Jonathan Atena Green

CS-477-A

Dr. April Kontostathis

15-11-2015

# SAMM machine learning engine

# Project Description

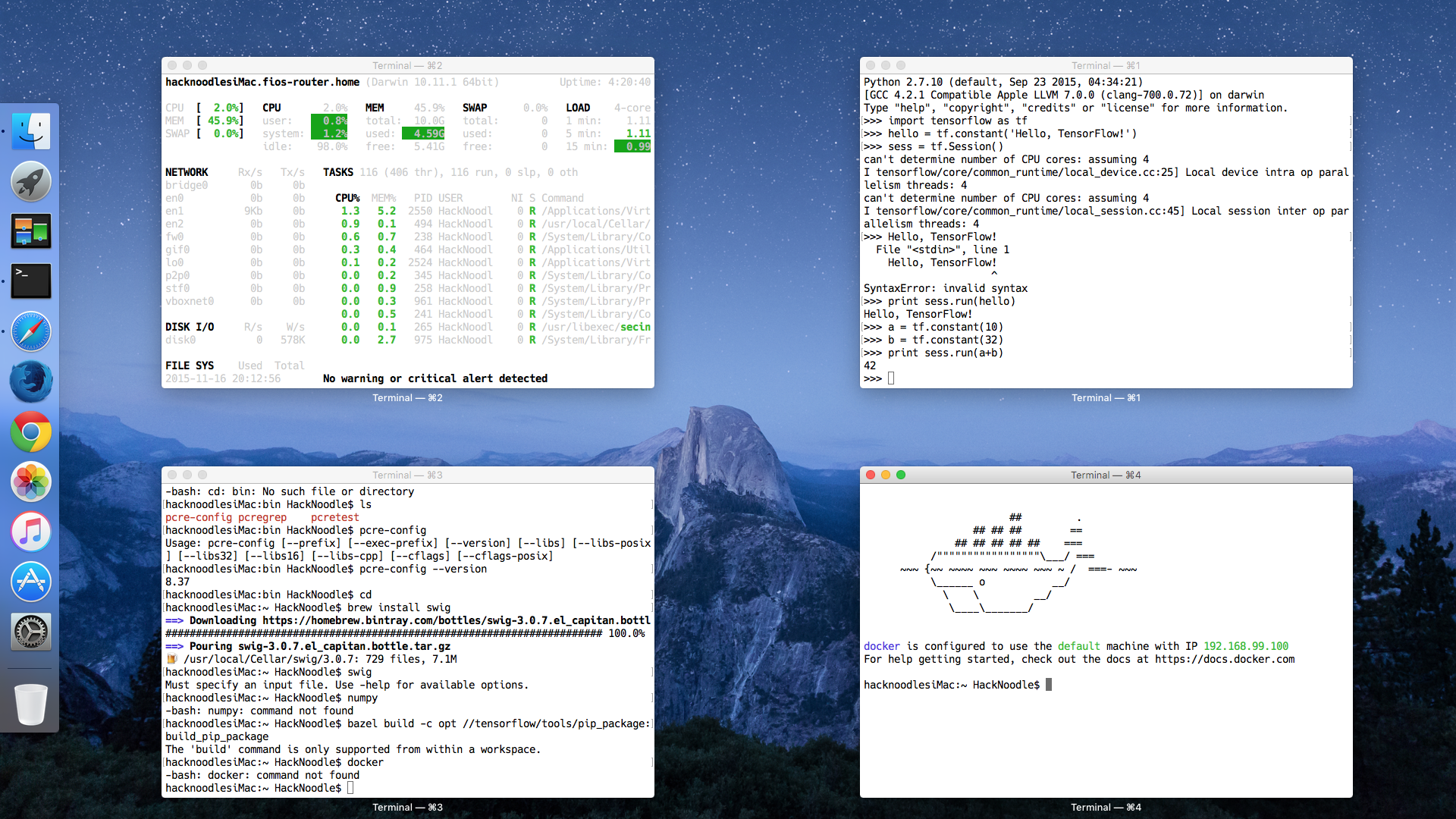
In recent years there has been a resurgence of interest in the potential of learning models that are built for computers, artificial neural networks. Inspired by the way the neural networks form pathways and connections in the human brain, artificial neural networks make it possible for a computer to learn. As an application of neural networks I will implement a machine learning artificial intelligence that harnesses the power of neural networks, focusing on the performance effects of unsupervised and supervised training sets. The Similar Animal Matching Machine, or SAMM, is an artificial intelligence that is capable of using supervised learning to identify patterns of objects within images and associate many different images with the same subject. As the user guides SAMM and limits the scope of a search, SAMM learns what features are important to distinguish similar animals from others. The purpose of designing SAMM is to associate animals (subjects) within their respective images to a database of other animals that are classified as the same animal.

