8-2020

Identifying Smokestacks in Remotely Sensed Imagery via Deep Learning Algorithms

Kenneth Moss
kmoss8@vols.utk.edu

To the Graduate Council:

I am submitting herewith a thesis written by Kenneth Moss entitled "Identifying Smokestacks in Remotely Sensed Imagery via Deep Learning Algorithms." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Geography.

Qiusheng Wu, Major Professor

We have read this thesis and recommend its acceptance:

Liem Tran, Nicholas Nagle, David Page

Accepted for the Council:
Dixie L. Thompson

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

Identifying Smokestacks in Remotely Sensed Imagery via Deep Learning Algorithms

A Thesis Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Kenneth Tyler Moss

August 2020

ACKNOWLEDGEMENTS

Firstly, I feel the need to thank Dr. Hyun Kim, Dr. Kelsey Ellis, Dr. Paulo Raposo, the Geography department, and the Graduate school for working with me through my mid-semester change of program to the Geography department. I further would like to express my deepest appreciation to Dr. Qiusheng Wu, for stepping in as my advisor during a time of uncertainty, keeping me on track through a tough timeline and always being available for assistance, aiding my growth as a scholar. My committee, Dr. Nicholas Nagle and Dr. Liem Tran, for guiding me in my writing and methods, as well as through my coursework, alway providing me with direction and advice when requested. I cannot begin to express my thanks to Dr. David Page for giving me the invaluable resources and tools I needed to conduct my research, and networking me with the Oak Ridge National Lab, without which my project would not have been feasible. I'm extremely grateful for my parents, Lane and Bryan Moss, for the ever loving support through both of my degrees, and always being there for me during times of stress. And finally to thank you to ChilledCow, for providing me much needed serenity and sanity, through 24/7 lo-fi beats to work and relax to.

ABSTRACT

Locating smokestacks in remote sensing imagery is a crucial first step to calculating smokestack heights, which allows for the accurate modeling of dioxin pollution spread and the study of resulting health impacts. In the interest of automating this process, this thesis examines deep learning networks and how changes in input datasets and network architecture affect image detection accuracy. This initial image detection serves as the first step in automated object recognition and height calculation. While this is applicable to general land use classification, this study specifically addresses detecting smokestack images. Different dataset scenarios are generated from the massive Functional Map of the World dataset, ranging from two to sixty-two classes, and network architectures from recent studies are used. Each dataset and network is analyzed in their performance by way of F-measure. Image characteristics are also analyzed from images that were correctly/incorrectly labeled by the algorithms, providing answers on what images the algorithms best predict and what qualities the algorithms cannot discern. The smokestack's accuracy is reported at its highest through a five class training dataset, using an Adam Optimizer over six epochs. More or less classes returned lower scores, as did using the Stochastic Gradient Descent optimizer. Extended epochs did not return significantly higher or lower scores. The study concludes that while using more data can be effective in creating more accurate algorithms, using less data which is better structured for the problem at hand can have a greater effect.

Table of Contents

List of Figures

1. Introduction

Finding and labeling objects in imagery is no new task in remote sensing. Locating objects and classifying images or regions is a critical task that analysts have been performing since the beginning of the use of aerial imagery. From land use and scene classification (Kussul et al., 2017; Zou et al., 2015), to specific object detection (Durand et al., 2007; Mayer, 1999), it is necessary to understand what is in remote sensing images to further analyze and use the information within them. With the massive amounts of data being created with modern technology, relying on human sorting and classification is becoming slow and inefficient, thus there have been many efforts to automate the process (Ma et al., 2019; Zhang et al., 2016; Zhu et al., 2017). This automation has increasing use in finding objects within imagery as computers can be trained to identify the slightest details and patterns through thorough training.

Much work has been done in the way of classifying images with machine learning (Ball et al., 2017; Mayer, 1999; Song et al., 2019). The most relevant work for this study in particular is several large challenges that have been conducted to bring researchers together to build the most efficient algorithms. These challenges have been conducted on both ground truth imagery and remote sensing imagery and have resulted in widely popular algorithms that are still in use years after their creation in this quickly growing field. The large draw towards creating these competitions is the lack of large prelabeled datasets relative to deep learning, thus the creation of such datasets and researchers being invited to use the data in a competitive manner. It is not often that a researcher finds largely accessible datasets for a particular object they are trying to detect. Therefore, in building a deep learning algorithm, the largest amount of time is often spent collecting and sorting data; this expensive and time consuming process must be optimized.

The objective of this thesis is to evaluate the impact of input datasets and hyperparameters of deep learning algorithms on the smokestack detection accuracy (measured in F-score, explained in section 4.4). Figure 1 shows example photos of smokestacks as well as other images to be used throughout the training. During the training, the study observed effects on the training by viewing how different datasets, as well as changing key parameters, affect the accuracy to better understand the complexity of the algorithm. This can also serve as a platform to build from, as key government organizations are seeking ways to automatically extract information such as object heights from remotely sensed imagery; the process which begins with locating the objects of interest in imagery, directly building on the type of algorithm employed in this thesis. Among those government organizations, the Oak Ridge National Lab (ORNL) and the National Institute of Health (NIH) are currently seeking an automatic height retrieval algorithm for smokestacks. Their interest in smokestacks extends from NIH public health concerns, and the exposure of surrounding and to dioxins, a heavy pollutant expended from smokestacks during industrial processes. The NIH has tasked ORNL with finding the heights for these smokestacks, and therefore ORNL has offered use of their computer systems and datasets to begin exploring this research. This research can also be applicable to many other object height related tasks and will be valuable to other government agencies which may be interested in object heights (Kohlbrenner et al., 2009).

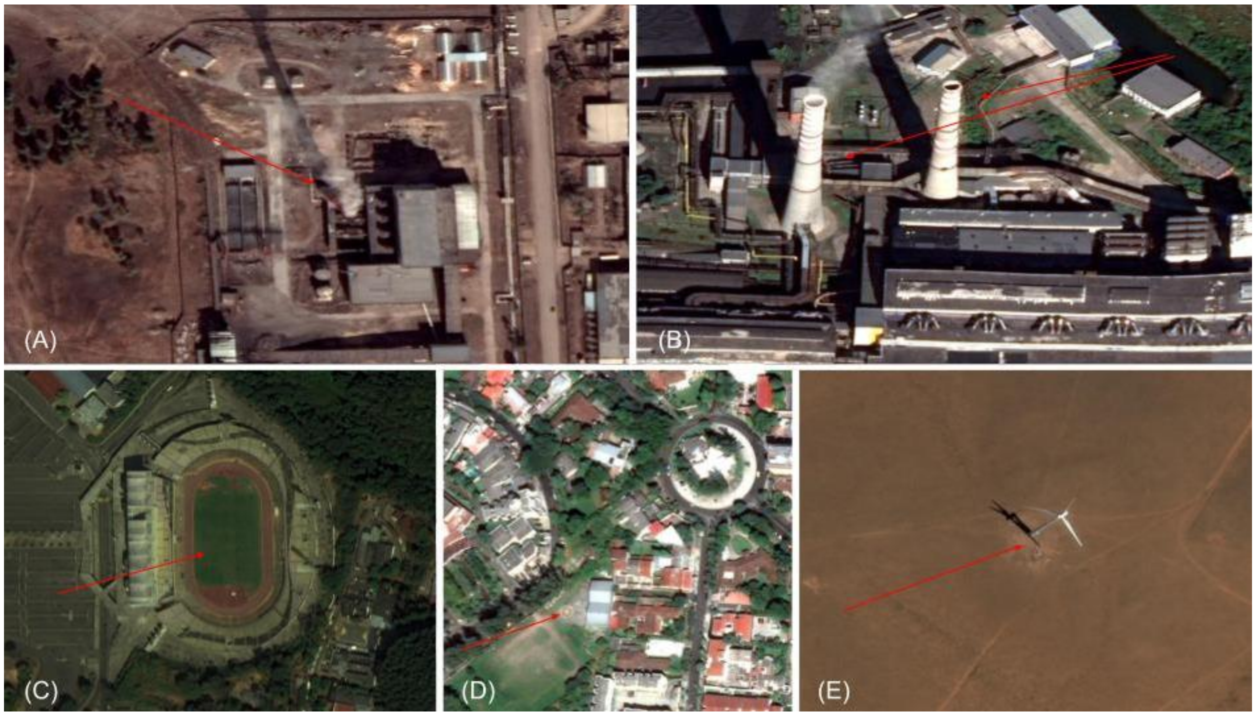


Figure 1. Example images from the fMoW dataset. Images A & B show smokestack images, while images C, D, and E show stadium, helipad, and windfarm examples, respectively. The fMoW consists of sixty-two classes with images varying in object size and quality.

2. Literature Review

2.1 Dioxin Health Interests

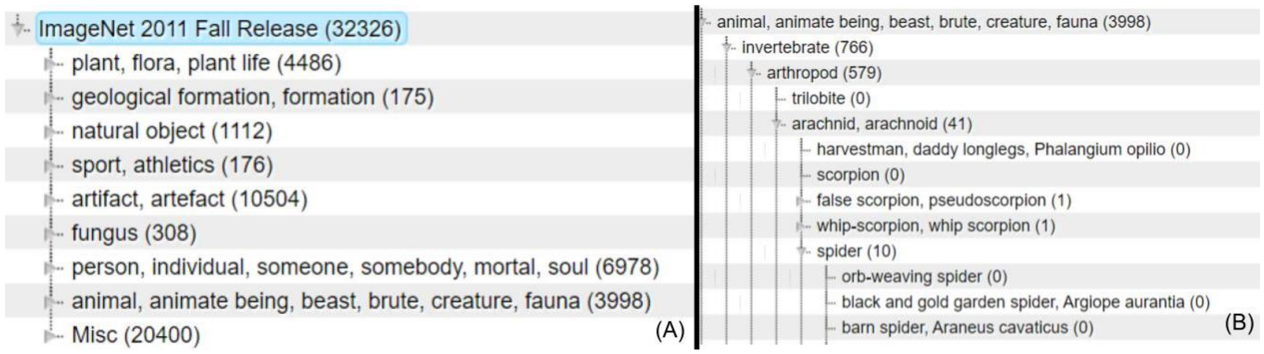
Growing interest in the modeling of dioxin spread from smokestacks has led to the need for locating such smokestacks in imagery. Dioxins are heavy chemical compounds that exude from smokestacks during industrial processes (mainly waste/fossil fuel burning), and they present environmental and health issues by polluting surrounding areas (Bertazzi et al., 1998; Pesatori et al., 1998). The toxins can lead to different types of cancers, developmental issues, and reproductive health problems, and can remain in surrounding animals and vegetation that often end up in human consumption, which creates a lasting impact as well as the immediate impact (Birnbaum, 1994). Modeling how these dioxins spread from smokestacks is a crucial step in deciding how to limit the spread of the toxins and contain them to limit human, vegetative, and animal exposure. In order to accurately model the spread of the dioxin pollutants, it is crucial to know the heights and locations of the smokestacks themselves, as the surrounding pollution area is dependent on the height above the ground and how the dioxins may spread while falling back to the ground. Locating smokestacks in imagery the first step to this problem.

