Using Deep Convolutional Neural Networks to Automatically Detect Changes in Geographical Regions from Satellite Images

Senior Thesis

Integrated Science and Technology Program

at James Madison University

By

Paul Ritter

Matthew Gish

Connor Hite

under the faculty guidance of

Dr. Anthony Teate

Senior Capstone Project

May 8, 2021

Submitted by:

Paul Ritter	*Paul Ritter*
Connor Hite	*Connor Hite*
Matthew Gish	*Matthew Gish*

Accepted by:

Dr. Anthony A. Teate	*A. A. Teate (5/8/2021)*

Table Of Contents

Introduction

Using an artificial intelligence and machine learning focused approach for geographical analysis offers a more robust and expedited means for processing information on a larger scale. Previously, all analysis was done manually by personal interaction with data. Humans have physical limitations in what they can see in data, and can focus only on one image at a time. The objective was to develop a system for processing and analyzing multiple sources of data using the appropriate machine learning models. Using a Convolutional Neural Network, we were able to build a high accuracy model that classifies images based on various classifiers, such industrial and residential buildings, annual crop, highway, and pasture. Using the same model, we were able to successfully detect change over time on a large scale, where significant change in images of the same region from two different years would be recognized, ultimately satisfying our goals.

The New Age of Satellite Imagery

Satellite imagery is an invaluable resource when applying new-age computer vision techniques like object classification and change detection. First commercialized in 1984, we have seen much growth in the capabilities in modern day satellites than those of the 20th century. Furthermore, we did not have nearly the amount of access that we do today. Much of this changed with passing the Land Remote Sensing Policy Act of 1992. This promoted research and public sector applications by allowing commercial companies to launch their own satellites and sell imagery. Today, almost 6,000 satellites orbit our planet, half of which are no longer operational, each serving their own purpose in a wide variety of niche applications. Some equipped with cameras, as we will utilize in this project, others enabling the communication and navigation infrastructures that we rely on so heavily today.

In our research, we primarily focused on open-sourced, earth observation satellites. Even with those requirements, there are many options to choose from. NASA, Copernicus, Sentinel Hub, and even google earth have archives of satellite imagery available, each with their own advantages and disadvantages. We chose data from the Sentinel 2 satellite, through the open source archives available in EO browser. There are many reasons that this dataset rose above the rest in terms of functionality, but that will be covered in the Testing Phase section.

Sentinel-2 Bands	Central Wavelength (µm)	Resolution (m)
Band 1 - Coastal aerosol	0.443	60
Band 2 - Blue	0.490	10
Band 3 - Green	0.560	10
Band 4 - Red	0.665	10
Band 5 - Vegetation Red Edge	0.705	20
Band 6 - Vegetation Red Edge	0.740	20
Band 7 - Vegetation Red Edge	0.783	20
Band 8 - NIR	0.842	10
Band 8A - Vegetation Red Edge	0.865	20
Band 9 - Water vapour	0.945	60
Band 10 - SWIR - Cirrus	1.375	60
Band 11 - SWIR	1.610	20
Band 12 - SWIR	2.190	20

Figure 1: Chart depicting the 13 spectral bands.

With the massive amount of data at our fingertips, the question ceases to be “How do we get access to the desired data?” Instead, we must ask ourselves what data is important for the classifications we intend to perform. Many raster files, including Color digital photos, are made up of multiple layers called bands. These can be described as a collection of images taken simultaneously of the same place. The bands and their basic descriptions can be found in **Figure 1**. Each of these bands contains specific sampling from the electromagnetic radiation (EMR) spectrum reflected or emitted from the Earth's surface.

When using this imagery, you can pick any combination of these bands according to what you intend to measure. For our model, we tried several permutations of these bands, eventually deciding on bands 2-4. RGB gave us the highest accuracy in classification, and is the most recognizable to the human eye. Other applications of these bands include climate change analysis. To measure this, we would use the first band, Coastal Aerosol, and the ninth band, the Water Vapor band. Bands 5-8a are used to measure regional chlorophyll content, which can be used to evaluate crop health. The three SWIR bands at the bottom stand for Short-Wave Infrared bands. SWIR bands are particularly useful because they can penetrate cloud cover,

allowing us to continuously track attributes like soil quality, potential mineral ores, or even sub surface deposits. Having decided on RGB bands, the next step is to frame our problem to the needs of our stakeholders and align our approach accordingly.

