

Machine Learning Engineer Nanodegree

Capstone Proposal

Mark Rode

Proposal

Domain Background

Inspired by Stephen Wolfram's "[Personal Analytics](#)" and Cal Newport's book "[Deep Work](#)" I'd like to propose a desktop application that can help people improve focus whilst keeping better track of their screen time.

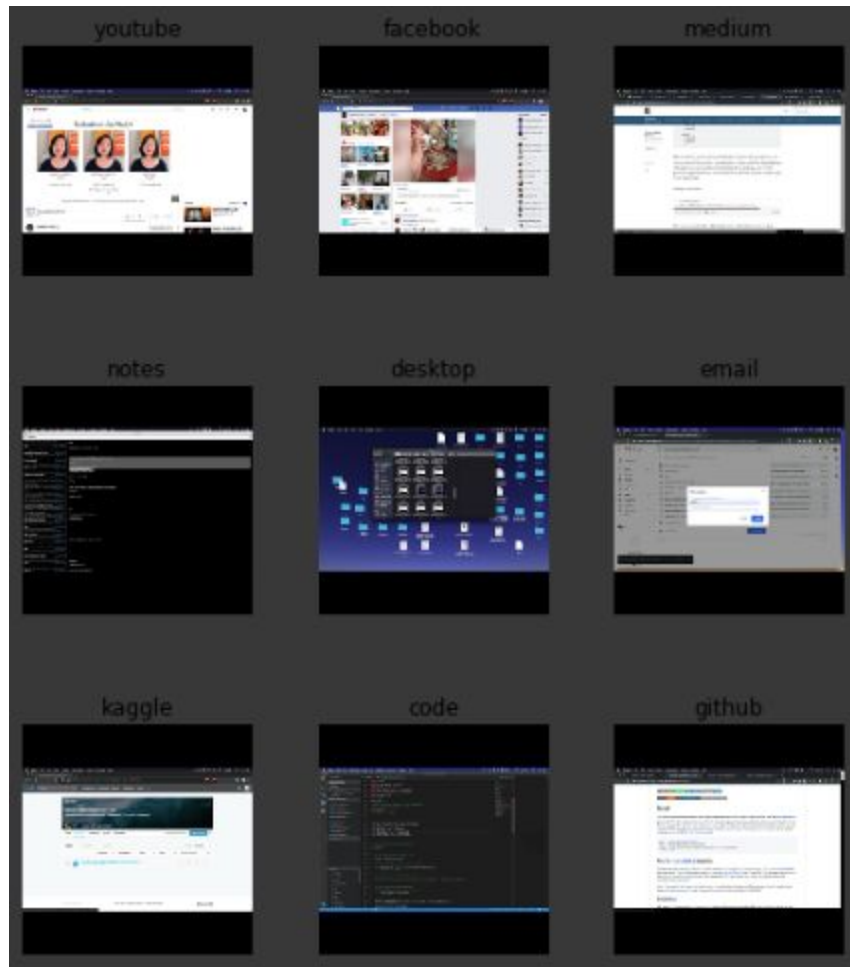
With ever-increasing technology-enabled distractions, I wanted to build something that utilizes technology to help people use their time more effectively.

Because I often catch myself mindlessly wandering to Youtube or routinely looking at one of many apps that has been engineered to condition us to regularly check-in on them, I think it could be helpful to have a tool that can tell me how I am spending my time and possibly 'warn' me if I don't catch myself wasting it.

Problem Statement

The application will be solving a multi-label classification problem with the following approach:

- 1.) Create an evenly balanced dataset that consists of screenshots of the following 14 desktop activities:
 - Writing code in an IDE
 - Navigating the desktop
 - Using Gmail in the browser
 - Browsing Facebook
 - Browsing Github
 - Google Searches
 - Working with a Jupyter notebook
 - Kaggle
 - Reading Medium articles
 - Netflix
 - Taking notes
 - Reading pdfs
 - Using the terminal
 - Youtube



Sample screens

- 2.) Use transfer-learning to train a pre-trained image classifier to distinguish between the listed activities
- 3.) Create a script that takes screenshots in regular time intervals and feeds the images to the classifier
- 4.) Make the app communicate with the user via Mac's built in notification functionality and provide the following information:
 - A summary of the user's last 30 minutes of activity in percentages.
 - A warning if the last 10 predictions classified the activity to be either Facebook, Youtube or Netflix
- 5.) Given that the listed activity classes will not cover all possible desktop activities, like browsing various websites or computer games, the application will save screenshots where the model's prediction confidence is low to a separate folder so that they may be used for training and improving future versions of the model.

Datasets and Inputs

The data has been collected by using a script that takes screenshots of various desktop activities over time.