2.2 ImageNet - ILSVRC Algorithms

ImageNet is a large database consisting of labeled ground truth images that are used for computer vision and artificial intelligence research (Deng et al., 2009). The database consists of over fourteen million images distributed through over 5,000 subsets, these subsets being derived from WordNet, a large database of subsets and synsets (80,000 synsets at the time of publishing) of nouns (Deng et al., 2009;

Fellbaum, 1998). The sysets range from plants, animals, natural objects, to people, artifacts, and geological formations (Figure 2). ImageNet images are used as pre-training images for deep learning algorithms as well, and therefore are necessary to this study.

The Large Scale Visual Recognition Challenge (LSVRC) hosted by ImageNet is based on the millions of pre-labeled ground images hosted by ImageNet, which are provided to researchers to train and test their algorithms to achieve the highest accuracies (Deng et al., 2009). Many of the state-of-the-art image classification and object recognition algorithms have resulted from the ImageNet LSVRC, such as AlexNet, ResNet, GoogLeNet, ZFNet, and DenseNet (He et al., 2016; G. Huang et al., 2017; Krizhevsky et al., 2012; Szegedy et al., 2015; Zeiler et al., 2011). These Convolutional Neural Networks (CNN), are deep learning algorithms which placed high in the LSVRC challenge and some of which built off the previous years' winners as well to create more efficient algorithms.


ImageNet 2011 Fall Release (32326)
- plant, flora, plant life (4486)
- geological formation, formation (175)
- natural object (1112)
- sport, athletics (176)
- artifact, artefact (10504)
- fungus (308)
- person, individual, someone, somebody, mortal, soul (6978)
- animal, animate being, beast, brute, creature, fauna (3998)
- Misc (20400)

(A)

animal, animate being, beast, brute, creature, fauna (3998)
- invertebrate (766)
  - arthropod (579)
    - trilobite (0)
    - arachnid, arachnoid (41)
      - harvestman, daddy longlegs, Phalangium opilio (0)
      - scorpion (0)
      - false scorpion, pseudoscorpion (1)
      - whip-scorpion, whip scorpion (1)
      - spider (10)
        - orb-weaving spider (0)
        - black and gold garden spider, Argiope aurantia (0)
        - barn spider, Araneus cavaticus (0)

(B)

Figure 2. (a&b). The ImageNet Synset structure is divided into major categories (a), that are further broken down into more specific subsets and groups. such as groups of animals, arachnids, and spiders, and specific spider types (b).

2.3 fMoW Challenge Data

The Functional Map of the World (fMoW) challenge is a similar challenge to the ILSVRC, providing researchers with training and testing data to implement their original algorithms on. However, the fMoW challenge specifically focused on the labeling of remote sensing images (Christie et al., 2018). The fMoW challenge was also only run for one year in 2017, as opposed to the 7 year annual run of ImageNet challenge (2010-2017). Researchers and teams were provided with a baseline algorithm, using a modified DenseNet architecture (Christie et al., 2018). The training and testing data provided has been made publicly available after completion of the challenge, which allows continuous advancement to be made on the achieved results by competitors.The challenge data has prompted many researchers to continue to develop algorithms on the data while pursuing higher accuracies and lower training times.

2.4 Deep Learning CNN Design

Deep learning is a term which encompasses many different disciplines, all working towards a similar goal of more accurate machine learning. Convolutional neural networks (CNNs), autoencoders, Boltzmann Machines, and VGG are a few of the major examples of deep learning methods (Guo et al., 2016; Zhu et al., 2017). CNNs, as the name would suggest, consist of multiple layers of convolutional layers that form a network roughly resembling that of the neural system of humans (Shanmuganathan, 2016). The layers perform convolutions, or filters, over a certain area of the image, and move on to the next area, until covering the entire image area. Values are combined and output based on the specific layers used and this output is used as input for the next layer, or held off from consideration until the last fully convolutional layer, typically the last layer of the network. Other layers of CNNs include pooling and connected layers (Ball et al., 2017).

The design of CNNs falls on a few basic traits, called hyperparameters; changes within these hyperparameters affect how the training is run (Song et al., 2019). Perhaps the most important hyperparameter is the epoch count. Epochs are a complete session of training where the algorithm views every image making its prediction, and receiving a verification, then adjusting based on its wrongness. Epochs can be increased for longer training sessions, though this will of course increase training time, and algorithms will not continually benefit from infinitely extended training sessions. To shorten the number of epochs needed, algorithms are commonly pretrained on data prior to use. Pretraining is the use of similar data to the target data to create roughly optimized networks that require only fine tuning on the target data, saving time (Z. Huang et al., 2017; Lévy & Jain, 2016).

The loss function is another critical component of the network architecture. Loss functions tell an algorithm after each epoch how far off it performed from the validation set. The optimizer function takes the loss values from the loss function, and decides how to best optimize the algorithm's functions and layers to perform better. Some optimizers require more work on the user end than others, and certain optimizers perform better for certain tasks. Two commonly used optimizers in image classification are SGD and Adam (Bera & Shrivastava, 2020). SGD requires more input on hyperparameters, while Adam uses self adaptive values that adjust through the training epochs (Kingma & Ba, 2014; Le et al., 2011).

2.5 Hydra Architecture

One of the most recently published papers expanding upon the fMoW dataset is the work of Minetto et al., creating their Hydra framework. Hydra is an ensemble of networks aimed at decreasing training time on the data while keeping results competitive with the competition results (Minetto et al., 2019). Hydra makes use of both ResNet and DenseNet architectures (Figure 3) for its functionality. Eight DenseNet algorithms and four ResNet algorithms are run simultaneously to explore the most efficient path towards a global minimum, and providing the highest F-Score. Using this multi-path method of searching allows for a more vast exploration of potential solutions, providing a higher likelihood of reaching the absolute global minimum. The Hydra ensemble uses a roughly optimized initial path, which is terminated early in the process, then allows the twelve algorithms to explore from this point to achieve the highest accuracy (Minetto et al., 2019). The Hydra network achieved accuracies that would have fallen in the top three scores during the challenge, however this was done in less training time due to their architecture and network structure, which consisted of ResNet and DenseNet algorithms using 11 training epochs, and an Adam optimizer.

3. Data

The algorithm was implemented on the fMoW dataset (for more in depth details, reference (Christie et al., 2018), but important details are summarized below). The formation of the dataset was to aid and encourage the advancement of image and scene classification algorithms, by allowing public use of the already classified and labeled training and validation imagery sets. The fMoW dataset provides sixty-two classes of remote sensing images (classes such as hospitals, car dealerships, airports,

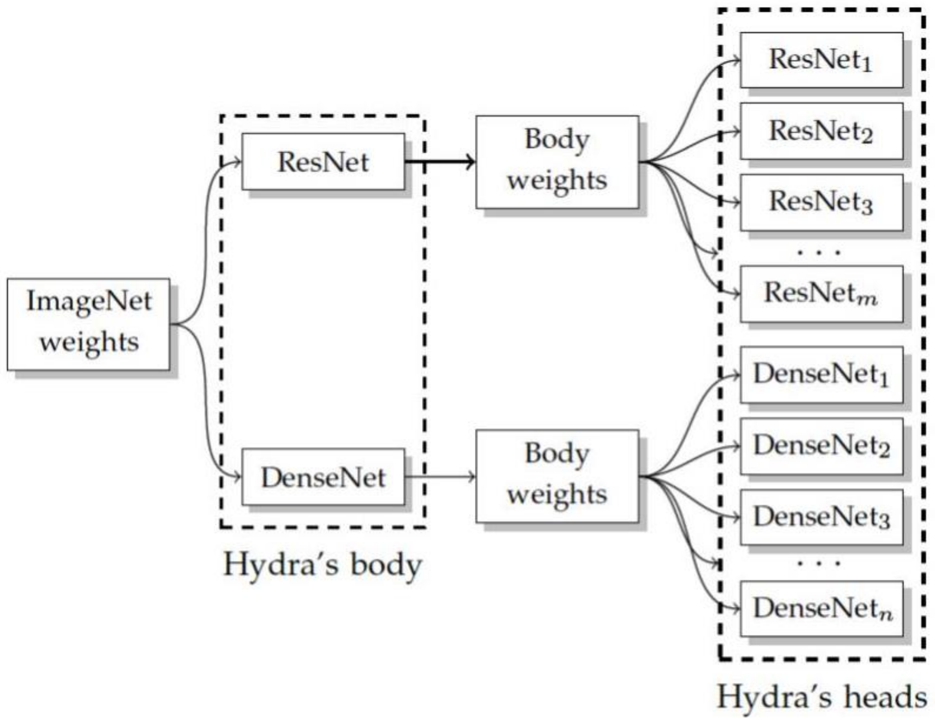


Figure 3. Hydra's network architecture is shown with its two major bodies, pretrained on ImageNet, which then receive the preliminary training body weights before being split into the twelve heads that make up the hydra ensemble. Figure and caption paraphrased from (Minetto et al., 2019).

smokestacks, race tracks, shopping malls, etc.). Figure 1 shows examples of images found in the dataset. The original challenge also included a false detection class that contained none of the other sixty-two classes. In total, the dataset consists of 470,086 images, with 295,843 allocated to training samples, and the rest reserved for validation images. The data contains metadata (UTM Zone, timestamp, ground sample distance, sensor angle, and bounding box) from the images as well, which was not given in full to competitors during the challenge, but was released after the competition concluded (metadata not used for this research) (Christie et al., 2018). The data formats released included full sized TIFF RGB images (pan sharpened and multispectral) as well as a smaller, compressed dataset consisting of JPEG RGB images to reduce the overall size of the data (Minetto et al., 2019). Shown in Figure 4, within each class, there are sets of specific image sites, which include multiple images of the same object, providing a temporal aspect to the object as well as differing viewpoints; this being helpful in discerning between two classes when time is a major factor, such as whether a structure classifies as a smokestack or a tower, as the smokestack's temporal images may include smoke being produced whereas a tower would not (Christie et al. highlight being able to identify office buildings by their parking lot occupancy during business hours). The images are derived from all over the globe, spanning 59 UTM zones, and over 14 years from the earliest to most recently taken images (Christie et al., 2018). It should be noted that the fMoW dataset does not have equal distribution in the number of images per class, and that varying image counts between each class may affect class diversity. The Oak Ridge National Lab has the fMoW dataset downloaded to their database in its full extent (however the dataset does not include the 63rd class "no detection"), so the original sized TIFF images are used by the algorithm. Using the PyTorch library, which provides deep learning architectures and other supporting code (optimizers, loss functions, etc), algorithms can easily be instantiated with desired parameters. The library also allows users to create pre-trained or untrained models to build.