Stakeholders

The primary stakeholder is the National Geospatial-Intelligence Agency (NGA), as James Madison University has a Memorandum of Understanding with them to perform research guided by their focus areas. Every aspect of this project was shaped and directed by the desired functionality listed on the NGA's website.

The NGA is a well-established intelligence organization that supports many different departments of our government. For example, they provide visualization for military branches and intelligence outfits alike. This can include changes in geography that will affect troop movements, or even tracking the creation of facilities in hostile territories. In a more civil capacity, they can provide valuable imagery for energy and environmental agencies, tracking deforestation and sweeping changes in our global ecosystem. A topical example of this is the amazon rainforest. Each day, 200,000 acres of rainforest are burned on account of our planet's love for overconsumption. These statistics can be tracked and measured only through the use of data provided by the NGA. Seeing as they are used both for combat support and general intelligence analysis applications, they have high demands for mass processing satellite imagery. They have been collecting and redistributing this sort of intelligence since the cold war era, so roughly the 1940's. The only difference is that then, they operated under the moniker National Photographic Interpretation Center, or NPIC for short.

By extension, the other stakeholders in the creation of this model includes any department in the federal government that may use these resources. Additionally, the list of stakeholders would include the american citizens that they are operating in the interest of. Now

that we have identified our objectives and deliverables, we must review the available Deep Learning techniques to decide on a course of action.

Overview of Deep Learning and Neural Networks

In order to fully understand the dynamic functionality behind the model that was created, a baseline comprehension of Deep Learning and Artificial Neural Networks is needed. Furthermore, in essence, " Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and functionality of the brain called Artificial Neural Networks" (Brownlee, 2020). These networks essentially aim to accomplish three main tasks, which include reading the data in various forms, training themselves to recognize patterns, and then producing a desired output, such as recommendations, predictions, and classifications. ANNs consist of "connected computational units or nodes called neurons," that are organized into various layers (Misra & Li, n.d.). Data passes through these layers in only one direction, which is often referred to as "feed-forward" data propagation. Each neuron associates a weight to itself based on the correctness of the task, where the weights of all the neurons within the same layer are linearly combined and passed through an activation function. For our model, we used a softmax activation function, which essentially presents the output classifications as percentages, where the ten percentages for the ten classifications in our model added up to 1 or 100 percent. Within the realm of Deep Learning, there are different types of neural networks that all have different operations. Being that the task at hand was image classification, the most feasible and effective choice was to build and utilize a Convolutional Neural Network. Convolutional Neural Networks,

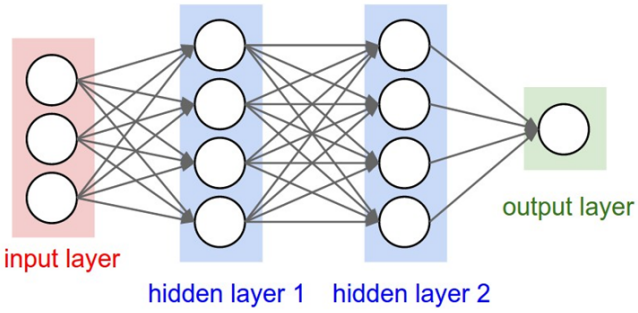


Figure 2: Diagram of Perceptron Model.

or CNNs, work best for image classification because they use two operations called ‘convolution’ and ‘pooling’. These operations reduce the image into its essential features and uses those features to truly understand and classify the image. Overall, a CNN “is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other” (Saha, 2020). An example of a CNN and its architecture can be seen below in **Figure 3** as it reads and classifies an image of two boats.

