Oregon State University
College of Engineering

CS Capstone Technology Review

June 12, 2018

Aerolyzer

Prepared for

Kim Whitehall

Prepared by

Group 19
Aerolyzer
Daniel Ross
Kin-Ho Lam
Logan Wingard

Abstract

The Aerolyzer Project aims to deliver a new source of air quality and weather information through leveraging existing weather data and image analysis algorithms. When complete, this open-source project shall feature a Python library that uses image classification and third-party weather APIs, displayed with an intuitive web-based user interface. This document outlines the software design descriptions for the Aerolyzer Library

1 Table of Contents

Contents

2 Sunset and Sunrise Classification

2.1 Overview

The objective of the Aerolyzer project is to create a tool that infers air quality based on the distribution of sunlight hues in images of sunsets and sunrises. To save server-side computation resources and decrease the chance of providing inaccurate analysis, the Aerolyzer library must filter out images that cannot be analyzed. Unacceptable images include pictures that do not feature a sunset, sunrise, or have objects obscuring too much of the horizon. However, variables such as camera manufacturer, perspective, post-processing, image format, obstructions, and lighting prevents a simple algorithm from classifying all possible image variations.

2.2 Criteria

Aerolyzers unique image criterion necessitates a predictive data model capable of detecting horizons with a minimum of 66% certainty and sunsets or sunrises with a minimum of 50% certainty. Third-party tools or libraries to be used in Aerolyzers horizon classifier shall be compared by availability and utility. The Aerolyzer project shall ideally use free and non-proprietary libraries to perform image analysis. Such libraries are preferably open source with usage licenses compatible with Apache License 2.0.

2.3 OpenCV

OpenCV is a BSD-licensed computer-vision library with many relevant sub-modules. Containing over 2,500 algorithms, OpenCV can analyze images and videos; recognizing faces, identifying objects, performing visual transformations, and much more. Open ALPR is an example of a powerful image classifier built with OpenCV. Open APLR is an automatic license plate recognition library, capable of identifying and reading license plates and car models in near real-time. Distinguishing cars and license plates is conceptually similar to Aerolyzers horizon detection goals. Both classifiers must distinguish acceptable objects despite vision obscurest, and image sensor variations. OpenCVs imgproc module performs image filtering, geometrical image transformations, and color space conversion. The objdetect module can detect predefined classes such as faces, eyes, and cars. These modules present useful tools that are capable of horizon detection given proper tolerances and algorithms. One can use image transformations to normalize image data such that a classifier can more easily distinguish what is in the picture. Pre-defined classes are useful in detecting obstructions that may block a good view of the horizon. OpenCV is a potential candidate for horizon/sunset/sunrise detection due to its feature-rich library, ample documentation, and pedigree of successful image classifiers. [1]

2.4 Tensorflow Inception

One of the most advanced machine learning libraries, Tensorflow can build versatile and accurate predictive data models. One can apply Tensorflow to the Aerolyzer project by training its Inception model. Rohan Vermas Galaxy Image Classifier is an example of a classification library built with Tensorflow Inception. This library can detect whether a submitted image contains a spiral or elliptical galaxy. Vermas galaxy classifier is conceptually like Aerolyzers planned horizon/sunset/sunrise classifier in that both can feed images into Tensorflow to build a predictive model, and both have specific criterion which can be reflected in a training dataset. Tensorflows ease of use, proven library, and ample documentation makes it a potential candidate to create a powerful image classifier. [2]

2.5 Amazon Rekognition

Amazon Rekognition is a supervised machine learning library with a focus on usability. Rekognition performs analysis on Amazons servers; its API is set up to service image recognition calls on-demand. When provided an image, Rekognition will simply return a JSON data structure containing elements detected in the image as well as a confidence score. Rekognition can be readily deployed onto the Aerolyzer AWS server, setup is minimal as most of the computation and complex algorithms are hidden to the developer. Rekognition is a less preferable alternative to OpenCV and Tensorflow due to its processing cost, limited library, and closed-source proprietary algorithm. [3]

2.6 Conclusion

Tensorflow and OpenCV are strong image recognition platforms. Their powerful libraries make building a predictive model to filter unacceptable images possible. Although the final algorithm and its components are subject to change; at the request of Dr. Whitehall, horizon classification development shall focus on OpenCV algorithms.