4. Methods

4.1 Identifying Smokestacks Using the ResNet Algorithm

The methods of this research follow two main training variables. The first variable consists of dataset creation. Different dataset scenarios were created for this research, both two-class and multi-class datasets. The datasets created are meant to test the viability of different classes of data being collected against the smokestack class, and to see how the data gathered and its organization may affect the accuracy of identifying

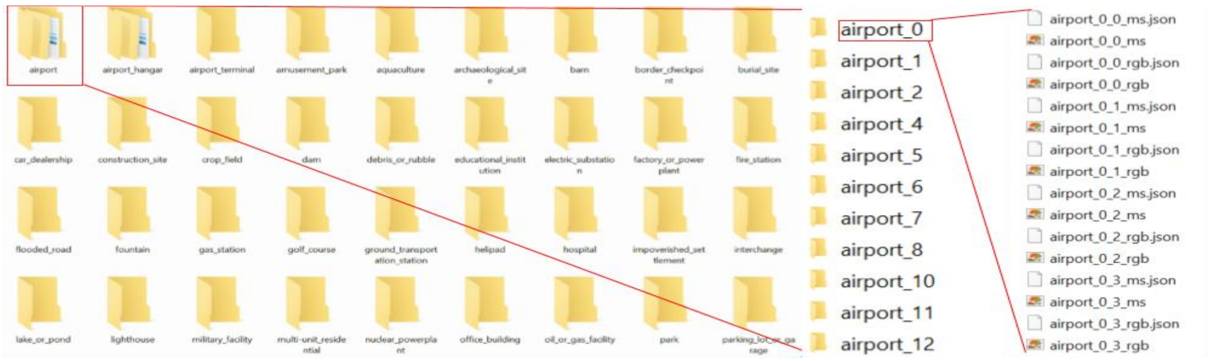


Figure 4. The fMoW structure is divided into the sixty-two major categories as seen to the left, which are further broken into specific instances, and each instance may have several images on a temporal scale, metadata files for each image, and different data types.

smokestacks. From the original 62 classes, the smokestacks classes are tested in two class scenarios, and multi class scenarios to view how the varying datasets tested influenced the smokestack's accuracy.

The second testing variable is the hyperparameters. Hyperparameters of the algorithm have direct influence on accuracy and on training time, and in order to explore and further understand the algorithm, four algorithms were used with different hyperparameters. The algorithm is run with varying amounts of epochs, two different optimizers, as well as without pretraining.

4.1.1 The Algorithm Selected

The algorithm used for the smokestack identification is Residual Net (ResNet), a popular CNN which was created by He et al. in response to the 2015 ILSVRC, but also due to the general need of more robust detection algorithms (He et al., 2016). ResNet upon its release was among the best classification algorithms as it won the ImageNet LSVRC-2015 contest with a leading error rate of approximately 6.7%. ResNet is commonly used for image classification algorithms (the problem faced in this thesis), but ResNet is also commonly used for specific object detection algorithms as well. This is partial motivation as well for the ResNet selection, as the algorithm can be expanded to object detection of the smokestacks, aiding in the automation of height retrieval.

The ResNet algorithm being utilized is pre-trained on ImageNet imagery, a concept widely used in machine learning to reduce training time as well as make better use of the data collected. Building the ResNet algorithm, initially a short training session of five epochs was used. This is done to allow the exploration of using more training epochs as a variable in the methods. Through expanding the epochs and changing optimizers, the algorithm is able to mimic the architecture used in the Hydra study. Both the original fMoW baseline algorithm and the Hydra algorithms use the Adam optimizer function. To compare the effectiveness of the optimizer, the ResNet algorithm for this thesis begins with the SGD optimizer.

4.1.2 Changing the Algorithm

The algorithm used in this study initially employs a 4 epoch training phase, with the SGD optimizer and cross entropy loss function. After this base algorithm was run for every dataset created, the optimization function was changed to an Adam optimizer (Kingma & Ba, 2014). Mimicking the architecture used in the Hydra network, it is possible to compare timing and accuracy to other commonly used hyperparameters. The design behind the Adam optimizer is that it adapts its own momentum parameter, which is linked to the learning rate (Kingma & Ba, 2014). By implementing the Adam optimizer there is one less hyperparameter to decide on. Implementing both optimizers across the datasets allows the comparison of whether or not Adam provides higher accuracies in every situation.

The algorithm is trained for 4 epochs throughout the majority of the algorithms. However to show the effects of longer training sessions on data, the highest accuracy dataset is run from 5 epochs to 15 epochs. As training sessions became longer, the deep learning algorithms began to give diminishing returns, and even started to stray away from the global minimum, giving less accurate results as it progresses into longer training sessions. Finally, the last algorithm ran was an untrained network (in order words, lacking the ImageNet pretrained network values) to reiterate the value of

ImageNet pretraining on algorithm training times and accuracies. Figure 5 gives an overview on methods used in this research.

4.2 Dataset variances

As the main part of the study, different datasets are used throughout testing in order to measure the effectiveness of adding more data, and more data classes, to test the best method for discerning your class of interest. Seen in Figures 6 and 7, the datasets prepared ranged from two classes (smokestack versus non-smokestack) up to thirty classes and sixty-two classes, with intermediate steps in between. For each dataset, separate training and validation folders were made to ensure the correct number of images would be processed by the album. The originally planned single step approach (consecutively training new algorithms at every number of classes between two and sixty-two) was decided against as time to create the dataset, train, and validate the algorithm would have consumed significant amounts of time, and the general trend in results is still seen by taking larger steps between class sizes. The datasets have been broken up into two categories: the first being two class datasets, and the second being multi-class datasets. For the two class datasets, images from different classes are bound into one class together, as a general non-smokestack class, and that the smokestacks are tested against this class. The multi-class datasets are meant to simulate a researcher creating their own imagery sets, and taking the time to properly sort their data into their respective classes, versus piling all data into two classes.

4.2.1 Two Class Dataset

Multiple two class datasets were created to test the viability of a simple

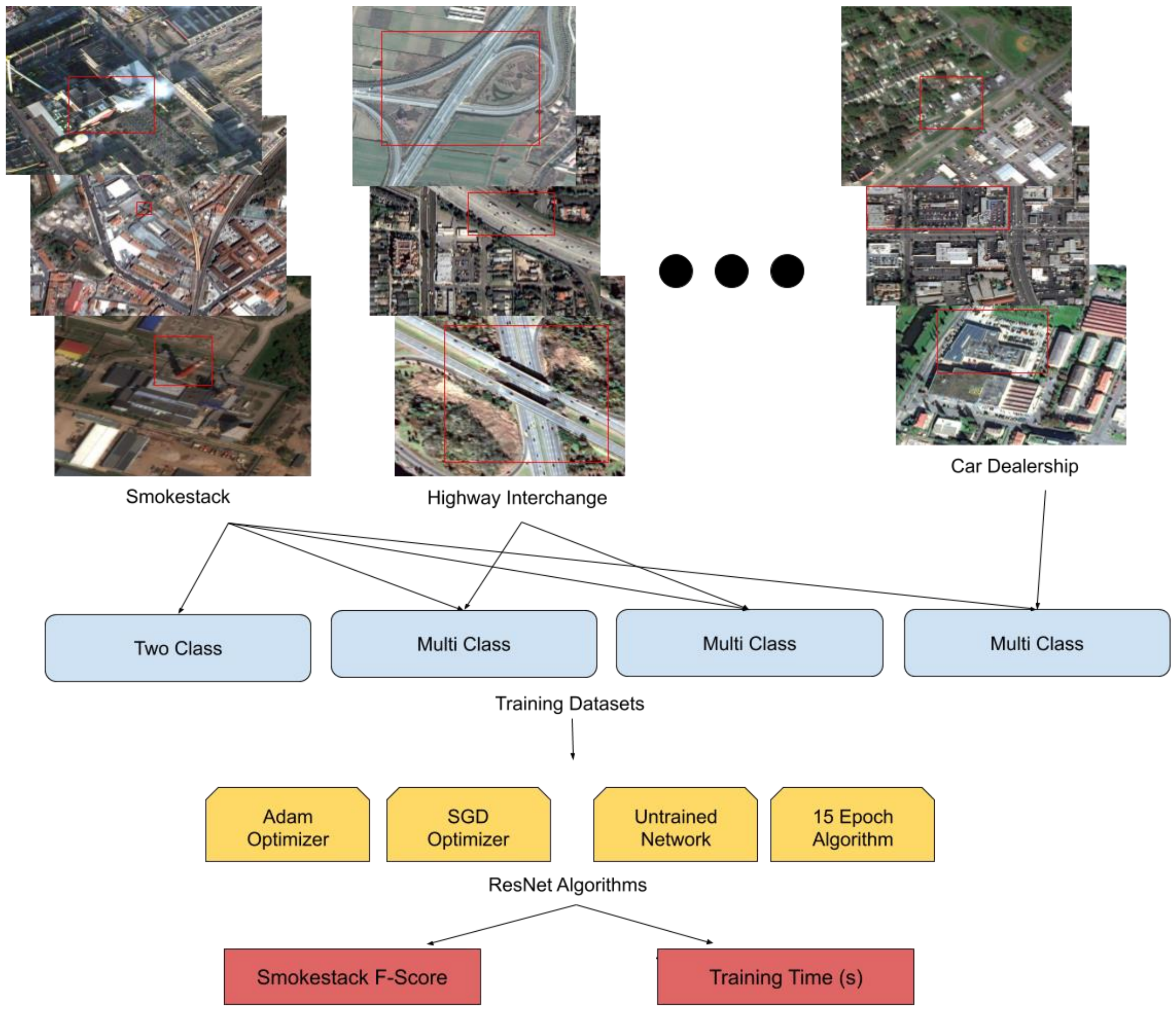


Figure 5. Flowchart showing the basis of the methodology. At the top level, training images are sorted into their classes (smokestack, highway interchange, car dealership, etc), and put into datasets. Note smokestacks are the recurring class in each dataset, and there are also no other classes explicitly set to recur. The training sets are passed through the different ResNet algorithms. The results recorded are the smokestack f-score, the confusion matrix, and the training time in minutes for every algorithm run.

smokestack versus non-smokestack model in building datasets. In every two class dataset, the smokestack class remains unchanged, consisting of every available smokestack image in the dataset, totaling 3429 training images and 451 validation images. However the makeup of the negative class is altered between each dataset. The first case created saw the negative class consisting of just one other class one to test the effectiveness of learning the difference between two classes and then testing on all the classes (or in real world terms, testing the viability of training against one class then applying to general data to sort out smokestacks). For the second case, the negative class consists of three other classes. This simulates a researcher selecting images of a few different types of objects to form a defined negative class. For this class, the selection of three classes was shuffled and the average smokestack f-score taken. In the last case, the negative class created consisted of 5% of the entire dataset (resulting in 14620 images) and this was to simulate finding all types of images and culminating them into one inclusive negative class. Figure 6 shows the makeup of the negative classes for the two class datasets. All renditions of the two class scenarios were tested against a validation set consisting of smokestacks versus a negative class containing images from every class.