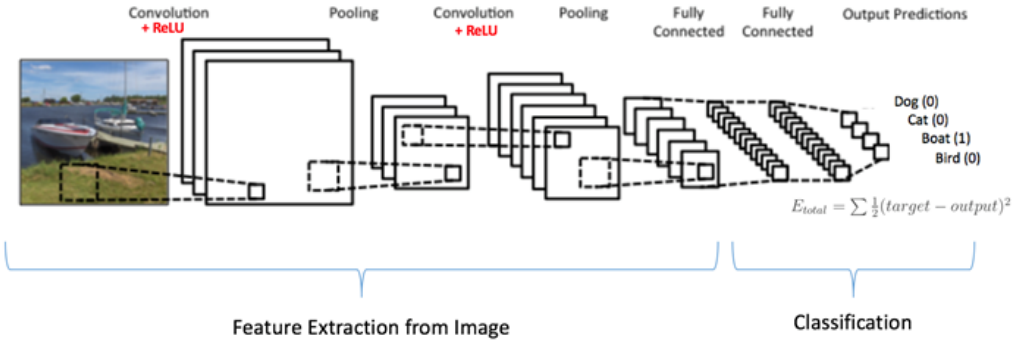


Figure 3: Convolutional Neural Network Example Architecture

Methods

For the purpose of organization and clarity, we first constructed a procedure that we would follow so that we were able to view what we have done versus what we still needed to accomplish. **Figure 4** shows a loose outline of how we attacked the problem at hand. The first task was to perform market research on existing satellite image processing softwares. Some examples of this are NASA, Copernicus, as well as Maxar. This allowed us to begin to identify the gap that our project intends to fill. By looking at other works

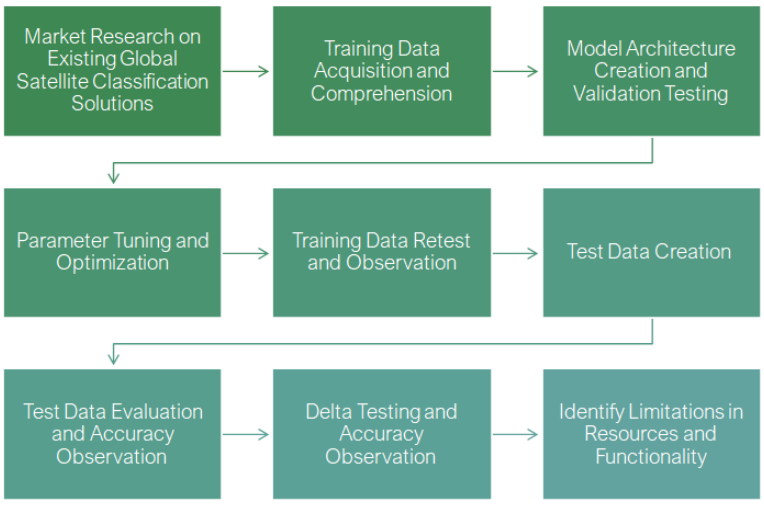


Figure 4: Project Procedure

and softwares that accomplish similar tasks, we were able to tailor our project so that we were attempting to fill a void in the market rather than replicating other works. Here, we found a variety of solutions that use satellite imagery to classify images with different attributes, but not many that can do so with the context of change detection. In terms of change detection, the objective was to build a model that could detect significant changes in the same image from two different points in time. The next step was to find a dataset to train our model with. This was accomplished by training the model architecture on the EuroSAT database, which we will cover later on. We then created an adequate model architecture for our Convolutional Neural Network, which acted as a working model to train and tune our parameters. It is important to understand that this was not our final architecture. From here, we used a tuner to tune the model's parameters until we got the best accuracy with our architecture. The next task at hand was to retest on our training data to evaluate the model on standardized data. Essentially, we pushed the training data back through the tuned model to verify the tuned architecture was in fact increasing in accuracy. Finally, we reached the point where we could integrate time series data for change detection purposes. To do so, we created our own testing set using time series imagery from the sentinel 2 satellite. We evaluated our model on the new test set and proceeded to delta testing, or change detection. This step happened to be the most time consuming step of the process. Essentially, most of the open-source databases and satellite imagery sources either did not have time series data or had the time series data but fell short of having a usable resolution. Another roadblock we encountered was locating regions of a country where evident change occurred between the two chosen points in time. Once a data source was found that had both requirements, we then had to figure out how to parse the same images from two different points in time. After due research, it was concluded that there were no open source scripts or tools that parsed images the way needed to. Instead, we wrote a script that cut each image into 25 64 x 64 pixel images, where the exact same cut images were present in both larger scenes. This eventually gave us the ability to send these images through our model and

observe if the model was able to detect the change in the image.This is where our model currently stands, so the next step is to identify future courses of action and limits of technology.