	Cost	License	Tools/Advantages
OpenCV	Free	Open Source, BSD License	Robust computer-vision library
Tensorflow	Free	Creative Commons Attribution 3.0 License, Apache License 2.0	Supervised machine learning library with proven image-recognition modules
Amazon AWS Rekognition	$0.01/1,000 images	Proprietary & Closed source	Supervised machine learning library with cloud platform processing

3 Aerosol Inference

3.1 Overview

Monitoring atmospheric aerosols is important due to their effects on the atmosphere's chemical composition and radiation distribution. These particles originate from natural sources such as volcanic eruptions, and unnatural sources such as fossil fuel combustion. Called the ”Rayleigh scattering effect”, aerosol particles whose diameter is smaller than the wavelengths of visible light scatter the suns beams. During the daytime when the sun is almost directly above an observer, the shortest wavelengths of sunlight (blues and violets) refract, resulting in the typical blue color of the sky. For an observer looking at the sun as it approaches the horizon during sunrise or sunset, the Suns light rays travel further through the atmosphere. This lengthened path scatters an increased amount of violet and blue light to out of the beam, noticeably reddening sunsets or sunrises. [4]

The presence of airborne aerosol pollutants, typically 0.5 to 1 m in diameter, diminishes the vibrancy of sunsets or sunrises, resulting in hazy colors. Red or deep orange sunrises or sunsets typically indicate larger aerosols which can suggest of poor air quality. [4] The Aerolyzer library hopes to feature a module that infers aerosol size based on the hue distribution in acceptable horizon images.

3.2 Criteria

Third-party tools or libraries to be used in Aerolyzer's aerosol inference module shall be compared by utility. Desirable libraries shall incorporate some method of statistical analysis to identify patterns.

3.3 Tensorflow Inception

One can use this relationship of horizon hues and aerosol size to train a Tensorflow classifier to recognize color change and infer air quality. Accomplishing this task will require two components: proper implementation of the Tensorflow Inception algorithm, and a large training dataset of images. The image training dataset shall be comprised of horizons whose colors exemplify characteristics of the presence of larger aerosols. The Tensorflow Inception model can be described as a chain of modules comprised of deep layers of convolutions and concatenations. During training, these modules will re-organize themselves to form a final classification model that fits its training set. When fed an image as input during classification, this model will produce a label and certainty percentage of how well the input image matched the model. [5]

The Xvision library uses a Tensorflow generated model to detect nodular growth in chest x-rays. Xvisions classification scope is like Aerolyzers aerosol detection in that the difference between high atmospheric aerosols vs low aerosols and nodular growth vs normal x-rays are discernable through analyzing color vibrancy. [6]

3.4 TFLearn

TFLearn is a wrapper built around Tensorflow. It features a modular deep learning library with an easy-to-use higher-level API to enable faster algorithm prototyping. While the underlying algorithms are still Tensorflow, TFLearn is advantageous in that it simplifies complex Tensorflow functions into less verbose calls. This can be combined with a Tensorflow Inception module implementation by simplifying lower-level algorithms to higher-level system calls. This simplification will make the overall inference algorithm more accurate. [7]

3.5 OpenCV

OpenCV's Support Vector Machine (SVM) library is a supervised machine learning algorithm. Given a sample dataset, SVM generates an optimized hyper-plane model to categorize future input. For example, an SVM can be trained by providing several pre-sorted sets of images of sunsets and sunrises. This can be applied to the Aerolyzer project by training an SVM with a set of horizons with known aerosol content. [8]

3.6 Conclusion

While the overall software focus is subject to change throughout the development of the Aerolyzer library, a supervised machine learning library is an optimum choice for performing aerosol analysis on images of the horizon.

