

Executive Summary

Problem Statement: Many large organisations use their Geographic Information Systems (GIS) for daily operations of tracking and maintaining assets. Over time, changes to assets and human errors in data entry can cause data issues which are costly and complex to reconcile and correct.

Solution: The emergence of image classification via machine learning and accessibility to Google GIS information, provides businesses the opportunity to find new ways to reconcile their GIS data. The first step to achieving new improvements to data quality is to conduct a proof of concept in the automated task of accessing Google images at specific Geospatial coordinates and classifying the images as either reconciling or not reconciling with existing business GIS information.

Contents

Executive Summary	1
Task A – Understanding the Data	2
Task B – Download Google Satellite Images of Coordinates	3
Task C – Download Google Street View Images of Coordinates	4
Task D – Altering Google Street View Images To Enhance Object Classification	5
Task E – Migrate to TensorFlow from SciKit Learn Packages	6
Task F – Conclusion	8
References	9

Task A – Understanding the Data

The initial step in this proof of concept is to be able to extract GIS data and be able to identify individual data points to locate, transform and interact with.

Two common GIS file formats for extracting data points and geospatial coordinates are “mid” and “mif” files;



transformer.pri_sec_location.mid



transformer.pri_sec_location.mif

With Python GeoPandas package, it is possible to plot each of the GIS data point coordinates in a basic visual format;

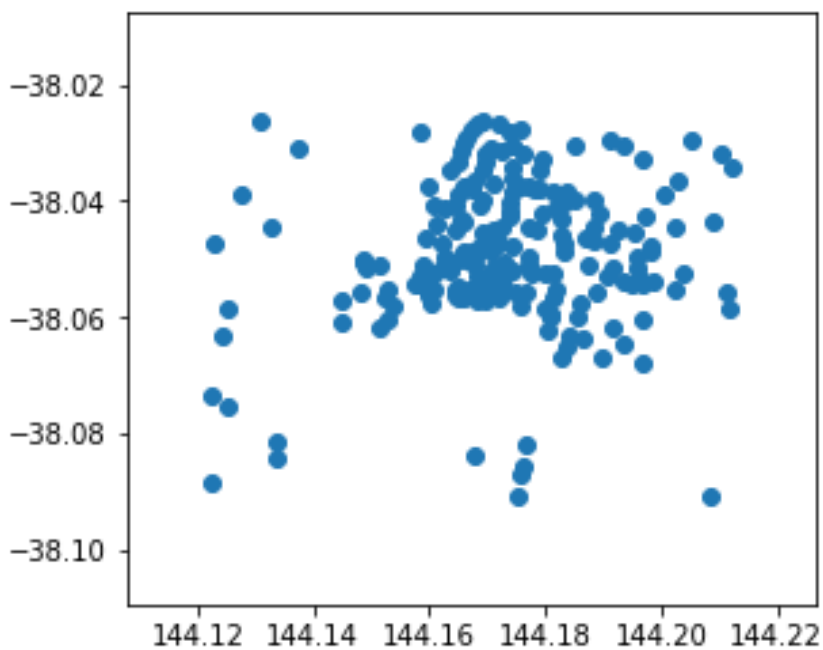


Image 1 - Python GeoPandas plot of GIS geospatial coordinates of known transformers

Task B – Download Google Satellite Images of Coordinates

Having created a Google API registered account, access was enabled to Google satellite image downloads. Google satellite images can be downloaded at difference scale (zoom levels). Images of specific coordinates were downloaded incrementally from scales 16 through to 22. Scale 16 images had clarity but the objects in the images were too small to classify, whilst the objects in the images got larger as the scale value incremented, at scale 20 the objects began to blur. Whilst scale 19 was the optimum, realistically, usage of Google satellite images were not feasible in the task of image object classification. An alternative was required.