4.2.2 Multi-Class datasets

The next classes set up were multi-class datasets. Where the two-class dataset consisted of smokestacks versus non-smokestacks, the mutli-class datasets force the algorithm to discern between each class (for instance smokestack versus hospital, highway, office building, etc). The aim of these classes is to sharpen the algorithm's

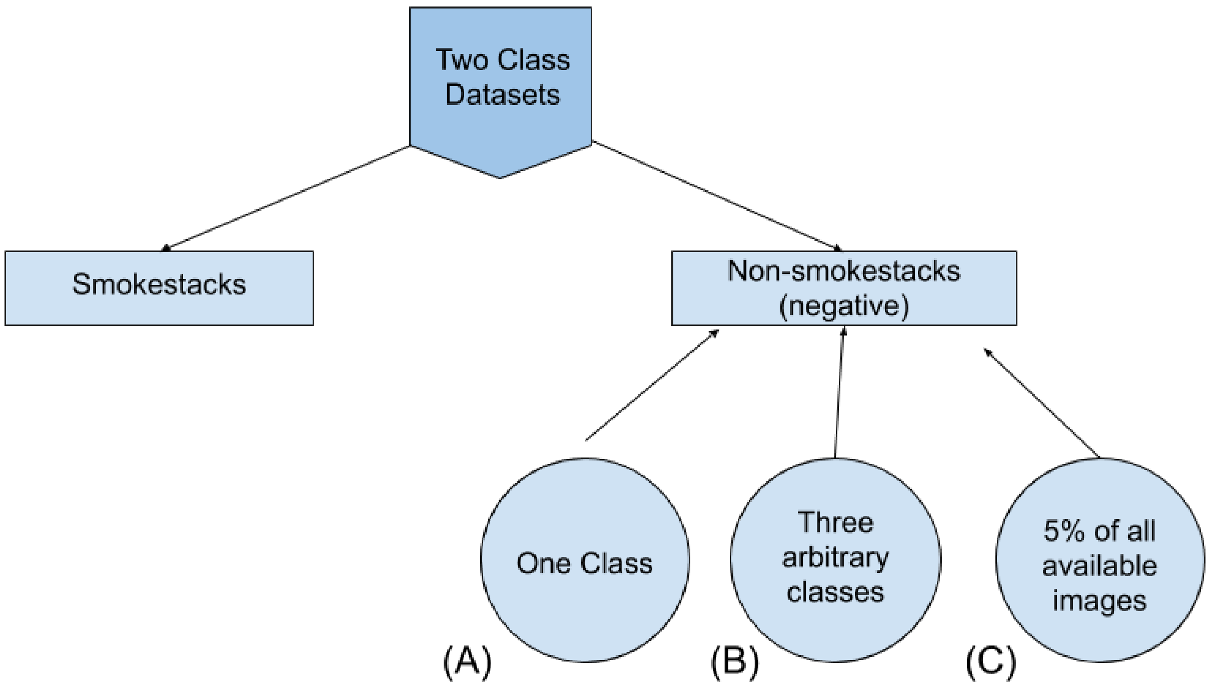


Figure 6. The two class datasets are composed from three different negative class compositions. Dataset A's negative class consists of one other class, chosen at random from the fMoW dataset. Dataset B's negative class consists of three classes chosen at random. Dataset C uses 5% of the images from every class in the dataset, forming a much larger and more comprehensive dataset.

definition of what isn't a smokestack, versus the negative class being a catch-all for borderline probability predictions (the algorithm thresholds predictions at 50%). It is reasonable to hypothesize that having numerous classes would shift a borderline (but incorrect) non-smokestack prediction back towards a more confident smokestack prediction as the other classes become more defined in their make-up.

Eight multi-class datasets were created: five, six, seven, ten, fifteen, twenty, thirty, and sixty-two class datasets (Figure 7). As mentioned, intermediate steps between the larger datasets were deemed sufficient as run times began to increase significantly and the general trend in time and accuracy remained similar between jumps.

4.3 Accuracy Assessment

The algorithms' success is being automatically measured by a function of the PyTorch library, which measures accuracy by how many correct predictions there were among the total number of images. While this number gives insight to how many correct predictions there were, the number does not tell a full story of an algorithm's performance. A largely common issue in deep learning algorithms is overfitting data, and simply telling the accuracy in this form does not explain the algorithm's degree of overfitting. Therefore, the metric of F-score is used to report the accuracy of the algorithm for this study. F-score is a popular accuracy assessment tool used for deep learning. While this number does not directly answer the question of how many images were guessed correctly, it does tell the bigger story on the algorithm's false positives,

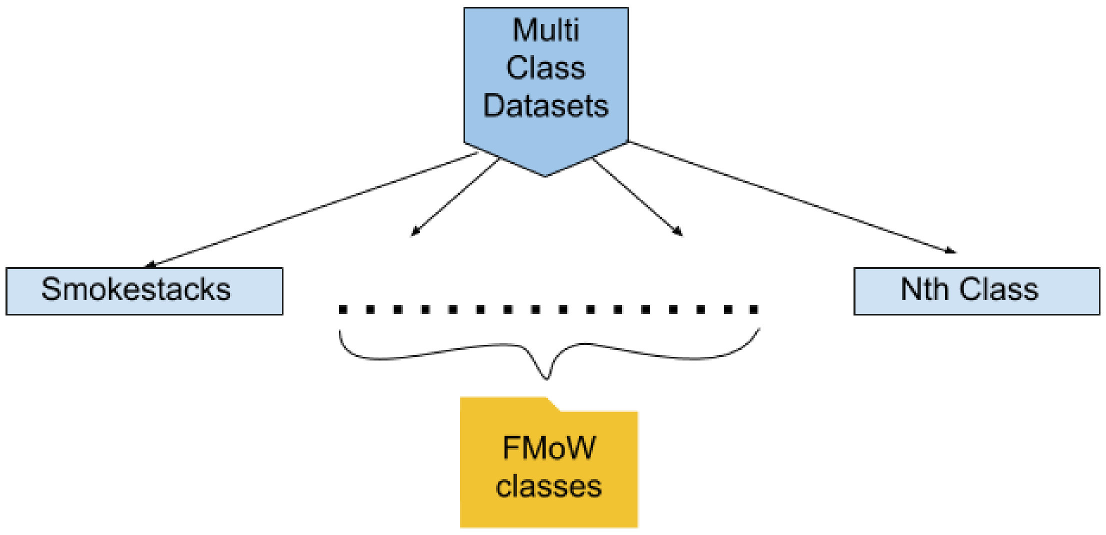


Figure 7. The multiclass datasets consist of individual negative classes, as opposed to a singular negative class as in the two class dataset. The algorithm is tasked with identifying features of each class separately.

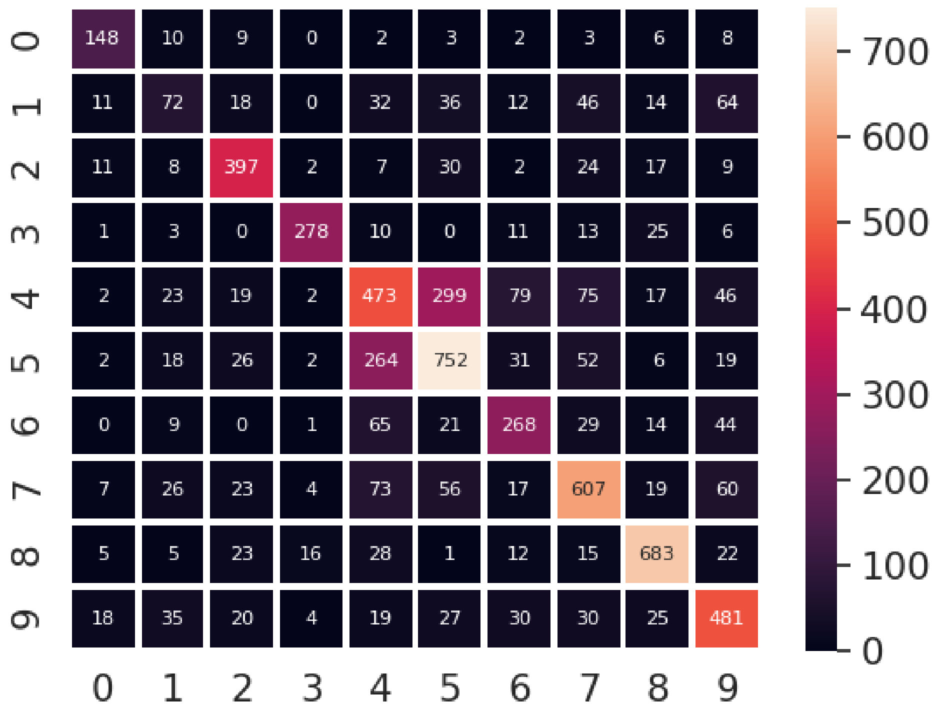
false negatives, and true positives as a whole. The F-score is calculated from the confusion matrix of the validation set; the confusion matrix is a graph showing ground truth labels plotted against predicted labels on opposing axes, showing the number of correct and incorrect predictions by the algorithm, as well as which classes the incorrect predictions were labeled as. A confusion matrix from this research is shown in figure 8.

