Network Enabled - Unresolved Residual Analysis and Learning (NEURAL)

Dwight Temple
ExoAnalytic Solutions Inc.

Mark Poole
Nou-systems

Matt Camp
SAIC

ABSTRACT

Since the advent of modern computational capacity, machine learning algorithms and techniques have served as a method through which to solve numerous challenging problems. However, for machine learning methods to be effective and robust, sufficient data sets must be available; specifically, in the space domain, these are generally difficult to acquire. Rapidly evolving commercial space-situational awareness companies boast the capability to collect hundreds of thousands nightly observations of resident space objects (RSOs) using a ground-based optical sensor network. This provides the ability to maintain custody of and characterize thousands of objects persistently. With this information available, novel deep learning techniques can be implemented. The technique discussed in this paper utilizes deep learning to make distinctions between nightly data collects with and without maneuvers. Implementation of these techniques will allow the data collected from optical ground-based networks to enable well-informed and timely the space domain decision making.

1. INTRODUCTION

In recent years, convolutional neural networks (CNNs) have enabled tremendous forward progression in an array of classification problems such as discrimination between types of flowers, recognizing individual people's faces, dog versus cat recognition, and even cucumber health and quality [1]. Because convolutional methods are readily applied to various domains with disparate applicability, transferring image classification into the space domain is a logical step. Deciding which application to tackle can be daunting, but the most useful ones appear to be the most obvious.

When observing azimuth-elevation plots of data, a subject matter expert can detect abnormalities in a completely ballistic dataset. Contrarily, translating this ease of detectability to a sorting algorithm or anomaly detector is orders of magnitudes more difficult. Because azimuth and elevation are results of non-linear dynamic motion, there is not a simple regression that can be performed to detect abnormalities; it is the trained human-intuition that can make the distinction.

This problem will be tackled two-fold for redundancy and robustness. First, an image classification system will be developed to attempt to make distinctions between geostationary satellites with and without an impulsive maneuver according solely to the azimuth-elevation plot. Second, a data based classification neural network will be implemented to detect anomalies or maneuvers within the observed values of azimuth and elevation as a function of time.

2. DATA GENERATION

To parallel as realistic geosynchronous orbits as possible, reasonable data must be generated to feed into the algorithm. For this aspect, using a high-fidelity orbit propagator, randomly injected maneuver times, maneuver magnitudes, and inclinations, 20,000 representative orbit data sets were generated for training, testing, and validation. To initialize testing, short-duration, impulsive maneuvers were considered. These would have the most apparent effects on azimuth and elevation as a function of time; accordingly, these were most likely to elicit the most favorable results. After this analysis, low-thrust, non-impulsive maneuvers will be analyzed for further model validation.

Below, examples of generated azimuth and elevation plots can be observed for various satellites. Each is specified as “maneuver” or “no maneuver” for clarity. Specifically, the maneuvers in the accompanying Fig. are two 15-minute burns of a total sum of 1-3 m/s. These are characteristic of geostationary station-keeping maneuvers and can be observed on a weekly basis. Fig. 1 and 2 portray geostationary satellites without and with maneuvers, respectively. Fig. 3 and 4 have non-normalized plot data and as a result contains images with excessive whitespace. Additionally, Fig. 5 and 6 normalize the plot values to a range of -1 and 1 to ensure that it occupies the entire image.

Fig. 1. "No Maneuver" Azimuth Elevation

Fig. 2. "Maneuver" Azimuth Elevation

Fig. 3. "No Maneuver" Azimuth Elevation with Visibility Constraints (not normalized)

Fig. 4. "Maneuver" Azimuth Elevation with Visibility Constraints (not normalized)

Fig. 5. "No Maneuver" Azimuth Elevation with Visibility Constraints (normalized)

Fig. 6. “Maneuver" Azimuth Elevation with Visibility Constraints (normalized)

3. IMAGE CLASSIFICATION

To classify images, a CNN was developed and deployed using Keras as a Tensorflow interface in Python [2][3]. The architecture of the model consists of convolutional, max pooling, and densely connected layers and can be observed below in Fig. 7 and 8.