Training Phase

To train our model, we used the EuroSAT dataset. Eurosat is a collection of 27,000 satellite images labeled corresponding to their land cover classifier. There are 10 classifications in this dataset: Industrial Buildings, Residential Buildings, Annual Crop, Permanent Crop, River, Sea and Lake, Herbaceous Vegetation, Highway, Pasture, and Forrest. The creators of EuroSAT developed this dataset after recognizing that “current land use and land cover datasets are small-scale or rely on data sources which do not allow the mentioned domain applications” (Helber, Patrick, et al.). The original images were taken from the Sentinel-2 satellites launched by the European Space Agency(Sentinel-2). The EuroSAT dataset is composed of 64 x 64 pixel “patches” taken from these Sentinel-2 images. There are two versions of this dataset. The first version contains data from the 13 spectral bands mentioned in **Figure 1.** The version that we used was the RGB dataset that only contained data from the Red, Green, and Blue bands of the 64 x 64 pixel images. We chose to use the RGB Bands because it would make our model simpler, and the creators of EuroSAT had found that RGB was the best 3 band combination when comparing performance accuracy. To gain access to the dataset and train a convolutional Neural Network on the EuroSAT images, we used Tensorflow and Keras. Tensorflow is a machine learning library created by Google, and Keras can be used with Tensorflow as a high level machine learning API to make machine learning models easier to develop. We loaded the dataset from *TensorFlow Datasets,* a list of easily accessible datasets in TensorFlow, and partitioned the images into training, testing, and validation sets. 80 Percent of the images were used for training. Of the other 5400 images, we used 4400 for our testing set, and the remaining 1000 images for our validation set. The training set contains the images that our Convolutional Neural Network will be learning on while each of the 21600 images passes

through the model. The testing set has the images that the model has not seen, that the model will use to measure it's accuracy after training is over. A validation set has images that the model uses during training as an "unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters" (Shah).

We then began developing our simple implementation of a Convolutional Neural Network which can be seen in **Figure 5**. With TensorFlow Keras's model library, we were able to develop a sequential model using convolutional 2d, pooling, dense, and dropout layers. When training this model, each image we train on passes through once at a time, so our input shape is the shape of one image, 64 x 64 x 3, representing the 64 x 64 pixel size and the 3 dimensions: the red green and blue color channels. Each convolutional 2d layer is followed by a max pooling layer. Max pooling layers are used to simplify the output of each 2d layer, reducing the amount of information that will be going into the next layer while maintaining the most important information that has been extracted from the previous layers (Brownlee). We eventually hit our dense layers. Dense layers are fully connected layers that take the output of the previous layers that have been extracting features, and applies the information to the classification of the image. Our Global Average pooling layers are then used to simplify the output of the dense layers, generating one feature map for each corresponding category of classification(Papers With Code). Dropout is a layer used to randomly ignore units in our network to prevent the model from overfitting (Brownlee). Our last layer is a dense layer with 10 Nodes, using the softmax activation function. Each of those 10 nodes corresponds to one of our 10 EuroSAT classifications. Softmax is used as an output layer to predict the probabilities of each classifier (Brownlee). The sum of all probabilities is always 1.

```
model = models.Sequential()
model.add(layers.Conv2D(64, 3, activation='relu', input_shape=(64, 64, 3))),
model.add(layers.MaxPooling2D()),
model.add(layers.Conv2D(32, 3, activation='relu')),
model.add(layers.MaxPooling2D()),
model.add(layers.Conv2D(64, 3, activation='relu')),
model.add(layers.MaxPooling2D()),
model.add(layers.Dense(128, activation='relu')),
model.add(layers.GlobalAveragePooling2D()),
model.add(layers.Dense(256, activation='relu')),
model.add(layers.Dropout(rate= 0.2)),
model.add(layers.Dense(10, activation='softmax') )
```

Figure 5: Final model architecture.