Using two lesser metrics, precision and recall, the F-score offers a more complete narrative about a particular class. The F-score can also be applied to the entire dataset by averaging the individual class scores and their weights. The equation to calculate F-score is shown in Equation 1. The F-score is not perfect, as varying amounts of false positives and false negatives can produce similar F-scores, but the number is far more telling than a general accuracy percentage. After each training scenario, an evaluation dataset was run and each class' F-score was measured and reported. A CSV file is also exported, containing the individual file paths and the algorithm’s prediction label versus the true label, as well as prediction confidence values. The accuracies reported through the results section refer to the F-score, and not the algorithm generated score.

5. Results

5.1 Two Class Datasets

The two class datasets return polarizing results. Reviewing the results from scenario A (Figure 9) shows examples of extreme overfitting. The algorithm returned a



	0	1	2	3	4	5	6	7	8	9
0	148	10	9	0	2	3	2	3	6	8
1	11	72	18	0	32	36	12	46	14	64
2	11	8	397	2	7	30	2	24	17	9
3	1	3	0	278	10	0	11	13	25	6
4	2	23	19	2	473	299	79	75	17	46
5	2	18	26	2	264	752	31	52	6	19
6	0	9	0	1	65	21	268	29	14	44
7	7	26	23	4	73	56	17	607	19	60
8	5	5	23	16	28	1	12	15	683	22
9	18	35	20	4	19	27	30	30	25	481

Figure 8. Confusion matrix of the ten class dataset. The main diagonal shows true positive predictions, while the columns show the false positive predictions for a specific class, while the rows show the false negative predictions for a specific class. High values that occur off the main diagonal show classes that are more difficult for the algorithm to distinguish. Class names of the indices: 0, aquaculture; 1, burial site; 2, debris/rubble; 3, interchange; 4, multi-unit residential; 5, single-unit residential; 6, smokestack; 7, swimming pool; 8, tunnel opening; 9, water treatment facility

$$\text{True Positive}_a\text{:} \quad CM_{a,a}$$

$$\text{False Positive}_a\text{:} \quad \sum_{b=1,\, b\neq a} CM_{b,a}$$

$$\text{Precision}_a\text{:} \quad \text{True positive}_a \times (\text{True Positive}_a + \text{False Positive}_a)^{-1}$$

$$\text{Recall}_a\text{:} \quad \text{True positive}_a \times (\text{True Positive}_a + \text{False Negative}_a)^{-1}$$

$$\text{False Negative}_a\text{:} \quad \sum_{b=1,\, b\neq a} CM_{a,b}$$

$$\text{F-Score}_a\text{: } (2 \times \text{Precision}_a \times \text{Recall}_a) \times (\text{Precision}_a + \text{Recall}_a)^{-1}$$

Equation 1. The formula for calculating the F-Score metric is shown, as well as the needed parameters for the computation, precision and recall.

low average F-score of 0.33 over ten runs (Table 1 displays the ten runs and their smokestack F-scores). The F-scores range from 0.01 to 0.61, with the majority being in the 0.4 to 0.5 range. The confusion matrices show the algorithm is largely overfitting the data; generally 80% of the smokestacks are correctly predicted, while large numbers of the non-smokestack class were also misidentified as smokestacks. The output is also indicative of underperforming when underfitting the smokestacks predictions, giving also no smokestack predictions (some runs saw as few as eight correct smokestack predictions, leaving 433 that were predicted as non-smokestack).

The second scenario two-class dataset did return better results, giving a F-score of 0.54 for the smokestacks class. While better, still examples of extreme over/underfitting are seen. The third two-class scenario performed the best, returning a smokestacks F-score of .62 using a total of 18,050 images. This score is competitive with the higher scoring multi-class datasets. It displayed a much lower number of false positives, however there are a greater number of false negatives. Confusion matrices for these datasets are also shown in Figures 10 and 11.

Table 1. F-score values return from the two class dataset (scenario A).

	Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Run 6**	**Run 7**	**Run 8**	**Run 9**	**Run 10**	**AVG**
F-Score	0.008	0.54	0.43	0.10	0.40	0.49	0.05	0.61	0.13	0.50	0.36

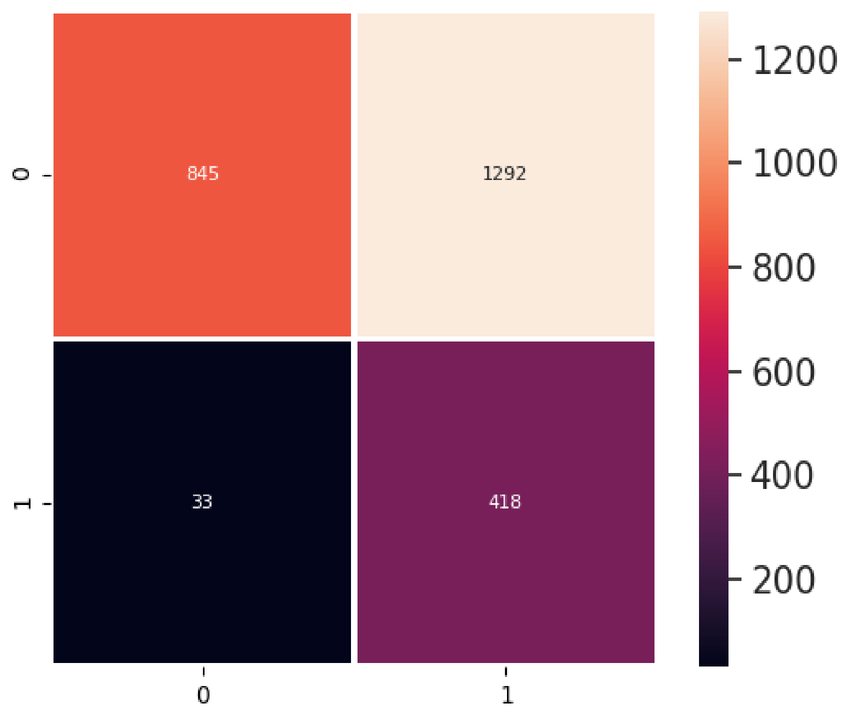


Figure 9. Confusion matrix representative of the two-class datasets (scenario A). The classes relative to the indices are 0: non-smokestacks, 1: smokestacks.

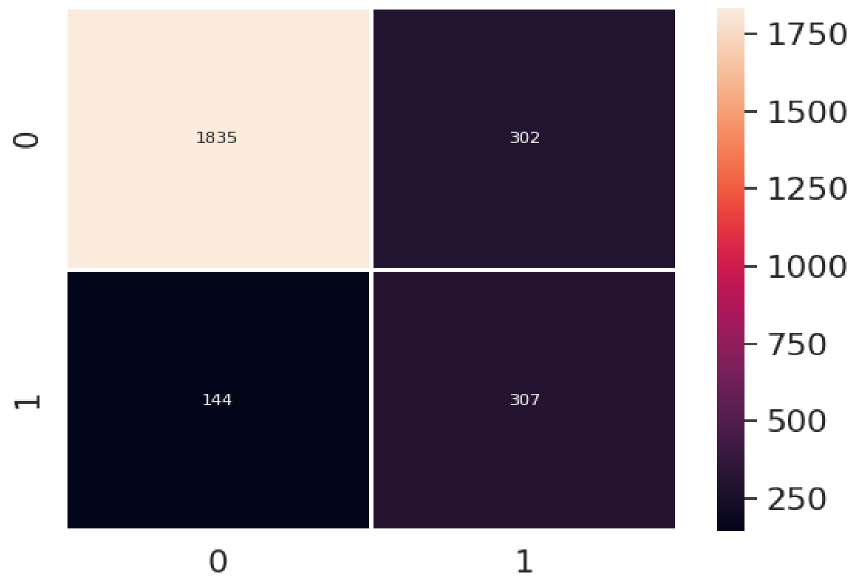


Figure 10. Confusion matrix representative of the two-class datasets (scenario B). The classes relative to the indices are 0: non-smokestacks, 1: smokestacks. Note the decrease in false positives in cell [0, 1].

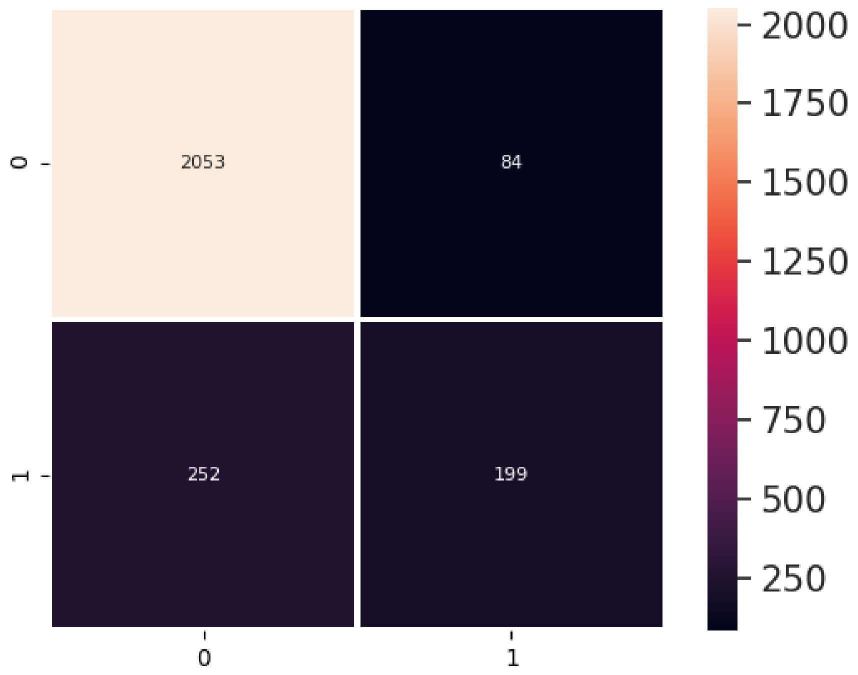


Figure 11. Confusion matrix representative of the two-class datasets (scenario C). The classes relative to the indices are 0: non-smokestacks, 1: smokestacks. Note the much lower number of false positives in cell [0, 1].