	Cost	License	Tools/Advantages
Tensorflow	Free	Creative Commons Attribution 3.0 License, Apache License 2.0	Supervised machine learning library with proven regression-analysis modules
TF Learn	Free	Creative Commons Attribution 3.0 License, Apache License 2.0	High-level API wrapper for Tensorflow
OpenCV	Free	Open Source, BSD License	Support Vector Machine library

4 Data Trend Prediction

4.1 Overview

The Aerolyzer project hopes to perform trend analysis and predict future aerosol data as a stretch-goal. Predicting air quality based on aerosol data requires in-depth atmospheric knowledge. However, one can expect that a trained neural-network can generate a probabilistic model given a large dataset. To accomplish this stretch-goal, Aerolyzer library requires algorithms that utilize deep statistical analysis libraries to interpret and present Aerosol information in an understandable format.

4.2 Criteria

The Aerolyzer project will need to generate a trend analysis using a large dataset. This dataset may consist of data scraped from 3rd party weather APIs and Aerolyzers own aerosol inference data.

4.3 Tensorflow RNN

The Tensorflow Recurrent Neural Network facilitates the creation of a probabilistic model which can assign probabilities to a sequential set of data. One can apply the Tensorflow RNN to the Aerolyzer project by first reshaping a set of training data into matrix of arbitrary batch_size rows. Neural Networks are trained by approximating the gradient of loss function with respect to the datas weights by looking at small subsets of data. This enables Tensorflow to build up a computational graph of multidimensional arrays, also known as tensors. When new data is added to the RNN, Tensorflow will apply its computational graph and adjust its trend analysis accordingly. In this way, one can use a Tensorflow RNN to generate a probabilistic model to approximate aerosol distribution or air quality from large amounts of data. However, the data fed into the Tensorflow RNN must be managed and organized such that understanding new data does not rely on extremely old data. As more information is added, this ”data gap” grows which will prevent an RNN from learning to connect new information. [9]

4.4 Tensorflow LSTM

Tensorflow Long Short-Term Memory networks are a special type of Recurrent Neural Network that avoid the long-term learning dependency problem encountered in some RNNs. This is accomplished by the LSTMs incorporation of sub-networks within each neural network module. In short, LTSMs do not immediately incorporate all data that passes through the RNN. Instead, an LTSM ”regulates” the information it uses to form its computational graph by choosing to leave some data unmodified. The result is a predictive model that adds, modifies, and concatenates information as new data re-shapes old models. Tensorflow LTSMs are widely used and have been proven to work in many probabilistic models. LTSM models have successfully been applied to shape prediction. One can surmise that given adequate testing data, the same can be done for aerosol weather trend inference. [10] [11]

4.5 R & Matlab Libraries

Leveraging the statistical power of R and Matlab libraries will enable the Aerolyzer project to interpolate weather patterns from sampled images. This shall be accomplished through regression analysis which shall consider dates of when images are taken, the inferred aerosol content of the image, patterns referenced from 3rd party APIs. R and Matlab libraries are readily available modules in the Python library which can be applied to Aerolyzers future dataset. [12] [13]

4.6 Conclusion

Data interpretation is a challenging task. This will require a large dataset of weather information, and a correct implementation of RNN, LTSM, or a universal weather statistical model. Included as a ”stretch-goal”, complete implementation of a predictive model may be out of the scope of this project given the time constrains and lack of training data. Current research efforts shall focus on Tensorflow LTSM.

	Cost	License	Tools/Advantages
Tensorflow RNN	Free	Creative Commons Attribution 3.0 License, Apache License 2.0	Supervised machine learning library Recurrent Neural Network module
Tensorflow LSTM	Free	Creative Commons Attribution 3.0 License, Apache License 2.0	Supervised machine learning library with proven Long Short-Term Memory module
R & Matlab Libraries	Free	Apache License 2.0	Proven statistical libraries. Widely used in academia.