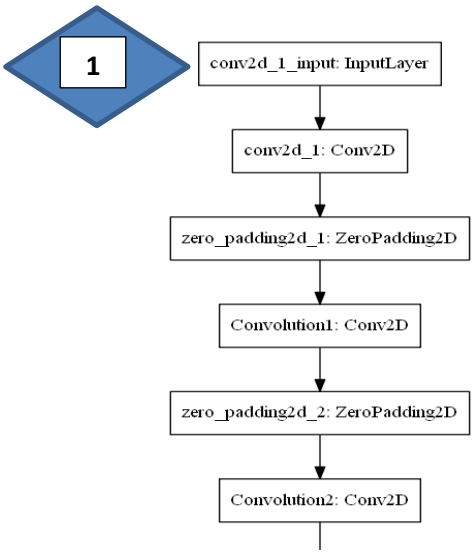
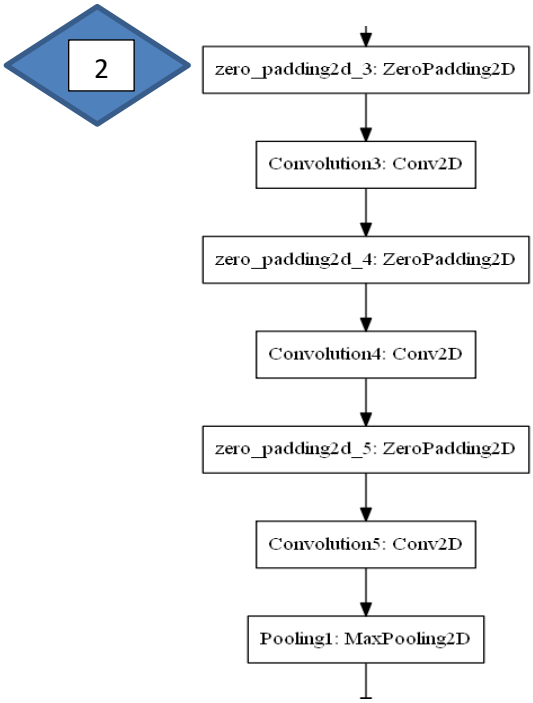
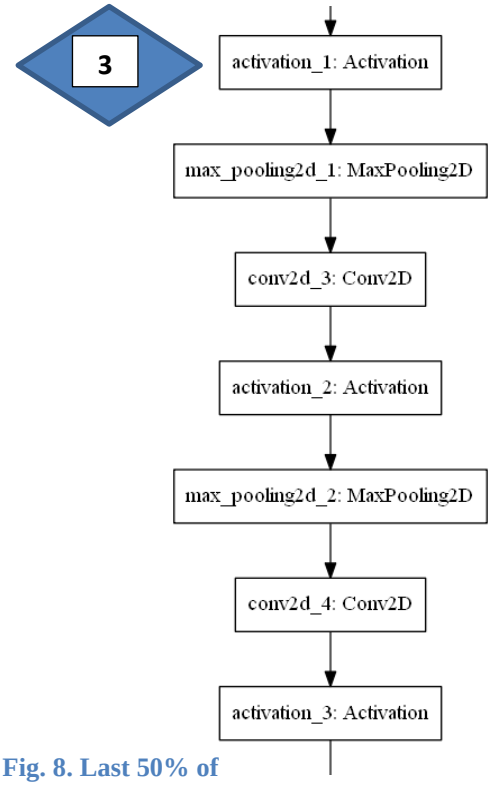


