size issue, referencing our previous sprint report, and the solution in this sprint, given above, we are now currently determining an appropriate approach to pre-processing our data in a timely manner, which utilizes multi-processing techniques. Additionally, we have made modification to our overall project design, shown below.

Design tweaks:

All combinations vs. smart combinations

> To narrow the number of trials and the time/space needed, we can begin by narrowing the resolution ranges we will consider based on memory usage. For example, if a certain resolution can't process a single example on Inception without exceeding GPU memory, we probably don't want to use that resolution (this is especially a 3D problem)

Transfer learning vs. more data

A CNN requires ~100k examples for training. By using transfer learning, Google is able to export pre-trained models and quickly adapt them to new data, ~100 examples needed per class in my experience. Even for the CNN models that don't have pre-trained weights available, we can train them ourselves on ImageNet and then transfer learn over to CT scans.

Problem is, ImageNet is 2D. There is no 3D dataset that I know of to pretrain our 3D CNNs on before training on the CT scans. I can still use transfer learning in 2D, but 3D has to get more creative.

Two options then, creative transfer learning or more data. Creating our own pre-training method for a 3D CNN seems way outside our scope (it's probably worth a phd dissertation). But more data is possible. I've at least doubled the size of the dataset, plus mirroring (4x/8x increases in 2D/3D respectively), plus blurring, plus multiple downsizings. Then we can grab the 11TB! dataset from a long-term study, which arguably has the best baseline scans (evens out the classes a bit).

That would give us the class-balanced, ~100k examples dataset we need for CNN training from scratch. Just probably looking to cloud compute for something like that.

Complete Backlog

Planned Sprint	Backlog Items
<u>Sprint 1</u> 10 Feb - 24 Feb	1. Research ~~a. Familiarize with TF Learn, TensorFlow, TensorBoard, Serving~~ ~~b. Familiarize with dicom format, pydicom/alternate libraries, current interpretation methods~~ ~~c. Consider existing models~~ 2. Pre-Processing ~~a. Consider existing methods~~ ~~b. Implement/verify candidates~~
<u>Sprint 2</u> 24 Feb - 10 Mar	1. C~~omputer Vision~~ ~~a. Consider existing~~ ~~b. Implement~~ 2. Time-Series

	a. ~~Consider existing~~ b. Implement ~~3. Probability~~ a. ~~Consider existing~~ b. ~~Implement~~
Sprint 3 20 Mar - 31 Mar	1. Connect Candidate Graphs 2. Test via TensorBoard/TPOT
Sprint 4 31 Mar - 14 Apr	1. Fixing Mistakes 2. Incorporating new developments
Sprint 5 14 Apr - 28 Apr	1. Deployment 2. Documentation