# Scope

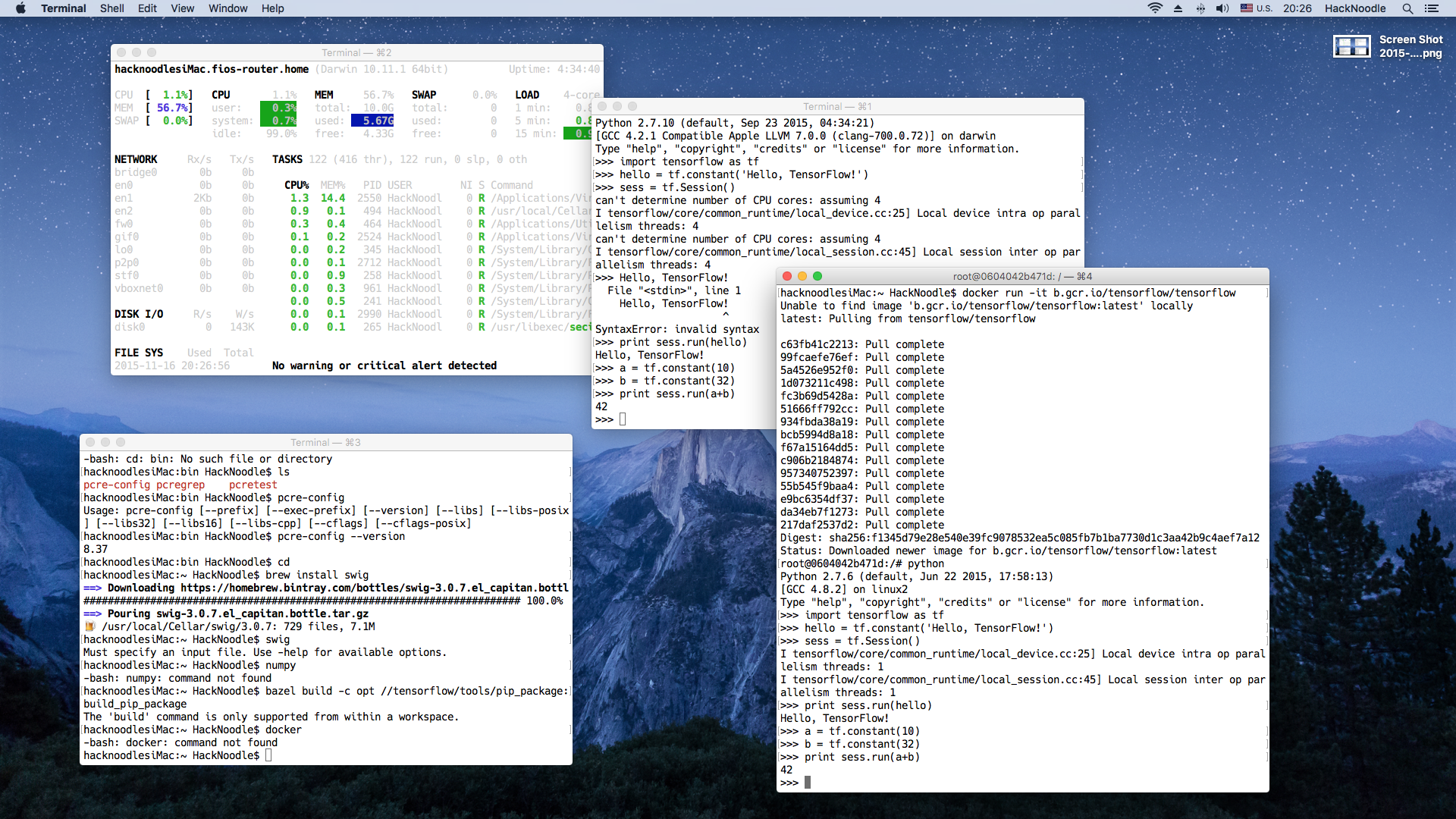
Although image processing and object recognition are significant and very active fields of research, it was more difficult than I previously imaged to begin my own AI research. Image recognition is a research effort that often involves teams of researchers working on computers configured for graphics-intensive experiments in commercial and academic settings. In general these projects are not open-source. There are open-source licenses available, but oftentimes the software and the target machines are outside the price range of an individual researcher. Among the handful of software available, I focused on configuring two systems: Intel's OpenCV and Google's TensorFlow.