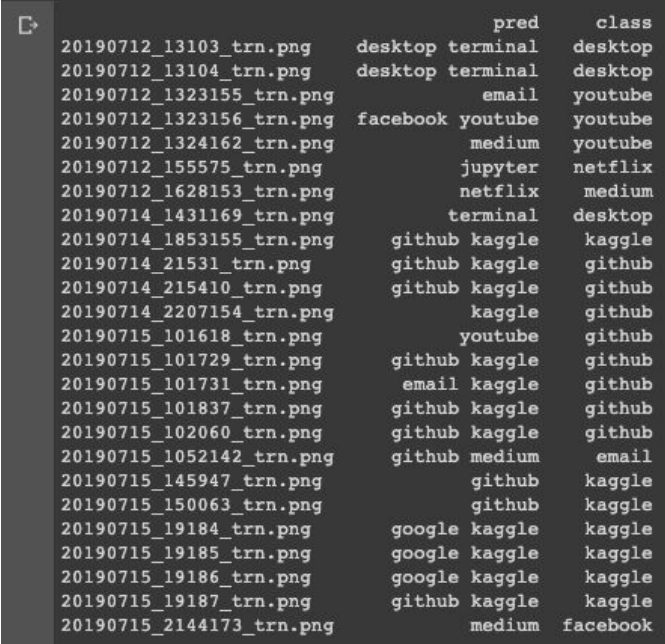
A pre-trained CNN will be used to speed up the learning process and reduce the required amount of data.

Given the relative ease of gathering data and studies suggesting that datasets with unbalanced classes have a negative effect on classification performance [1], I have opted to create a balanced dataset with 14 classes and 100 images per class for the training set and 329 images in the test set. Each image was reduced from the original 1440x900 to 244x244 to save memory and prepare the data for later use.

Benchmark Model

I have chosen to use a pre-trained Alexnet as a benchmark model for this project that was trained for 4 epochs. The resulting model had an accuracy of 92.31% on the test set, given a caveat where the model could predict more than 1 class if a prediction threshold of 0.27 was exceeded for a given screen. The thinking behind this is to allow for screens where e.g. a jupyter notebook is being viewed on Github or a smaller terminal window is opened in the desktop.

The prediction errors on the test set are shown in the image below.



	pred	class
20190712_13103_trn.png	desktop terminal	desktop
20190712_13104_trn.png	desktop terminal	desktop
20190712_1323155_trn.png	email	youtube
20190712_1323156_trn.png	facebook youtube	youtube
20190712_1324162_trn.png	medium	youtube
20190712_155575_trn.png	jupyter	netflix
20190712_1628153_trn.png	netflix	medium
20190714_1431169_trn.png	terminal	desktop
20190714_1853155_trn.png	github kaggle	kaggle
20190714_21531_trn.png	github kaggle	github
20190714_215410_trn.png	github kaggle	github
20190714_2207154_trn.png	kaggle	github
20190715_101618_trn.png	youtube	github
20190715_101729_trn.png	github kaggle	github
20190715_101731_trn.png	email kaggle	github
20190715_101837_trn.png	github kaggle	github
20190715_102060_trn.png	github kaggle	github
20190715_1052142_trn.png	github medium	email
20190715_145947_trn.png	github	kaggle
20190715_150063_trn.png	github	kaggle
20190715_19184_trn.png	google kaggle	kaggle
20190715_19185_trn.png	google kaggle	kaggle
20190715_19186_trn.png	google kaggle	kaggle
20190715_19187_trn.png	github kaggle	kaggle
20190715_2144173_trn.png	medium	facebook

To improve on the benchmark model I intend to test and compare the performance of more modern CNN architectures like the ResNet and InceptionNet models in addition to implementing further techniques like unfreezing and fine-tuning the model layers.

Evaluation Metric

To keep things simple and because the data distribution is balanced I've chosen to use accuracy to measure the performance of the classifier, as any form of incorrect classification will reduce the utility of the tool.

$$Accuracy = \frac{true\ positives + true\ negatives}{dataset\ size}$$

A further factor to be taken into account will be the time that the trained network will require to actually classify an individual image and provide feedback to the user.

Project Design

The application will be built in two phases:

1.) Phase I: Model comparison and training

The first phase involves training and fine-tuning multiple models on the training data. The best performing models will then be saved and downloaded to a desktop computer.

2.) Phase II: Classifier inference and feedback

The next phase will use a script that takes a screenshot every 10 seconds, converts the screenshot to a smaller 244x244 pixel image and runs the screenshots through the pre-trained classifiers.