Fig. 7. First 50% of Model





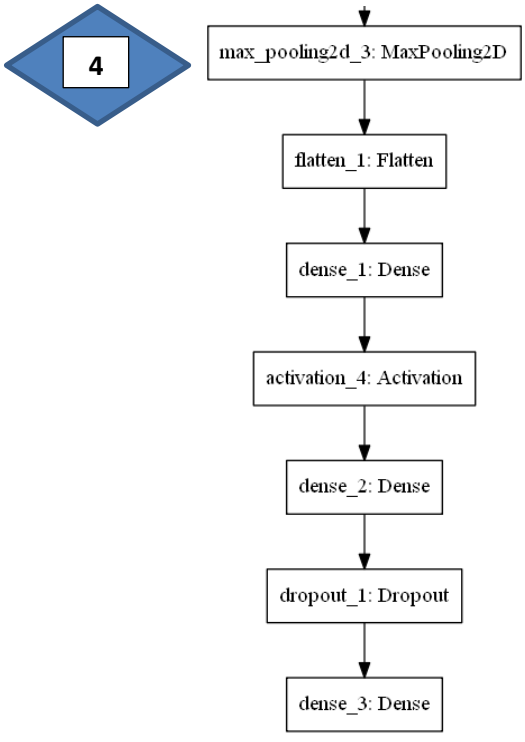


Fig. 8. Last 50% of Model

CNNs perform optimally on image-based classification problems due to their intrinsic feature-extraction capabilities [4][5]. Because of this, feature engineering and extraction can be assumed away to decrease the complexity of the model inputs. For example, when undergoing numerous convolutions, features extracted may include edges, color distinctions, or the smoothness of curve, all of which serve as potential indicators for distinction. In Fig. 5-8 below, the input images can be observed after each convolution in the layer as features that are more abstract are extracted.

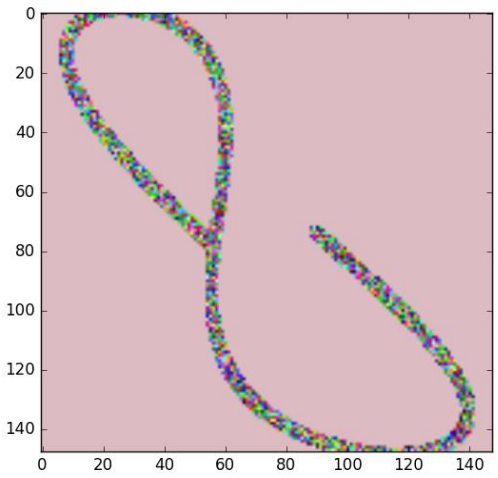


Fig. 5. Image after 1 Convolution

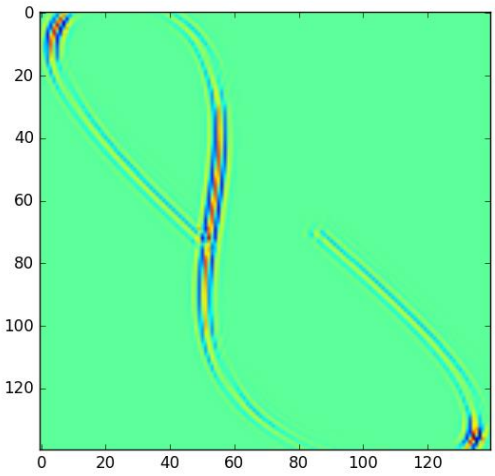


Fig. 6. Image after 1 Convolution and 1 Filter

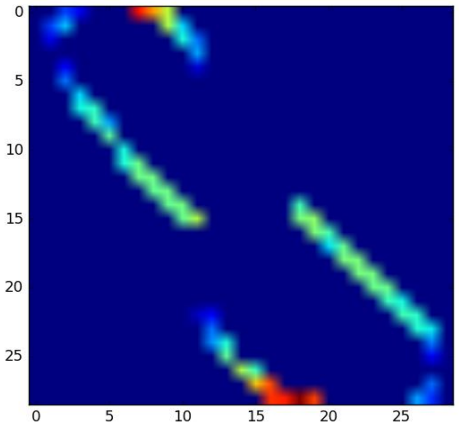


Fig. 7. Image After 1 Convolution 1 Activation and Pooling

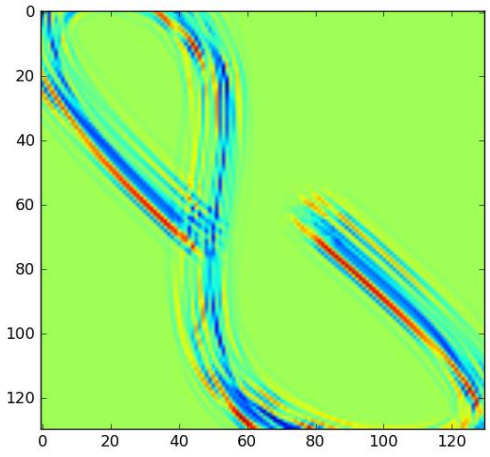


Fig. 8. Image after Multi-Dimensional Filter (10x10x10)

As one can note, the above Fig. are abstracted versions of the input images. Fig. 5 depicts the first two-dimensional convolutional layer. It serves as a basis to detect the easily discernable image features (such as the black plot on the white background). Fig. 6 and 7 portray additional abstract features created from additional filters and multi-dimensional convolutions in intermediate steps. Note that in Fig. 8, since the filters were multi-dimensional, the image displays a flattened depiction of this higher-dimensional data (namely the 10 multi-colored plots). Finally, Fig. 7 shows the combination of several layers after convolution and pooling (summing the maximum values in each layer). Shown in red are regions of peak interest to the CNN. Notably, this is an example of a maneuver azimuth-elevation plot and these highlights are indicative of key distinctive features relevant to discriminating between the two categories.