5 Web Framework

5.1 Overview

Web frameworks are what will enable the Aerolyzer library to interact with the users, so the framework’s performance will directly affect the end product of the Aerolyzer app. Depending on which framework the Aerolyzer app utilizes, the integration of the Aerolyzer python library could be simpler. The web framework will also inform the structure of any user input, so clarity in this aspect is critical.

5.2 Criteria

1) Speed
2) Security
3) Web Development features

5.3 Options

5.3.1 Express.Js

Express is a streamlined version of Node.js. Express itself doesn’t have database support, but is capable of running third-party modules to interface with the database. Express is useful for API development as it includes the neccesary http functionality to provide users functions. Since Express is a form of Node, an app built on Express will crash if an exception goes uncaught. This can make potential debugging a hassle since the app will have to be restarted completely if anything goes wrong. Express typically performs a little quicker than it’s parent framework, Node.js, but it’s speed can suffer significantly under high traffic. Security isn’t built in Express functionality, but there is third-party middleware that sets up HTTP headers and transmission security. [14]

5.3.2 Ruby on Rails

Ruby on Rails, ’Rails’ for short, is a Ruby based web framework. Rails claims to value web application conventions and will default to many of those conventions in the case that the developer didn’t make custom configurations. This hopes to make Rails code shorter and less buggy than other frameworks that require minutiae from the developer. Rails runs off of a SQLite3 database and is best suited for interfacing with that type of database. Rails has a default HTTP security built directly into the framework. Being written in Ruby and having many features built-in means that Rails isn’t the fastest framework. [15]

5.3.3 Django

Django is a python based framework that claims to make database-driven Web development fast and easy. Django also supports clean URL designs and in framework templating. The first step to using Django is writing the model which maps out your database layout. [16] Since Django apps are centered around these models, Django has a built-in API for manipulating any objects in the model. [17]

5.4 Web Framework Discussion

Express immediately takes 3rd place amongst these choices, because it’s overly reliant on third party middleware [14]. Rails and Django are similar in respect to included features, but Django has it’s model API for increased ease of use [15] [17].

5.5 Conclusion

Django offers the most complete set of framework features while maintaining speed. Since Django is written in python and the Aerolyzer library is also written in python, integrating the Aerolyzer library would be easiest with Django. [17]

Web Framework		
Choice	Pros	Cons
Express.js	Lightweight, Quick	Requires Middleware for many features, Prone to crash
Ruby on Rails	Robust features	Not particularly fast
Django	Robust features, Quick	Largest framework

6 Database Management System

6.1 Overview

A database is how computers store data and its relationships with other values. For Aerolyzer to work properly we'll need a database to store photos and weather analysis under the zip code they apply to. Since the 3 most reasonable choices are all SQL based, the management system that works with the web framework will most likely be the best choice.

6.2 Criteria

1) Speed
2) Size

6.3 Options

6.3.1 SQLite3

SQLite boasts it's status as the most deployed database in the world [18]. SQLite manages the SQL database within a single disk file and the library itself is about 1.8 MiB [18]. Since SQLite is self-contained and lightweight it's suitable for most low traffic applications [19]. Where SQLite starts to suffer is it's speed when given requests for data that are large relative to the available memory. For Aerolyzer this may be problematic, since the database hase to be capable of fetching high definition pictures quickly [18].

6.3.2 MySQL

MySQL is Oracle's SQL database server. MySQL has a larger library to install and has administrative functionality included. This means that MySQL would require considerable set up and potential maintenance. The size and effort required for MySQL is all in order to provide a database that can handle large requests and high traffic. [20]

6.3.3 PostgreSQL

PostgreSQL is the SQL supporting descendant of the POSTGRES database system. PostgreSQL includes a wide function library with Python support. The transfer size PostgreSQL supports is more than enough for the Aerolyzer app. The administrative functions included in PostgreSQL would make monitoring the database efficient. [21]

6.4 Database Management System Discussion

All three database systems would work in the Aerolyzer app, but the question remains which would work the best. PostgreSQL and MySQL would most likely have similar performance once set up, while SQLite might have issues under high traffic.