5.2 Multi-Class Datasets

The results of running the algorithm over different sized multi-class datasets show peaks and troughs in accuracy of smokestacks as output sizes increases, while training times show significant increase as the number of images increase. The training times as a response to the increasing number of classes (and therefore growing image count) are shown in Figure 12. Through the thirty class dataset, the algorithm training times follow an exponential growth pattern, as opposed to a linear trend that may be expected. The exponential trend is broken when the jump to sixty-two classes is made; the algorithm becomes slightly more efficient, but still far from linear.

The multiclass datasets, using both SGD and Alex optimizers display an upward trend in smokestack accuracy as the number of classes increases from three to five classes. After the peak in smokestack F-score value in the five class dataset, the accuracy begins to suffer a loss as classes increase to thirty classes. After the thirty class dataset, a sixty-two class dataset was run in order to compare to the hydra results, and the smokestacks individual F-score increased by 22 percent.

5.3 Stochastic Gradient Descent vs Adam Optimizer

In switching from the SGD optimization to the Adam optimizer (used in the Hydra network) a jump in smokestack F-score is seen, as well as slightly longer training times in most cases. The Adam optimizer F-scores are generally .6% higher, with some class sets giving differing or even lower results. The Adam scores follow the same trend of accuracies through classes as the SGD optimized algorithms, as seen in Figure 13. The training times jump anywhere from 8% to 27% increases in training time using the Adam optimizer, Figure 14 shows the differences in training times vs training image count.

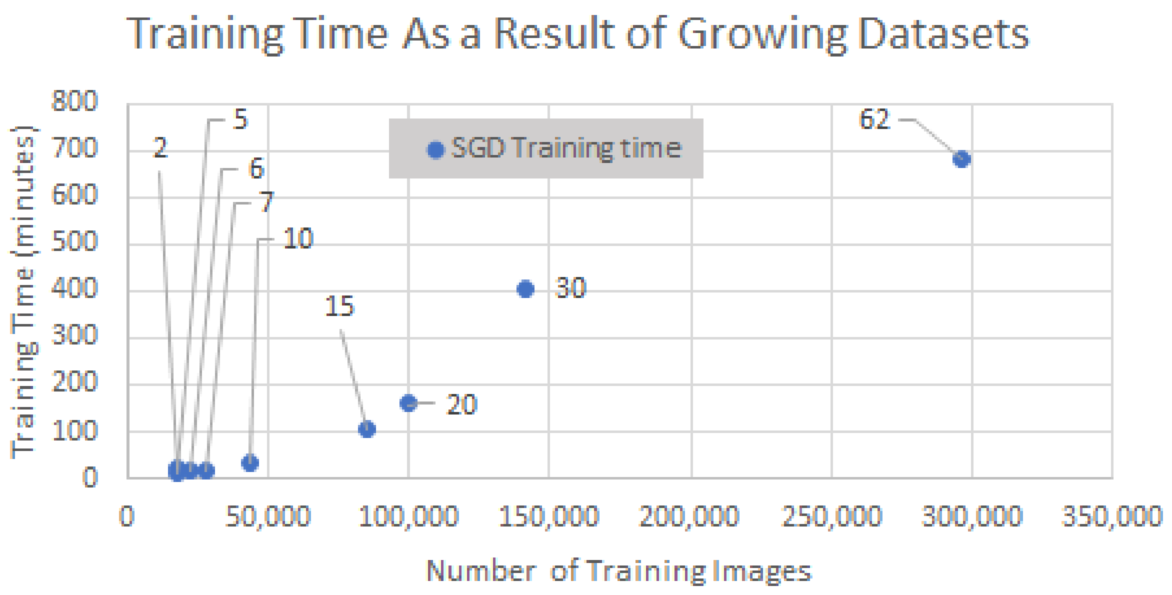


Figure 12. The base algorithm’s training time response to an increasing dataset.

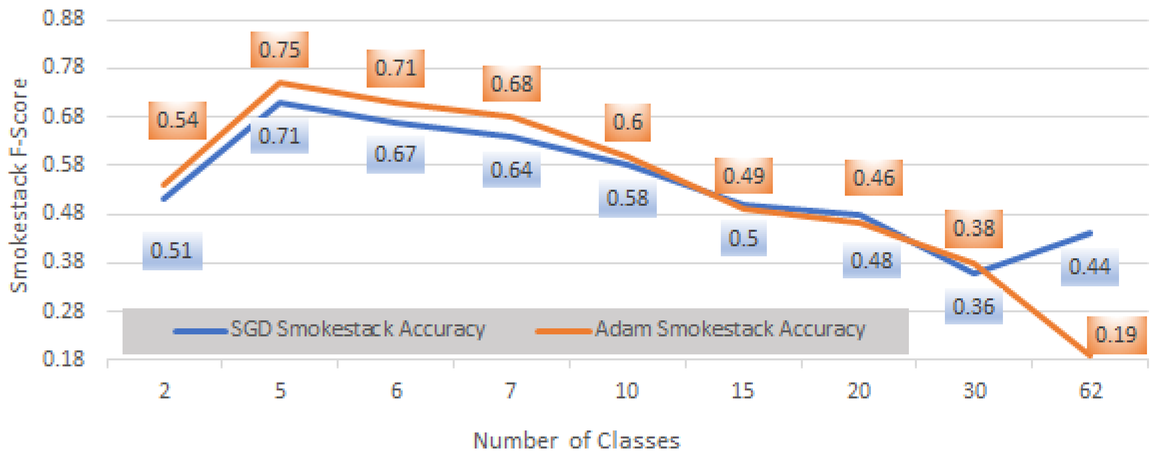


Figure 13. Graph displaying the smokestack F-score as the number of output classes increases using both SGD and Adam optimizers.

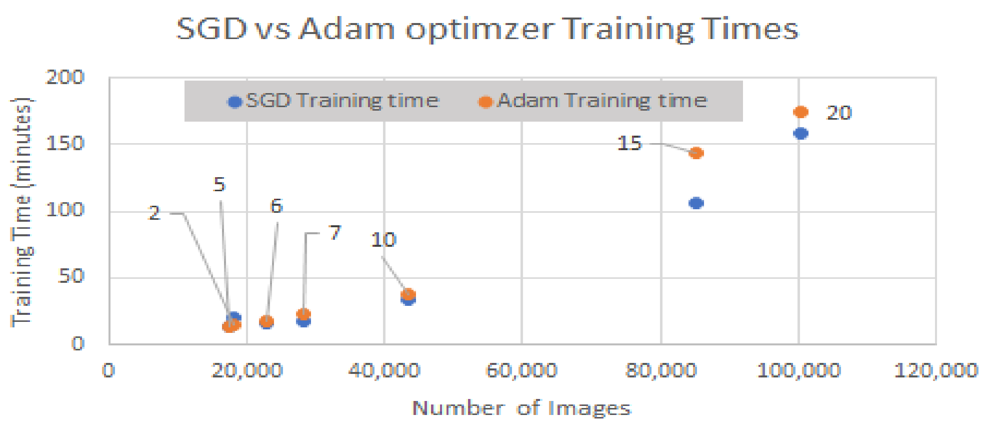


Figure 14. Graph displaying time differences between the SGD and Adam optimizers.

5.4 Longer Epochs

After running the algorithm with the different optimizers, the base algorithm (4 epochs, SGD, 5 classes) was run with longer training sessions. The training sessions remained consistent in the results returned, having no peaks or troughs in accuracy through the longer epochs. The average accuracy returned was 0.725. Small fluctuations from the average are seen but none that would suggest a significant difference from the average. Figure 15 shows the fluctuations in accuracies from five to fifteen epochs, using the same training and validation sets each time, as well as the same ResNet50 algorithm architecture. Several longer training sessions were run on bigger datasets to test the idea that more data required more training but the results did not differ from the ones seen in Figure 15.

5.5 Non pretrained Networks

The network run without pretraining did not fare well in accuracy. The network was run with five classes over 20 epochs. One class did not receive any predictions (the

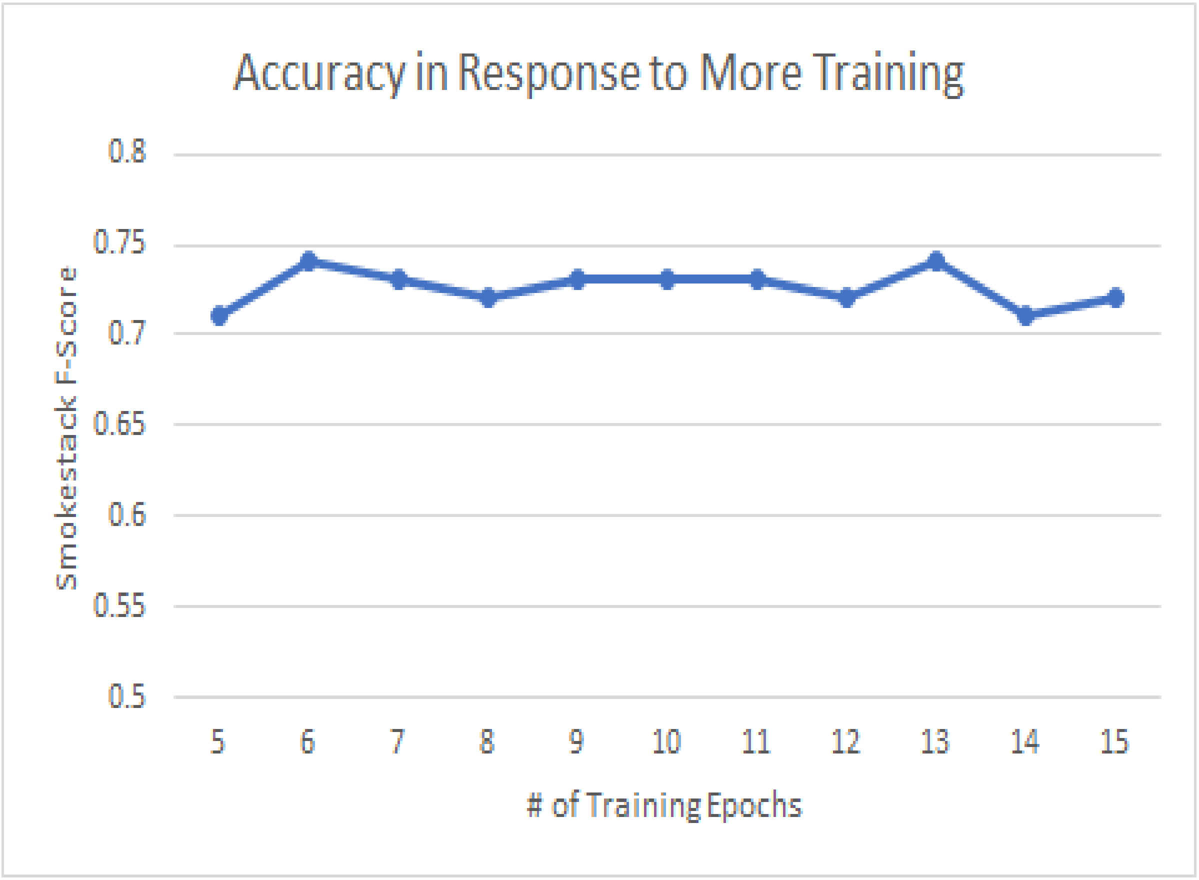


Figure 15. Graph showing the deviations in accuracy as training epochs are increased.

algorithm did not assign any predictions of class index zero), and the F-score achieved for smokestacks was .03 with 443 false predictions of smokestacks being mislabeled. A total of 8 smokestacks were predicted correctly, and only 48% of all validation images were correctly predicted. Figure 16 shows the confusion matrix for the network's validation set. The effects of pre-training are well seen through this data.