After the data generation, and with a basic understanding of the internal model operations, the fully realized model is capable of being trained, validated, and tested against numerous representative data samples. Specifically, this CNN was trained using 16,000 samples, validated using 2,000, and tested using an additional 2,000.

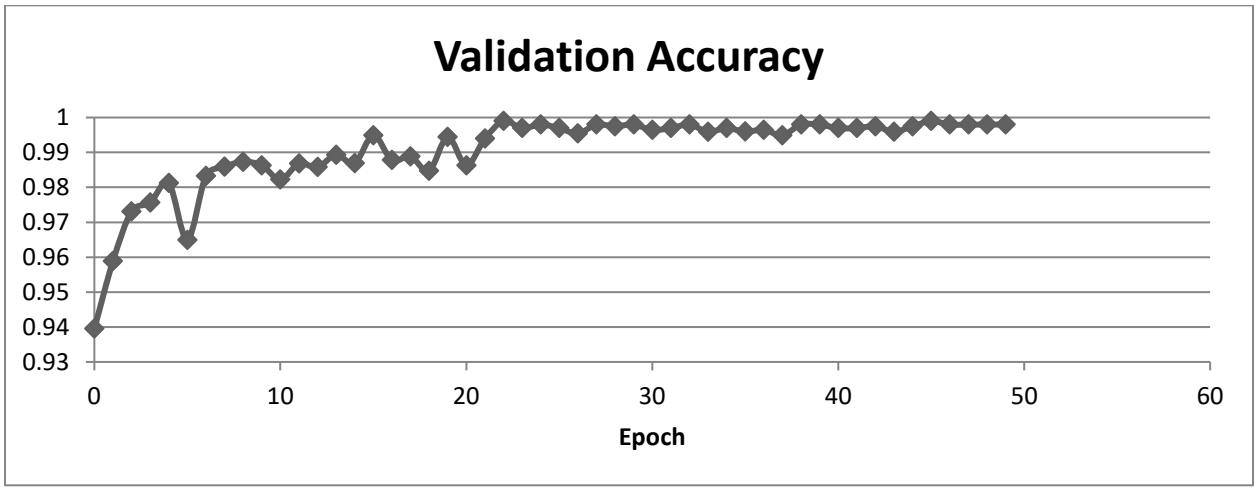


Fig. 9. Validation Accuracy with dropped sections of data

While the results on the previous azimuth and elevation Fig. are promising, they were not entirely representative of data that would be collected from optical ground sensors. Realistic collections will have uncertainties and non-continuous collection intervals. Moving forward, these perturbations were modeled through incorporating +/- 5 micro-radians of positional uncertainty and modeling elevation, Earth-shadow, and solar exclusion constraints. The line-of-sight uncertainty results in a positional deviation of up to ~180 meters from actual position at geostationary orbit. In these examples, an increased cadence of 10 seconds between measurements was explored, since that is not atypical of ground-based optical collections. However, this cadence was later decreased back to one-minute due to the indifference in appearance of plots for training and unnecessary increase in data generation time. For simplicity, one sensor was utilized as well. Moving forward, multiple sensors at disparate locations can be implemented to further mirror optical ground networks.

On the first rendition of this model, it performed quite poorly due to the relatively small size of the plot compared to the available whitespace. This large amount of whitespace disallowed the convolutional filters to obtain as many features as a more normalized region. Because of this, identical data was generated, normalized to the plot space, and then run through the convolutional network again. Disconcertingly, the model still performed below expectations. Upon this realization, changing back to a continuous line plot instead of a scatter plot was implemented. It was inferred that perhaps the dotted pattern of a scatter plot was being poorly interpreted by the CNN. This change improved the model performance as depicted in Fig. 10. Inclusion of the entire azimuth elevation plot resulted in near perfect accuracy; realistic observation conditions resulted in a validation accuracy of comparable measure. This finding is heartening because it will allow for a more one-to-one transfer of applicability for implementation of the algorithm.