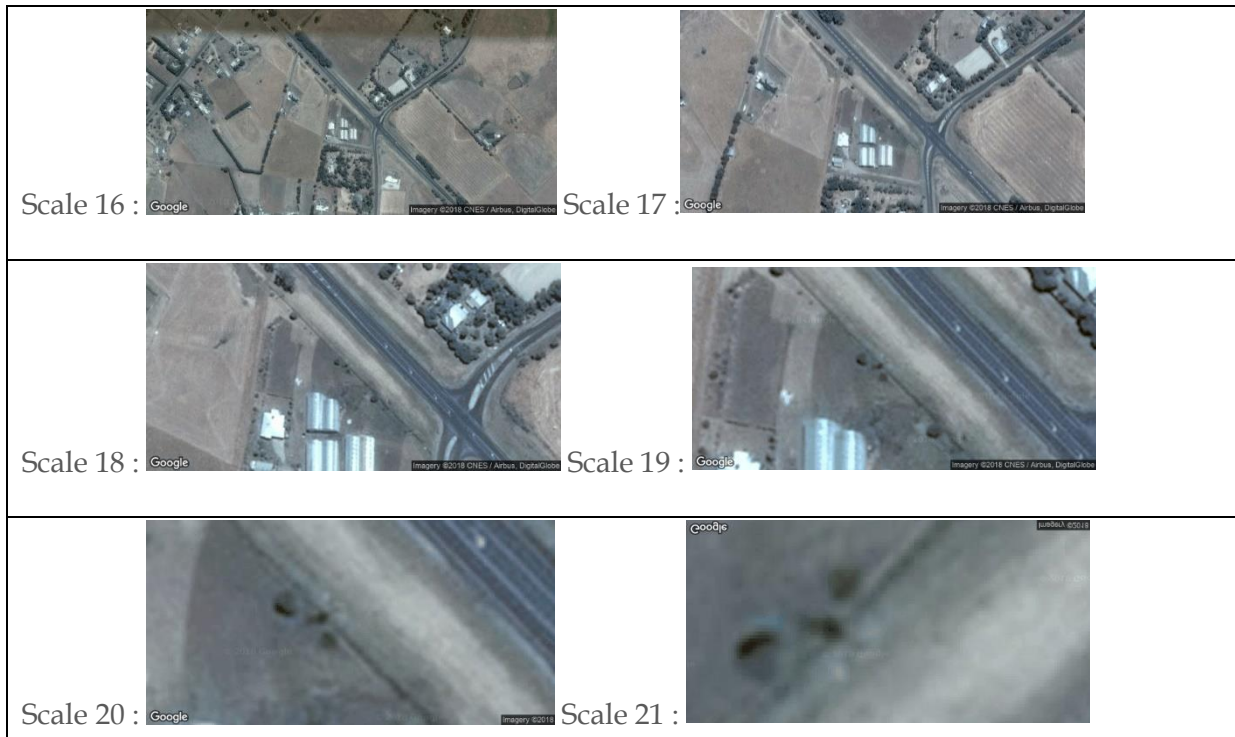


Table 1 - Google Satellite Images – Zoom scale 16 to 21.

Task C – Download Google Street View Images of Coordinates

The alternative to Google Satellite images was Google Street View API. Instead of choosing a zoom scale for Google Street View, users opt for a “pitch” degree view of the coordinates. The example below is of latitude and longitude values -38.0268632392842, +144.1851624938. It is evident that as this location there exists no electricity asset. This example shows a clear business case in machine learning image classification and reporting a data error at the coordinates provided.