6.5 Conclusion

MySQL is the most complete SQL system of these choices, and its performance is most likely going to be similar to PostgreSQL. The fact that MySQL has a more expansive library means that it's more likely to have the database solutions we need during development.

Database Management System		
Choice	Pros	Cons
SQLite3	Lightweight, Fast	Not suited for high traffic
MySQL	Fast, Robust features	Large, Administration required
PostgreSQL	Fast	Administration required

7 Visualization & Display

7.1 Overview

Presenting the output from the Aerolyzer library is the key step in making the user understand the project. Having unclear or unappealing visuals to show the results could potentially be worse than simple outputs. Most of the presentation can be acheived with HTML alone, but the main display will need a script to insert charts.

7.2 Criteria

1) Clarity
2) Reliability

7.3 Options

7.3.1 Google Charts

Google Charts is a JavaScript that is inserted into the web page and is then fed the data to be charted. There are 28 different types of charts and they all are rendered as SVG or VML. Google Charts lets the chart make a query to the database directly and displays customizable tooltips when moused over. [22]

7.3.2 CanvasJS

CanvasJS, in addition to having 30 types of charts, claims to be an ultra fast HTML5 charting library. Rather than render SVG charts, CanvasJS inserts HTML code when rendered. With a wide library of customization, CanvasJS claims to make the developer's charts work on Chrome, Firefox, Safari, and Internet Explorer. [23]

7.3.3 ChartJS

ChartJS renders charts into HTML5, like CanvasJS, but is open source. ChartJS only has 8 chart types, but offers custimization that lets developers mix chart types. [24]

7.4 Visualization & Display Discussion

CanvasJS is the most robust JavaScript of the 3, which may take some additional code to get the desired results, but the end product will most likely be closer to the ideal visualization. [23]

7.5 Conclusion

Having visuals rendered into HTML5 elements would be ideal. It ensures that the graphics are rendered with the rest of the page. Of these three libraries, CanvasJS is most appealing but isn't open source. Aerolyzer is an open source project and avoiding conflicting licences is important. Since this is the case, Google Charts is the best option since it's terms of use won't interfere with Aerolyzer's liscencing. [22] [23]

Visualization & Display		
Choice	Pros	Cons
Google Charts	Free to use, sufficient features	not as fast as CanvasJS
ChartJS	Open Source	Not as many features
CanvasJS	Very Robust	Not Open Source

8 Neural Networks

8.1 Available Technologies

8.1.1 Shallow Neural Network using Numpy

Neural networks are matrices of functions and floats that weighs each value itself after "learning" from multiple sets of data. It does this using a long, but simple sigmoidal function seen here:
$\sigma(x_0 * w_0 + x_1 * w_1 + ... + x_m * w_m + b)$ [25]
Where the Ws are the weights, b is the bias, and Xs are the inputs. With enough data, a shallow neural network could easily be written and trained in a simple python program using only base python and numpy. The program "learns" by generating random weights and biases, then checks the solution against known data, and adjusts the weights accordingly, until it has the accuracy desired.

8.1.2 Deep Neural Network using Tensorflow by Google

For deep neural networks, a supervised machine learning library such as Tensorflow by Google would be ideal. Deep neural networks are just neural networks with two or more hidden layers. With the more hidden layers, the faster and more accurate the neural network will be, though manually creating these using numpy or other libraries could be very time consuming.

8.2 Goals

Neural networks could use color data extracted from images as input, and through numerous tests, use accurate weights and biases to accurately identify the types of aerosols in the air, and if they could possibly be harmful or not. To train a neural network such as this would take time and a lot of data, though would definitely be doable.