When training our model on the 21,600 images, we used the 1,000 images from the validations set to monitor validation loss and accuracy in each epoch of training. One epoch in **Figure 6** can be seen as one cycle of training with corresponding statistics on it's results and efficiency. **Figure 6** shows the first 5 and the last 5 epochs of our 50 epoch training cycle. In the first 5 epochs the models validation accuracy jumps from 68 to 79% as the model begins to learn the data. Just in the first 3 epochs there is a large jump from 68 to 76%. By the 24th epoch, the model has almost reached the highest accuracy that it will reach, at nearly 91% validation accuracy. By 50 epochs the model increases 3 more percent to 94.5%.

```
Epoch 1/50
644/644 [==============================] - 9s 9ms/step - loss: 1.7531 - accuracy: 0.3056 - val_loss: 0.9322 - val_accuracy: 0.6840
Epoch 2/50
644/644 [==============================] - 5s 8ms/step - loss: 0.9889 - accuracy: 0.6290 - val_loss: 0.9419 - val_accuracy: 0.6350
Epoch 3/50
644/644 [==============================] - 5s 8ms/step - loss: 0.8491 - accuracy: 0.6830 - val_loss: 0.6917 - val_accuracy: 0.7600
Epoch 4/50
644/644 [==============================] - 5s 8ms/step - loss: 0.7648 - accuracy: 0.7201 - val_loss: 0.7110 - val_accuracy: 0.7400
Epoch 5/50
644/644 [==============================] - 5s 8ms/step - loss: 0.6770 - accuracy: 0.7555 - val_loss: 0.5925 - val_accuracy: 0.7930

Epoch 46/50
644/644 [==============================] - 5s 8ms/step - loss: 0.1386 - accuracy: 0.9524 - val_loss: 0.2296 - val_accuracy: 0.9260
Epoch 47/50
644/644 [==============================] - 5s 8ms/step - loss: 0.1202 - accuracy: 0.9578 - val_loss: 0.2402 - val_accuracy: 0.9370
Epoch 48/50
644/644 [==============================] - 5s 8ms/step - loss: 0.1173 - accuracy: 0.9570 - val_loss: 0.2246 - val_accuracy: 0.9380
Epoch 49/50
644/644 [==============================] - 5s 9ms/step - loss: 0.1182 - accuracy: 0.9611 - val_loss: 0.2482 - val_accuracy: 0.9310
Epoch 50/50
644/644 [==============================] - 5s 8ms/step - loss: 0.1115 - accuracy: 0.9604 - val_loss: 0.1930 - val_accuracy: 0.9450
```

Figure 6: First 5 and Final 5 Training Epochs

Testing Phase

Figure 7 depicts a visual representation of the validation accuracy from training going up over time. It is important to acknowledge that when trying to run the model longer, even up to 100 epochs, the validation accuracy does not go over 95%. This tells us that with the simplicity of our model, it does most of it's learning in the first 20 to 25 epochs and is not capable of learning much more after that. **Figure 7** shows

```
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'],
         label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')
test_loss, test_acc = model.evaluate(images_test, labels_test, verbose=2)
print(test_acc)
```

```
169/169 - 1s - loss: 0.2214 - accuracy: 0.9315
0.9314814805984497
```