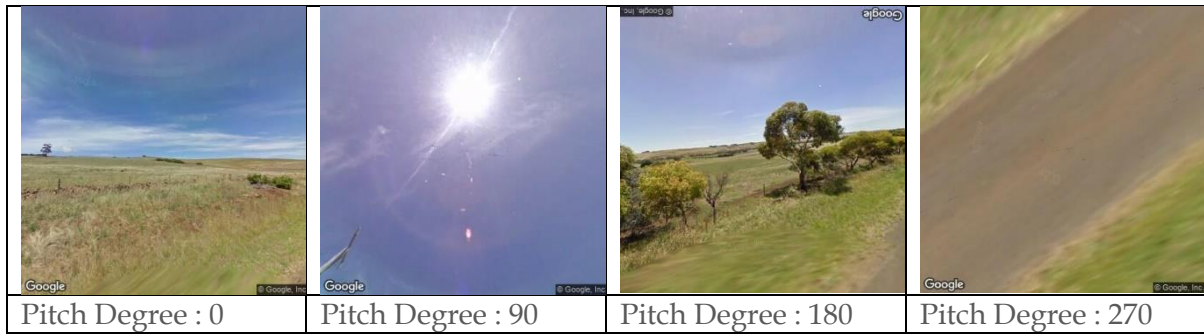


Table 2 - Google Street View Images – Pitch degrees from 0, 90, 180 to 270.

Task D – Altering Google Street View Images To Enhance Object Classification

Classifying the existence of a transformer located on a wooden power pole in a Google Street View image with a scale of 600 pixels x 300 pixels, it not a simple task. Most of the coordinates in the sample data set also contain grass, trees, houses, vehicles, roads, etc. Even when a transformer exists in an image, if the existence of a larger object such as a tree, road or house is present, machine learning classification will likely choose to classify the image as whatever the largest object is.

In this scenario, we need to find an innovative method to give preference to likely transformer objects.



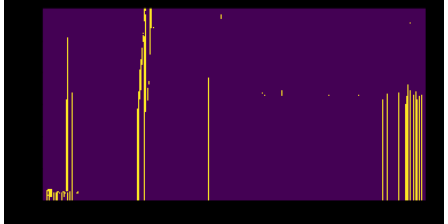

Black and White :		Transformer Precision = 0%
Outline :		Transformer Precision = 37%
Horizontal Lines :		Transformer Precision = 43%
Background Whitening :		Transformer Precision = 62%

Table 3 - Python Image Altering techniques – Improving Detection of Transformers.

Task E – Migrate to TensorFlow from SciKit Learn Packages

In testing Neural Networks image classification precision, recall and accuracy, usage of the Python SciKit Learn Neural Network package experienced memory performance issues. An alternative to SciKit Learn is TensorFlow. TensorFlow is often described as a lower level, more configurable package for neural networks, in comparison to SciKit Learn.

"TensorFlow is more of a low-level library; basically, we can think of TensorFlow as the Lego bricks (similar to NumPy and SciPy) that we can use to implement machine learning algorithms whereas scikit-learn comes with off-the-shelf algorithms." --

<https://sebastianraschka.com/faq/docs/tensorflow-vs-scikitlearn.html>

"SciKit Learn" package found 23% of the transformers and was correct 62% of the time.

Object	Precision	Recall
Transformer	62%	23%

Table 4 - Classification Matrix 1 – SciKit Learn

For the same training and test set for the TensorFlow package of neural networks, 42% of the transformers were found, an improvement of 19% and the precision of 62% was maintained.

Object	Precision	Recall
Transformer	62%	44%

Table 5 – Classification Matrix 2 - TensorFlow

In addition to improved precision, TensorFlow package enabled scaling the number of neural networks training runs. Epochs is a term used to describe the number of training runs you perform. The table below shows the best precision (% of transformers identified) and recall (% of predictions correct) was highest at exactly an epoch number equal to the number of images in the training set + 1;

Object	Epochs	Precision	Recall
Transformer	240	34%	42%
Transformer	230	34%	56%
Transformer	225	34%	48%
Transformer	221	48%	42%
Transformer	220	52%	30%
Transformer	219	52%	39%
Transformer	218	45%	42%
Transformer	215	31%	56%
Transformer	210	34%	45%
Transformer	200	48%	47%