5.6 Example Images

Viewing images and their predictions by the algorithm provides insight on which types of images are being correctly labeled and which images prove to be difficult for the algorithm to distinguish. Beginning with the images that were guessed correctly across all algorithms run (that is, every training input scenario provided an algorithm which correctly identified the image), it is clear that the algorithm clearly guesses smokestacks which exhibit a few key characteristics. Figure 17 (a-d) shows correctly labeled smokestacks that have prominent shadows extending from their base. In reviewing the images, most images with prominent shadows were correctly identified by the algorithms. Another characteristic that was often picked up on by the algorithms is the presence of patterns on the smokestacks. Some smokestacks contained no pattern, and are a solid color, while others contain a striped pattern.

The striped smokestacks were typically labeled correctly by the algorithm, presumably because such patterns aren't seen in other images. The stripes are prominently displayed in Figures 17c and 17d. Another characteristic displayed by Figure 17 are off-nadir images. Most images that were taken directly on-nadir do not allow any of the major characteristics of the smokestacks to be displayed (namely, their height and profile above the surrounding environment, providing the profile seen in the

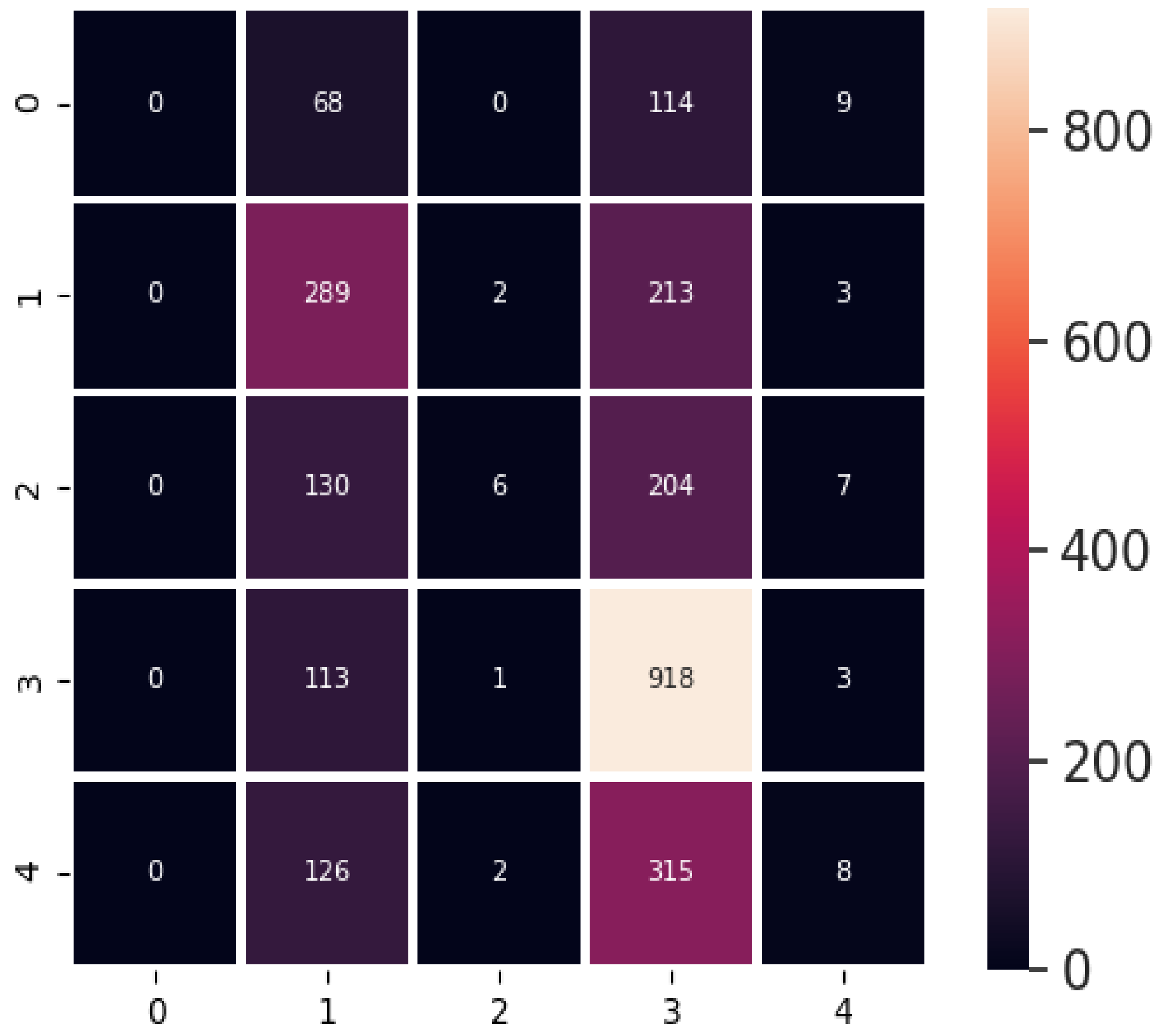


Figure 16. Confusion matrix of the non pre-trained network. While there are two classes with high values of high positives, there are also high numbers of false positives for both classes. The other classes also received little to no labels. The class labels for this matrix are as follows, 0: aquaculture, 1: debris/rubble, 2:interchange, 3: multi-unit residential, 4: smokestack.

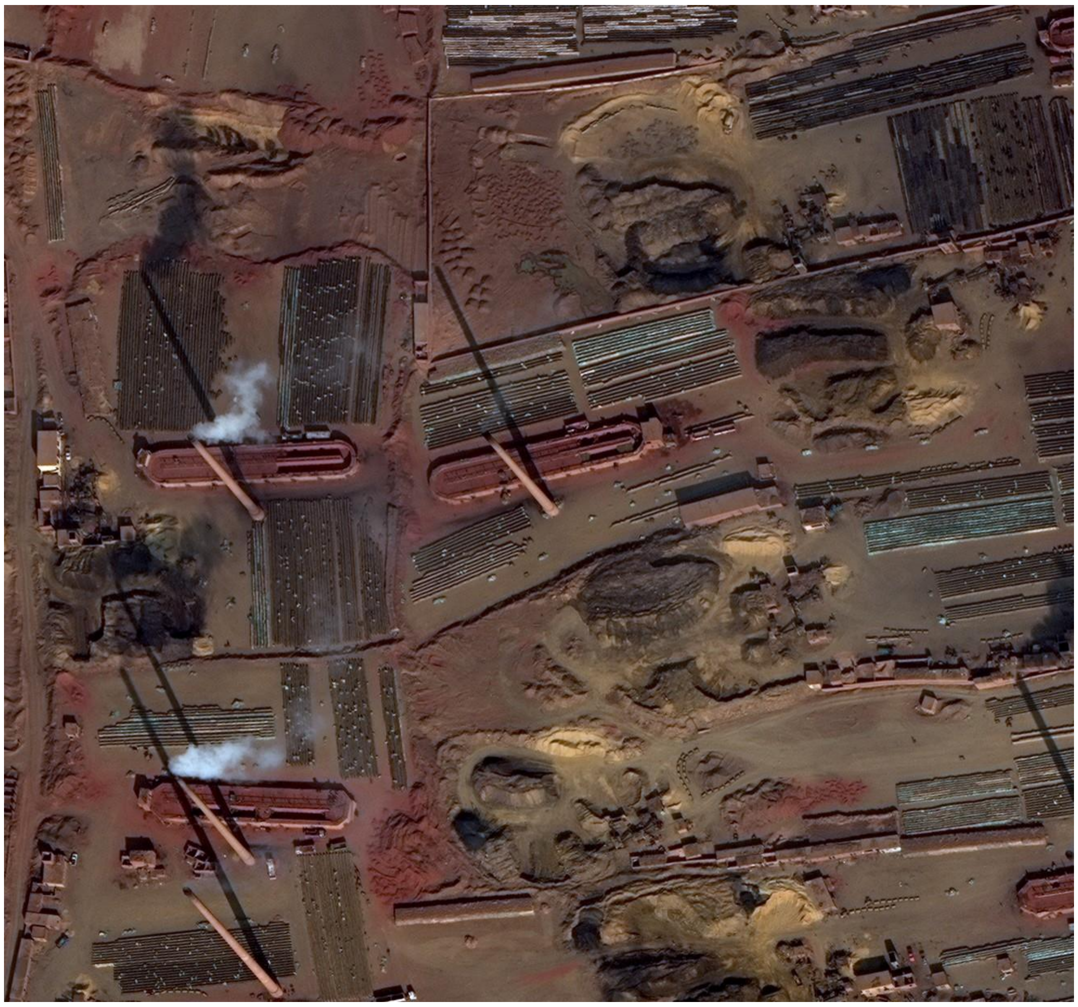


Figure 17a. Image of smokestack which was correctly labeled by all algorithms. Major characteristics are the tall smokestack profile and the strong shadows.