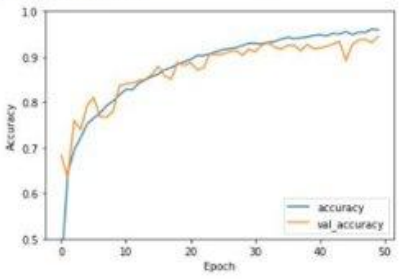


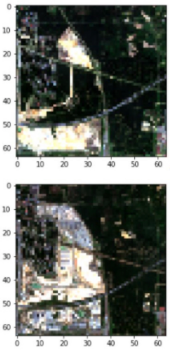
Figure 7: First 5 and Final 5 Training Epochs

that very well, where you can see the large spike in validation accuracy in the beginning, but as the training goes on, the validation accuracy begins to plateau. Above the plot in **figure 7** the model's accuracy of 93% can be seen after being tested on our testing set with the data that the model has never seen. That means that when predicting the classification of new, unseen EuroSAT images, our simple implementation of a Convolutional neural Network was able to correctly predict the classification 93.1% of the time. We can compare this to the creators of EuroSAT who reached an accuracy of 98.7% on the EuroSAT dataset when Training with ResNet-50, a much larger model with 50 layers and over 23 million trainable parameters.

Change Over Time

With the model being trained, and our group being happy with the accuracy, we were able to start looking for data to begin detecting change over time. The idea was to find time-series satellite images that show change over time, and detect the change based on the resulting classifications of the image when being run through our model. To acquire the data we used to detect change over time, we used EO Browser, a mapping software to view sentinel 2 data (“Sentinel-Hub EO-Browser3”). This was the only resource we found that made it easy to view change over time, which helped us find the best data to test on. Once we found the places that showed change, we then downloaded the best quality image from as far back as the data provided, along with another image of the same spot that was taken most recently. When downloading the data, EO Browser gave us the option to only download specific spectral bands. This took out one step of the preprocessing for us. Another way that this resource helped minimize the amount of preprocessing needed was by resampling the spatial resolution of each spectral band. Each band, red, green, blue, and the others listed before that we are not using in our model, have a spatial resolution defined by how much area can be viewed in one pixel. Sentinel 2 images have slightly different spatial resolutions for each spectral band, and it is useful to normalize these resolutions, but EO browser already does this for us. Since the

images only include the RGB bands, and the bands are already resampled for a normalized spatial resolution, the next preprocessing step we did was "subsetting", or in other words, cropping the two images according to geographic coordinates to ensure that both images are the same. We use the SNAP software (Sentinel Application Platform) to do this (STEP). With other data we attempted to use, we did not have the luxury of the data being resampled or the images only having the bands we needed, but SNAP has tools available for all preprocessing needs with sentinel-2 data. The images that we collected are quite large and cover a large area of land. In order to get the images in the format that coincided with the images used in eurosat, we parsed each larger image into 25 64 x 64 pixel images using a python program we created. We resized the larger image to be 320 x 320 pixels. Starting in the top left corner we cropped out the first 64 x 64 pixel image. We used loops to get each 64 x 64 square of this image without ever overlapping with previous 64 x 64 pixel squares. The two images in **figure 8** are from the same plot of land taken from the same plot of land in Malaysia. The image on top is from the area in 2017 and the image on bottom is from 2021. When running the first image through the model, the model was 65% sure that the image was of a highway, and 34% sure that the image was of industrial buildings. In this case, we can confirm that these are both correct when looking at the image, as well as referencing the larger image that it was taken from. On the second image, we can visually see change. We know from looking at the larger image that the prior industrial areas became populated with more industrial buildings, as well as a large plot of vegetation being taken out and used for



	AnnualCrop	Forest	HerbaceousVegetation	Highway	Industrial	Pasture	PermanentCrop	Residential	River	SeaLake
0	0.0000	0.0000	0.0525	65.5610	34.3371	0.0000	0.0128	0.0000	0.0367	0.0000

	AnnualCrop	Forest	HerbaceousVegetation	Highway	Industrial	Pasture	PermanentCrop	Residential	River	SeaLake
0	0.0000	0.0000	0.0009	3.4280	96.5707	0.0000	0.0000	0.0002	0.0001	0.0000

Figure 8: Predicted Probabilities of Time-Series images in Malaysia

industrial buildings. We can see that as industrial buildings became much more prominent in the image, the model recognized that, and was 96% sure that the image was of industrial buildings. It is important to acknowledge that each time we run the model from the start, there are differences in the way it predicts these images. However, each time it is correct. Sometimes the prediction acknowledges that there is vegetation in the image, and other times there are differences in the probability distribution between highway and industrial. However, it never gives a probability to something that is not in the image, and more importantly, every time we test on these images the model notices change. Especially that there is more industrial buildings in the image.