8.3 Criteria

There are two main factors that will determine whether the use of a neural network would be valid or not, and these are the amount of data we have, and the amount of time we have. With much data that has already been classified by hand, the neural network would be as simple to set up as manually inserting data into a long array,and letting the program do its thing. Unfortunately, classifying the images by hand so that the program could learn can take very long and may not be faster than creating a function that will analyze the colors.

8.4 Conclusion

Neural networks have a wide array of uses and may not be limited to color analysis. A shallow neural network made using functions available in the numpy python library could be used to check the validity of Aerolyzer's horizon detector. For color analysis, a supervised machine learning library such as Tensorflow would be ideal for using a neural network to analyze images.

Neural network		
Choice	Pros	Cons
Shallow	Simple, easy to write and train	Wouldn't be as accurate as a deep neural network
Deep with supervised learning	Very acurate	Difficult to implement, would need Tensorflow

9 IMAGE SCRAPING

9.1 Available Technologies

Getting enough data to work with is essential to testing the validity of images the Aerolyzer app will receive from its users. Finding these images is more complicated than it would come across however. While the functionality of the finished program will include receiving images from users, for testing purposes, a method of scraping websites for valid images is a must. [26]

9.1.1 python image scraper 2.0.7

There are multiple python libraries and third party API for image scraping that could supply the data needed. The python library that would work best would most likely be python image scraper 2.0.7.. Because this is a python library available from python's package index, it is fairly simple to install and use. With options such as max images to scrape, image sizes and types, and even order to scrape them, this is an efficient tool for scraping images from URLs where the images are known to be valid. [26]

9.1.2 3rd Party API

Other available technologies for image scraping includes a google image scraper and Instagram's image scraper. There are several Instagram image scrapers, one of which is available on Pypi named Instagram-scraper 1.5.14. As for Google images, there are even more readily available image scrapers, many on Pypi, that could pull images directly from Google images. [27]

9.2 Goals

What Aerolyzer hopes to get out of the image scraper is a high volume of valid images. Testing the validity of classifiers for percent accuracy requires hundreds, if not thousands of images that do not have filters, and have valid exif data. These images need to be available quickly as if an image scraper takes too long, it would just be faster to manually find images.

9.3 Criteria

The image scraper that will ultimately be used has to consistently supply many images that do not have filters and have valid exif data. The most important criteria however is the consistency. An Instagram image scraper may supply numerous valid images, though these could be taken when location data was turned off, or have filters applied, making it an impractical image scraper for Aerolyzer. The image scraper to be used will find consistently valid images fast and in high volumes to make testing as simple as possible.

9.4 Conclusion

With such an extreme volume of images needed for testing purposes, most likely multiple image scrapers will get used. Due to horizon detection not needing exif data and can have filters, Aerolyzer could use any and all image scrapers to find enough images to test, while color detection is more precise and will likely need many filters to weed out unwanted images, therefore will be found using Instagram scraper or Google scraper.

Image Scrapers		
Choice	Pros	Con
python image scraper 2.0.7	Free, versatile, and easily implemented	We would still need to find
instagram image scraper	Free, fast, easily implemented, and has a large selection of immages	Many of the images have filter

10 Python Modules/Libraries

10.1 Available Technologies

10.1.1 Pypi

PyPi is Python's package index. It is a massive database of packages created by developers to make custom python libraries widely accessible. This is the most valid technology to use for importing python modules. With the way Aerolyzer's repositories are set up, it would be best to upload the module to pypi. As for the design of our module, there is not much wiggle room as to importing and using the module. [28]

10.2 Goals

Aerolyzer's goal is to create a custom library that will contain all the tools Aerolyzer creates in order to complete the goal of image classification, analysis, etc. Aerolyzer will not only produce an app that utilizes the Aerolyzer python library, but will also make the library readily available for others to use.

10.3 Criteria

The criteria that determines the library setup will be the availability and convenience. If the Aerolyzer library isn't easily accessible, no one will use it, or it would be too much of a hassle to import and use Aerolyzer's horizon detection, color analysis, and others.