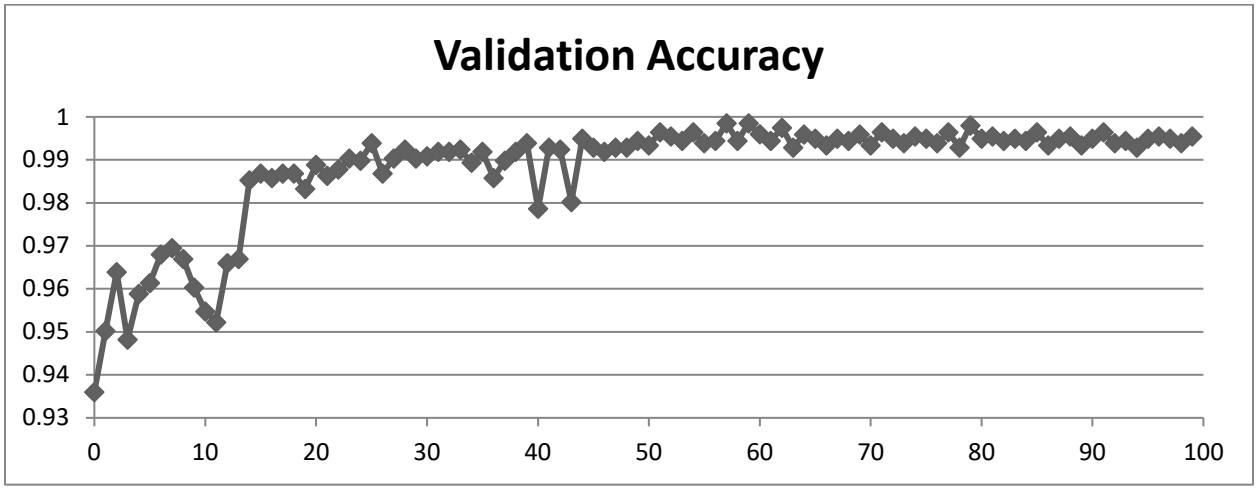
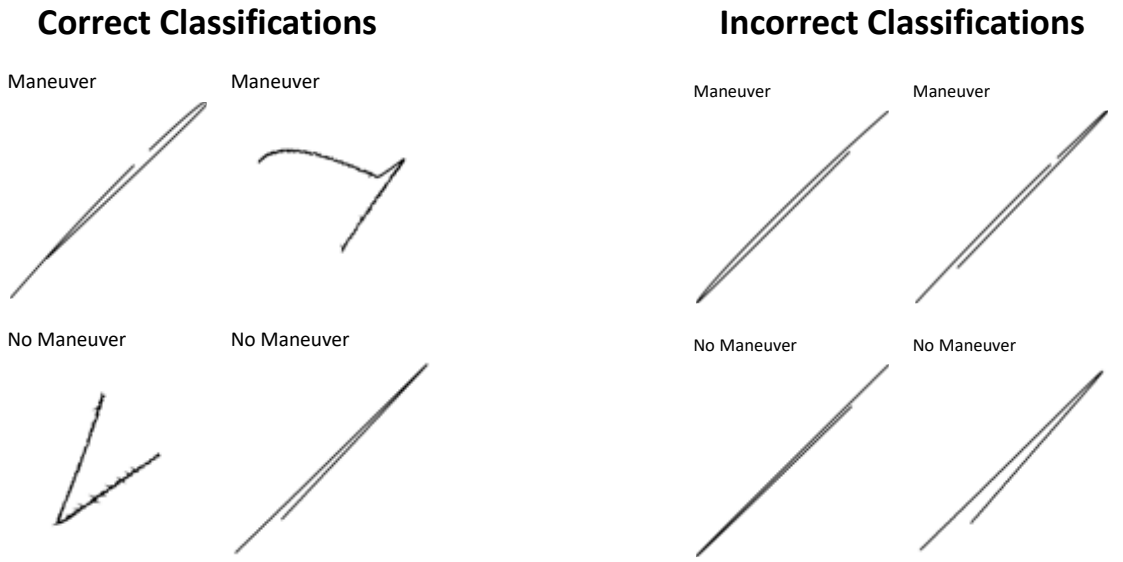


Fig. 10. Validation Accuracy with Realistic Solar Exclusion



Once adequate model performance was obtainable using the realistic dataset. A parametric hyperparameter search was implemented to discover an ideal combination of dense neurons, and convolutional layers. To reduce computational time, three combinations of each were iterated. Table 1 below depicts the results obtained in test score percentage from each of these variations.