What Comes Next?

Overall, we are very happy with the functionality of our current model. We have cultivated a reliable architecture that regularly achieves high accuracy in classification, and detects change over time. The future of the model is entirely based upon desired functionality. The NGA is interested in a model that can sift through the entire globe and identify anomalies, or in other words, point out locations where the classification probabilities have changed. Our model proves that this is possible. For an automated procedure, we would need to finalize a simple data pipeline for the collection, storage, and classification of said images. For example, the classifications could be stored in s3 buckets on AWS, with each of the images identified by the dates and coordinates of origin. Only then can they be compared with the classifications of the same images for purposes of change detection.

Other desired enhancements for our model include further fine-tuning of our architecture, to be tested on a new dataset. Having a dataset that stores time series information will be invaluable for determining the efficacy of our model in different environments. Data like imagery directly from the NGA would be ideal for these purposes. This would also allow us to tune more parameters for greater precision on these datasets.

Next, we would like to expand the scope of our models classifications. The biggest inaccuracy that we currently have to deal with is that not every image we collect has an accurate classification for our model to label it as. This leads to misclassifications, lack of confidence in classifications, or even numerical bias to the existing classifications. The presence of non-distinct imagery that doesn't fit our 13 classifiers can actually skew our probabilities, inhibiting our ability to identify those classifications. When widening our capabilities by increasing the range of our classifiers, we would get more specific and meaningful classifications in the long run.

Lastly, we would love to pursue object detection. This is a feature that the NGA has expressed much interest in, but we have found not to be feasible with the resources available for free to the public. There are many libraries that would enable us to integrate object detection with our existing software, but the images are simply not high enough resolution to have objects detected. For reference, the images we collect are 64x64 images. Within each image, you can fit several large industrial buildings. To scale, if we wanted to identify a car, we would have to do so using anywhere from 4-16 pixels. This is not enough information for a model to identify objects. When these resources become available, our model will be more useful than ever, and the potential functionality will be revolutionary.

Bibliography

Brownlee, J. “A Gentle Introduction to Pooling Layers for Convolutional Neural Networks.” *Machine Learning Mastery*, 5 July 2019, https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks

Brownlee, Jason. “Dropout Regularization in Deep Learning Models With Keras.” *Machine Learning Mastery*, 26 Aug. 2020, machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/.

Brownlee, J. (2020, August 14). *What is Deep Learning?* Machine Learning Mastery. https://machinelearningmastery.com/what-is-deep-learning/

Brownlee, Jason. “Softmax Activation Function with Python.” *Machine Learning Mastery*, 23 June 2020, machinelearningmastery.com/softmax-activation-function-with-python/.

Helber, Patrick, et al. “EuroSAT: A Novel Dataset and Deep Learning Benchmark for Land Use and Land Cover Classification.” *ArXiv.org*, 1 Feb. 2019, arxiv.org/abs/1709.00029.

Misra, S., & Li, H. (n.d.). *Artificial Neural Network - an overview | ScienceDirect Topics*. ScienceDirect. Retrieved December 3, 2021, from https://www.sciencedirect.com/topics/engineering/artificial-neural-network

“Papers with Code - Global Average Pooling Explained.” *The Latest in Machine Learning*, paperswithcode.com/method/global-average-pooling/.

Saha, S. (2020, October 15). *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. Medium. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Shah, Tarang. “About Train, Validation and Test Sets in Machine Learning.” *Medium*, Towards Data Science, 10 July 2020, towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7.

Sentinel-2, sentinel.esa.int/web/sentinel/missions/sentinel-2.

“Sentinel-Hub EO-Browser3.” *Dashboard*, apps.sentinel-hub.com/eo-browser/?zoom=10&lat=41.9&lng=12.5&themeId=DEFAULT-THEME&toTime=2021-05-06T00%3A31%3A03.982Z.

STEP, step.esa.int/main/toolboxes/snap/.