10.4 Conclusion

Python's package index is a must use to make the Aerolyzer library easily available to the public.

11 Bibliography

References

[1] matthill. (2017) Open alpr. https://github.com/openalpr/openalpr.

[2] Rhnvrm. (2017, Oct) rhnvrm/galaxy-image-classifier-tensorflow. [Online]. Available: https://github.com/rhnvrm/galaxy-image-classifier-tensorflow

[3] Amazon rekognition deep learning-based image analysis. [Online]. Available: https://aws.amazon.com/rekognition/

[4] S. F. Corfidi. (2014, Sep) The colors of sunset and twilight. [Online]. Available: http://www.spc.noaa.gov/publications/corfidi/sunset/

[5] Image recognition — tensorflow. [Online]. Available: https://www.tensorflow.org/tutorials/image_recognition

[6] ayush1997, "Xvision," Jul 2017. [Online]. Available: https://github.com/ayush1997/Xvision

[7] A. Damien. Tflearn: Deep learning library featuring a higher-level api for tensorflow. [Online]. Available: http://tflearn.org/

[8] Introduction to support vector machines. [Online]. Available: https://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm

[9] E. Hallstrm. (2016, Nov) How to build a recurrent neural network in tensorflow (1/7). [Online]. Available: https://medium.com/erikhallstrm/hello-world-rnn-83cd7105b767

[10] C. Olah. (2017, Aug) Understanding lstm networks. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[11] Mouradmourafiq. (2017, Jun) mouradmourafiq/tensorflow-lstm-regression. [Online]. Available: https://github.com/mouradmourafiq/tensorflow-lstm-regression

[12] Molecular organisation and assembly in cells. [Online]. Available: https://www2.warwick.ac.uk/fac/sci/moac/people/students/peter_cock/python/lin_reg

[13] stepwiselm. [Online]. Available: https://www.mathworks.com/help/stats/stepwiselm.html?requestedDomain=www.mathworks.com
[14] (2017) Express guide. Node.js Foundation. [Online]. Available: https://expressjs.com/en/guide/routing.html
[15] (2017) Rails guide. [Online]. Available: http://guides.rubyonrails.org/getting_started.html
[16] (2017) Getting started with django. Django Software Foundation. [Online]. Available: https://www.djangoproject.com/start/
[17] (2017) Django at a glance. Django Software Foundation. [Online]. Available: https://docs.djangoproject.com/en/1.11/intro/overview/
[18] (2017) About sqlite. Hwaci. [Online]. Available: https://sqlite.org/about.html
[19] (2017) Appropriate uses for sqlite. Hwaci. [Online]. Available: https://sqlite.org/whentouse.html
[20] (2017) Mysql 5.7 reference manual. Oracle. [Online]. Available: https://dev.mysql.com/doc/refman/5.7/en/
[21] P. G. D. Group. (2017) Postgresql 10.1 documentation. [Online]. Available: https://www.postgresql.org/docs/10/static/index.html
[22] (2017) Google charts. Google. [Online]. Available: https://developers.google.com/chart/glossary
[23] (2017) Canvasjs. CanvasJS Team. [Online]. Available: https://canvasjs.com/docs/charts/basics-of-creating-html5-chart/
[24] (2017) Chartjs. https://github.com/chartjs/Chart.js/graphs/contributors. [Online]. Available: http://www.chartjs.org/docs/latest/
[25] S. Raval. (2016) Build a neural net in 4 minutes. Youtube. [Online]. Available: https://www.youtube.com/watch?v=h3l4qz76JhQ
[26] A. N. S. (2017) Imagescraper 2.0.7. Python. [Online]. Available: https://pypi.python.org/pypi/ImageScraper
[27] R. Arcega. (2017) instagram-scraper 1.5.14. Python. [Online]. Available: https://pypi.python.org/pypi/instagram-scraper
[28] Python. (2017) Pypi. Python. [Online]. Available: https://wiki.python.org/moin/FrontPage#start