Transformer	180	3%	100%
-------------	-----	----	------

Table 6 – Classification Matrix 3 – Epoch Training Runs

Task F – Conclusion

Having settled on image alteration method and optimum number of epochs, other tuning parameters of interest are specifying the number of pixels in the image and scaling the image size similar in number of neural networks hidden layers. This is where a scale up of memory size is recommended. At 8GB of memory, the neural network layer number ratio to Google Image pixel number was a 5% match. Dima Shulga's Image 2 below, is a visualization of a Neural Network. The image pixels represent the input layer. Hidden layers reside in the middle of the neural network. The output layer is the number of image classification categories known, performance generally improves by introducing balanced numbers of categories. This can prevent the neural networks algorithm opting to predictions favoring the largest known objects.

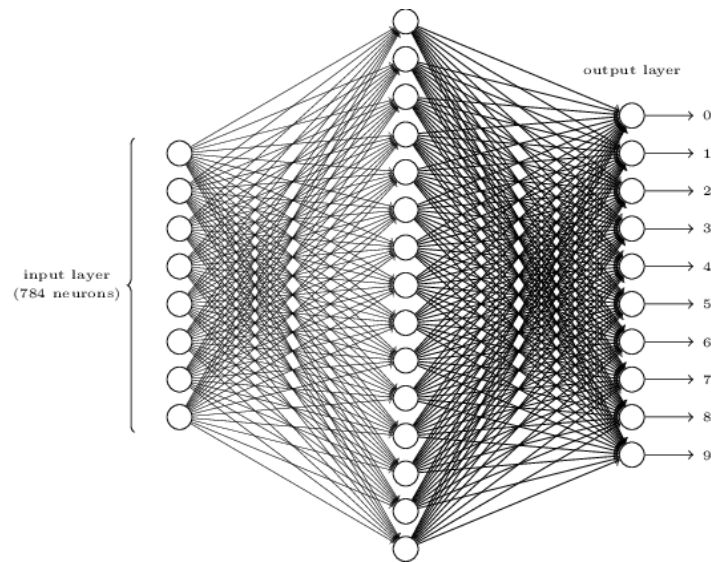


Image 2 – Neural Network Visualisation – Dima Shulga -

<https://towardsdatascience.com/exploring-activation-functions-for-neural-networks-73498da59b02>

References

<http://www.deakin.edu.au/students/studying/study-support/academic-skills/report-writing> Report Writing, Deakin University, Last Updated May 17, 2016.

http://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

http://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

<http://geopandas.org/> GeoPandas 0.4.0, GeoPandas Developers, 2013-2018.

<https://www.geeksforgeeks.org/python-get-google-map-image-specified-location-using-google-static-maps-api/> Geeks for Geeks, Python, Get a Google map image of a specified location using Google Static Maps API, 2017.

<https://developers.google.com/maps/documentation/streetview/usage-and-billing> Google Maps Platform, Street View Static API Usage and Billing, Jul 2018.

https://docs.opencv.org/trunk/dd/dd7/tutorial_morph_lines_detection.html OpenCV, Open Source Computer Vision, Extract horizontal and vertical lines by using morphological operations, Jan 2019.

<http://answers.opencv.org/question/201115/removing-image-background-from-image-with-python/> OpenCV, Open Source Computer Vision, Removing image background from image with python, Oct 2018.

https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_canny/py_canny.html OpenCV, Open Source Computer Vision, Canny Edge Detection, 2013.

<https://towardsdatascience.com/exploring-activation-functions-for-neural-networks-73498da59b02> Towards Data Science, Exploring Activation Functions for Neural Networks, Dima Shulga, Jun 2017.

<https://sebastianraschka.com/faq/docs/tensorflow-vs-scikitlearn.html> Machine Learning FAQ, Sebastian Raschka, 2013-2019.

https://www.tensorflow.org/tutorials/keras/basic_classification TensorFlow, Train your first neural network, basic classification, Jan 2019.