Correspondence with Mr. Cushing

November 3, 2017

Hello Mr. Cushing,

We have made a significant amount of promising progress on the project. We have been using TensorFlow to retrain the final layer of convolutional neural networks to classify images that we pass to the system. Our methodology is as follows:

* We have developed bash scripts\* to automate the collection of images from the WBT via VNCViewer.
* Our current focus is on the Waterfall display. We are using our script to cut, crop and format Waterfall images from the WBT at specific center frequencies. The images are then saved into appropriately named folders, which will later become the classification labels.
* Our first data set contained 750 images of various signals, and 750 images of no signals.
  + - We used the MobileNet .75 neural network paired with TensorFlow’s retraining module to run a classifier on this data set. Afterwards, when any new image was passed to the model as a parameter of the Python script, the model was able to correctly identify if the image contained ‘Signal’ or ‘No Signal’ at a 99.9999% confidence rate.
  + Our second retraining has focused distinguishing between different signals and the absence of a signal. We used the script to gather 1100 images with a bandwidth of 40 and a center frequency in the range of 88-108 (for image variance), and saved the images to a folder named FM. We recorded 1100 images with a bandwidth of 40 and a center frequency in the range of 2400-2420 (for image variance), and saved the images to a folder named WIFI. Furthermore, we included a file called NoSignal to the same file directory, containing 750 waterfall images containing no signal.
    - We achieved the same level of accuracy on this classifier as the previous one in terms of confidence. If an original image is passed to the Python script following the retraining, and it contains an FM signal, a WIFI signal, or no signal, the image will be correctly identified at a > 99% accuracy level.
      * Caveat: Based on preliminary tests, if an image containing a signal is passed to the module but contains a signal that is ***not*** an FM or WIFI signal, the classifier will label it FM or WIFI regardless, although the confidence level drops significantly ~75-90%. We are optimistic that when retraining with more classes and larger data sets, we will minimize these false positives.

Classification specifications:

So far, each of our retraining models have used Google's MobileNet .75 CNN for TensorFlow. This network was designed for use in resource limited environments. This network is small (~19MB) and relatively fast, although this slightly affects the accuracy. In the future we will be using the larger (~85MB) Inception v3 CNN, which promises significant gains in terms of accuracy an precision, although the retraining period takes longer. In terms of deployability, we do not foresee the need to ever have the network on the device that will be recognizing images; only the retrained graph will be necessary.

The images passed to the classifier in our two initial tests were pulled with the following WBT settings:

Attenuation: 20

Sample Rate: 42

Bandwidth: 40

Waterfall Color: Jet

The waterfall images pulled from the VNC and passed to MobileNet were of size 533x179.

The IMAGE\_SIZE passed to MobileNet was 224px.

The specifications for the ‘screenshooter.sh' script can be viewed in the header of that file.

Upcoming Goals:

1. Gather large data sets from a wider frequency range.
2. Explore grouping image sets by a common frequency rather than by name, and using a script to name signals by frequency.
3. Explore efficiency increase of including frequency values listed beneath graph in images. Hypothesis is that classifier will be able to relate signal patterns to the values, providing a more precise recognition.
4. Explore classifying
5. Formally meet with the Virginia Tech Vision Team to exchange idea to enable collaboration.
6. Set up Skype meeting to discuss status, progress and project direction.

Current environment specifications:

OS: Ubuntu 16.04

GPU:  NVIDIA GeForce GTX1060 6GB

CUDA Version: 8.0 V8.0.61

Python Version: 2.7

cuDNN Version: 6

OpenCV Version: 3.3.0

TightVNC Viewer Version 1.3.10

\*In order to run the bash script that automates image capturing, the system must have ImageMagick installed.