Table 1. Hyperparameter Grid Accuracy

Hyperparameter Grid	2 Layers	4 Layers	8 Layers
512 Neurons	93.34%	50.19%	50.29%
1024 Neurons	93.64%	50.21%	50.21%
2048 Neurons	95.56%	50.22%	50.22%

For illustrative purposes, the hyperparameter search was completed using only 25 epochs for each rendition. Notably, increasing convolutional layers did not increase the accuracy of the model; doing so substantially decreased the goodness of the model. This effect is likely because the input plots are already in black and white space. Adding numerous convolutions decreases the resolution and interpretable features of the plots. Because the features are readily obtainable without added convolutions, fewer are better in this circumstance. Additionally, increasing the width of the model through increasing the number of dense neurons increases the accuracy of the model. While a width of 2048 neurons was tested, it would be interesting to explore expanding the width and comparing computational times to increase in accuracy.

4. A LOOK FORWARD

While the premise of this model is simple, its implications for SSA are unique in that it delivers the capability to detect sections of data that might contain maneuvers. However, this application has the potential to be extended in further support of SSA as well. These applications could be the estimation of maneuvers and detection of miniscule impulses.

Foremost is the application to maneuver estimation. Typically, to estimate maneuvers, observations must be available to perform orbit determinations on the data preceding and succeeding the potential maneuver. With this information, two state vectors can be used to compare magnitudes of potential change in velocities and their most likely epochs. However, utilizing a CNN to estimate the magnitude of maneuvers presents a novel and less computationally expensive method and could allow the verification of additional maneuver estimation techniques.

Secondly, applying a CNN to the detection of small maneuvers also shows promise. Training on enough various ballistic orbits allows the neural network to know how true ballistic orbital motion appears in azimuth-elevation space. Likewise, when a deviation from this smooth behavior appears, it will no longer fit the mold of the CNN.

Finally, the comma separated value (CSV) files containing the azimuth and elevation data can be fed directly into a CNN for binary classification. Implementation of this data classification method could be done to enable another classification method for further model robustness. It allows for the cross-validation of model types to ensure the most accurate classification is made prior to sending the information to another algorithm or a subject matter expert for additional analysis. For this implementation, primary feature extraction can be performed prior to model injection. This preprocessing simply extracted the azimuth and elevation derivatives and appended these values to the input dataset. Since the CNN will operate on numerical data, handcrafted features of importance are more vital to the performance of the algorithm. The time-rate-of-change of azimuth and elevation are highly indicative of non-ballistic motion occurring in the state-space.

5. WRAPPING UP

Convolutional neural networks have rapidly altered the horizon for scientific vistas. A plethora of new opportunities will unveil themselves as CNNs are applied to various classes of seemingly unsolvable problems. While beginning with the visual recognition of spacecraft maneuvers, progressing to the annotation of these maneuver locations, and concluding with the estimation of these maneuver magnitudes, the path forward for this instance is set. With time, additional SSA applications that are currently too arduous for automation will be tackled in an effective and efficient manner.

6. REFERENCES

[1]Sato, K. (2016, August 31). How a Japanese cucumber farmer is using deep learning and Tensorflow. Retrieved July 24, 2017, from https://cloud.google.com/blog/big-data/2016/08/how-a-japanese-cucumber-farmer-is-using-deep-learning-and-tensorflow

[2]Chollet, F. (2015). Keras. Retrieved July, 2017, from https://github.com/fchollet/keras

[3] Mart´ın Abadi, Ashish Agarwal, Paul Barham, Et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org

[4]Collet, F. (2016, June 5). Building powerful image classification models using very little data. Retrieved July 1, 2017, from https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

[5]Yung, J. (2017, May). Explaining Tensorflow Code for a Convolutional Neural Network. Retrieved July, 2017, from http://www.jessicayung.com/explaining-tensorflow-code-for-a-convolutional-neural-network/