I configured my two of my home machines (Model: Dell XPS 8300; CPU: Intel Core i7-2600 CPU @ 3.40Ghz x 8 2nd Generation; GPU: Nvidia Geforce 550 Ti; RAM: 11.7 GB; Model: iMac (21.5-inch, Mid 2011); CPU: Intel Core i5 CPU @ 2.7 Ghz x 4 3rd Generation; GPU: AMD Radeon HD 6770M 512 MB; RAM: 10 GB), for OpenCV. Intel's image processing software is organized in C++ there are extendable but difficult to setup. A large investment of time installing and managing the video and media codecs and dependencies that the most recent version of OpenCV was required. It became a concern for me that the original scope of the SAMM project needed to be revised after a significant (5+ days) amount of time was spent figuring out how to use the software correctly.

Recently, Google made a bold step by open-sourcing their machine learning engine to the research community in order to accelerate change and enable individual researchers to make new discoveries in AI research. I configured TensorFlow easily. It was possible to run TensorFlow in software container (a virtual Linux OS) will all dependencies setup.



**Figure 1. Launching Docker.** In the desktop above, there are four terminal windows showing all the configuration necessary. In the bottom-left, there is an example of trying to install and run Google's build system Bazel locally. In the top-right, TensorFlow's libraries are configured correctly and I am writing code using the TensorFlow libraries in Python. In the bottom-right I have docker installed correctly and running a default Docker instance with TensorFlow setup in a container.

****

**Figure 2**. **Inside Docker using TensorFlow.** Docker running TensorFlow . In the large terminal window, Docker first unpacked all the deltas of dependencies that were not setup in the default container instance, installed everything, and then I launched the Python interpreter in order to work with TensorFlow right away. The process saves much more time over building the software from source (some combination of chmod, ./configure, make, make install, and make distclean for multiple components).



**Figure 3. Wordnet Configuration.** In order to prepare annotated imagesets to examine supervised learning, SAMM project will use ImageNet's web API and WordNet 3.0. Above WordNet CLI is configured for the local machine.

# Status Report

With Google's TensorFlow as the image processing software I will use for the SAMM project, I will revise the scope of the project. I will produce two sets of training images for object recognition, one set will be un-annotated (read unsupervised) images and one annotated set from ImageNet (read supervised). My SAMM image sets will be curated image sets of dogs. Both sets will be small (15 images) because of time constraints and because I will be manually verifying /recording their accuracy in the experiment. Each set will be preceded with training sets (around 100 annotated images) using expectation-maximization algorithm (EM). Another experiment will train TensorFlow with 100 un-annotated images and the process the small sets. The last experiment will observe the effect of performance without training TensorFlow and processing the small sets.

Before the SAMM experiment, I will use TensorFlow to recognize simpler datasets, such as the MNIST computer vision dataset. This dataset contains abstract and blurry numbers that are hard to distinguish. The MNIST dataset contains both annotated and un-annotated image sets for unsupervised and supervised learning researchers. I will also train TensorFlow with EM before processing the MNIST datasets. Researching simpler datasets will enable me to expand the scope of inquiry to object recognition of more complex datasets such as the SAMM dog imageset.

# Tasks Completed To Date

1. Identified expectation maximization (EM) algorithm as the training algorithm for SAMM. (3-4 hours)
2. Installed ffmpeg, libavcodec-dev, libavformat-dev, libswscale-dev, libtbb2, libdc1394 2.x, libpng-dev, Bazel, Numpy, Swig, OpenCV (10+ hours)
3. Properly setup Docker and TensorFlow, ready to process datasets. (10~ minutes)
4. Configured and installed WordNet 3.0. (1 hour)
5. Read image processing documentation and tutorials for OpenCV (5 hours)

# Plan (18-22~ hours/week)

1. Read image processing documentation and tutorial for TensorFlow
2. Use video resources to master Tensorflow features.
3. Select SAMM supervised/unsupervised image sets.
4. Run MNIST dataset without EM training and record results.
5. Run MNIST with EM training and record results.
6. Run SAMM dataset without EM training and record results.
7. Run SAMM with EM training and record results.