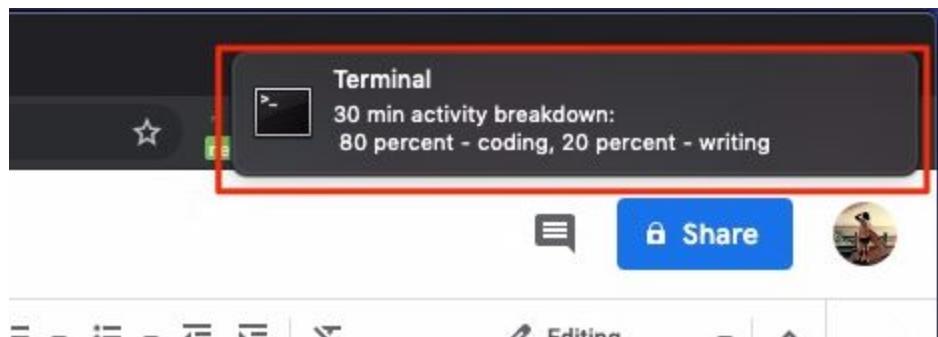
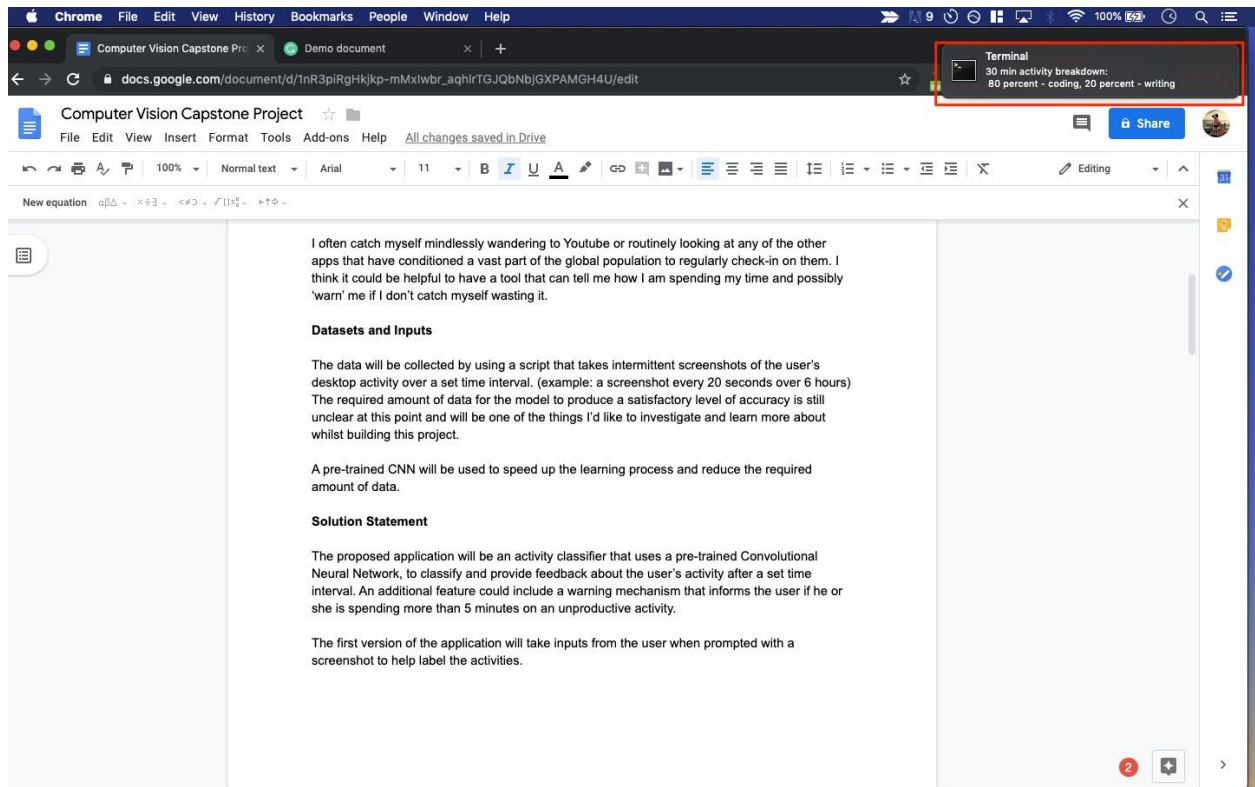
A model will then be selected based on both accuracy and inference speed for use in the actual application.

The selected model will be used to classify the user's future desktop activity and provide the user with a quick report on how the last hour was spent e.g. 60% coding, 5% Facebook, 5% Soundcloud 30% Medium using Mac's built-in notification center.

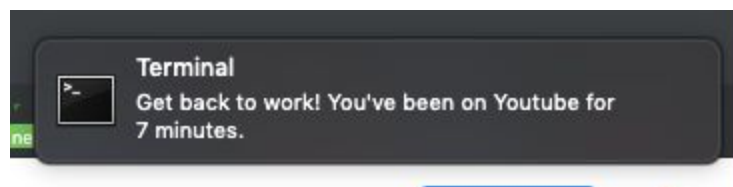
Images where the model's prediction certainty falls under a threshold of 0.2, will be saved to a separate folder, so that they can then be used to improve on future versions of the application.

Sample screenshot of application output:

Application providing summary of last 30 mins activity



Application output with warning to get back to work.



References

[1] Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. Link: <https://arxiv.org/pdf/1710.05381.pdf>