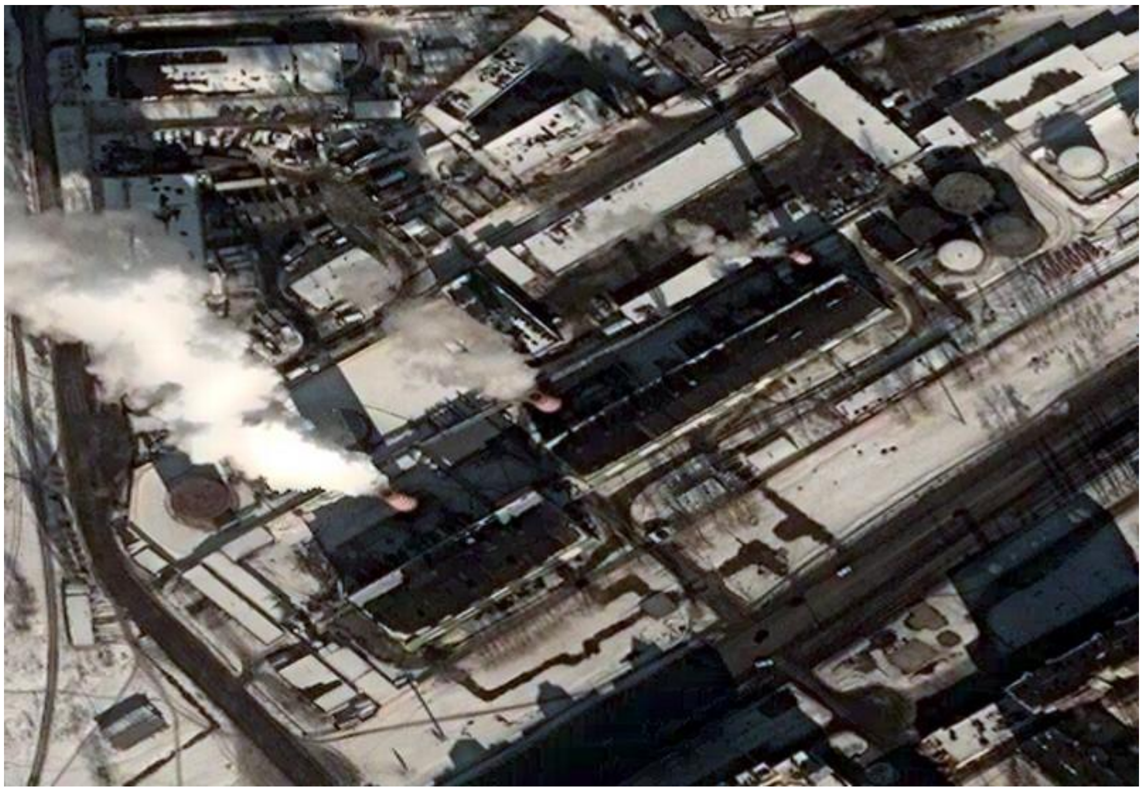


Figure 17b. Image of smokestack which was correctly labeled by all algorithms. The smoke from the smokestacks often leads to a correct prediction.

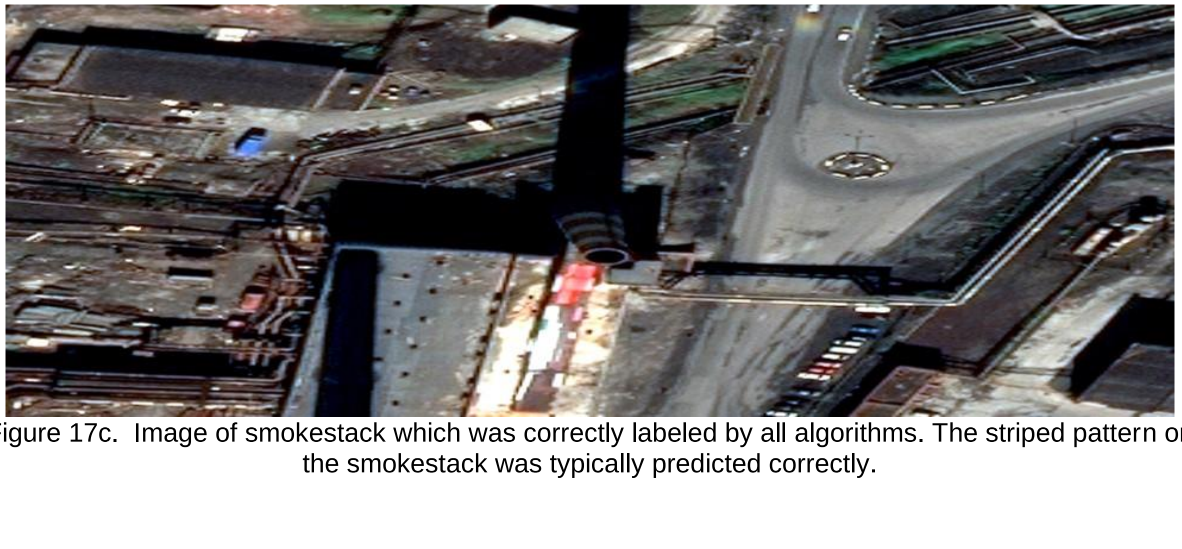


Figure 17c. Image of smokestack which was correctly labeled by all algorithms. The striped pattern on the smokestack was typically predicted correctly.

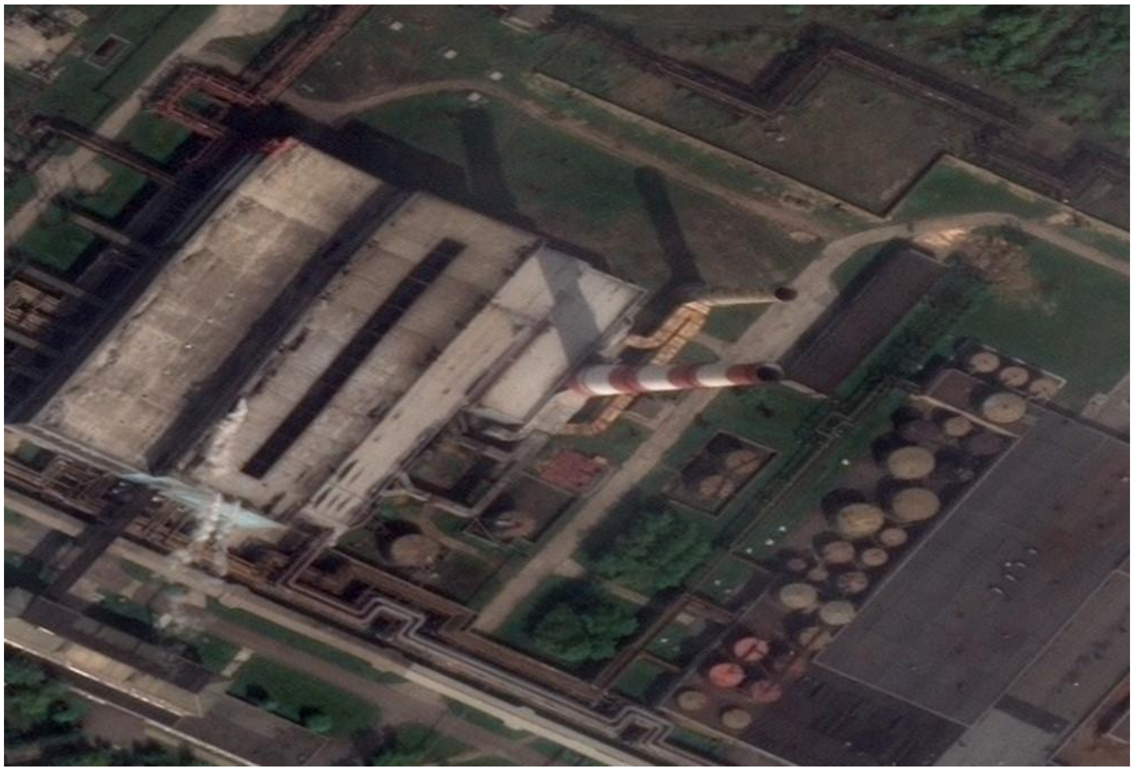


Figure 17d. Image of smokestack which was correctly labeled by all algorithms. The striped pattern on the smokestack was typically predicted correctly.

smokestacks in the aforementioned figures). On-nadir imagery must rely on only shadows of the smokestacks, as well as the presence of smoke being expended from the smokestack, which is likely to aid the algorithm in labeling as well, as none of the other features in the fMoW dataset have this characteristic.

Some images, the algorithm struggled greatly with, and none of the training scenarios could produce correct labels for the images. These images often were difficult even for a human observer to pick out the smokestacks; Figure 18 (a-e) depicts examples of images that were incorrectly labeled across all algorithms. Figure 18a does contain the prominent shadow, but the image displays lower contrast and a near on-nadir viewpoint of the smokestack. Figure 18b displays the prominent shadows as well, but due to the time of day (judged by surrounding environment shadow lengths), the high solar angle does not allow for the shadows to have longer lengths as seen in the other example images shown. The near on-nadir photo again prevents the tall smokestack profile from appearing. While Figures 18c and 18d both display off-nadir images, one image displays a much larger area, therefore the smokestack appears very small and not a prominent feature, and the other is shot from the shadowed side of the landscape, and therefore much of the photo is dark and lacks contrast. The smokestack shadow is difficult to make out for a human observer, and the structure itself is easily mistaken for a shadow or tree. As an example of low contrast being difficult for the algorithm to locate features, Figure 18e shows two examples of the same location, but in different weather conditions (the image appears to be blocked by light cloud coverage). While a human observer may not be misguided by this, the algorithm obviously struggles when contrast is low, as the high contrast image (top) received correct labels through all data scenarios, while the bottom received no correct labels.

6. Discussion

Employing the algorithm architecture used in the Hydra framework by Minetto et al. gives a comparison value of using the metadata from the competition during training, vs training only using images. The original fMoW baseline algorithms were also run on metadata only, to understand the effect the metadata has on algorithm success, and reported an F-score of 0.2 for the smokestacks, running a 62 class algorithm (Christie et al., 2018). Although run on a different deep learning model, this shows the importance of metadata, and can help to explain the departure of the accuracies reported here versus the Hydra papers, as the only difference in the architectures is the lack of metadata inclusion on the end of the network. Another facet of this research is looking within the Hydra paper, and seeing how input dataset manipulation affects the outcomes of one class of interest. Minetto et. al post their final accuracies for the entire 63 class dataset (0.781, including the false detection class) but do not go into detail of how the specific classes performed beyond a confusion matrix (Minetto et al., 2019). Replicating the architecture and altering the input datasets/output formats provides insight to how certain classes may respond to larger or smaller datasets. Minetto et al. also show how the Hydra ensemble responds to longer training sessions, confirming the results seen in this study, where there is an asymptotic leveling of the accuracy as the epochs increase.

The two class datasets are generally returning moderately low F-scores (~.5), which is largely due to overfitting. In outright accuracy, they are returning roughly 80%

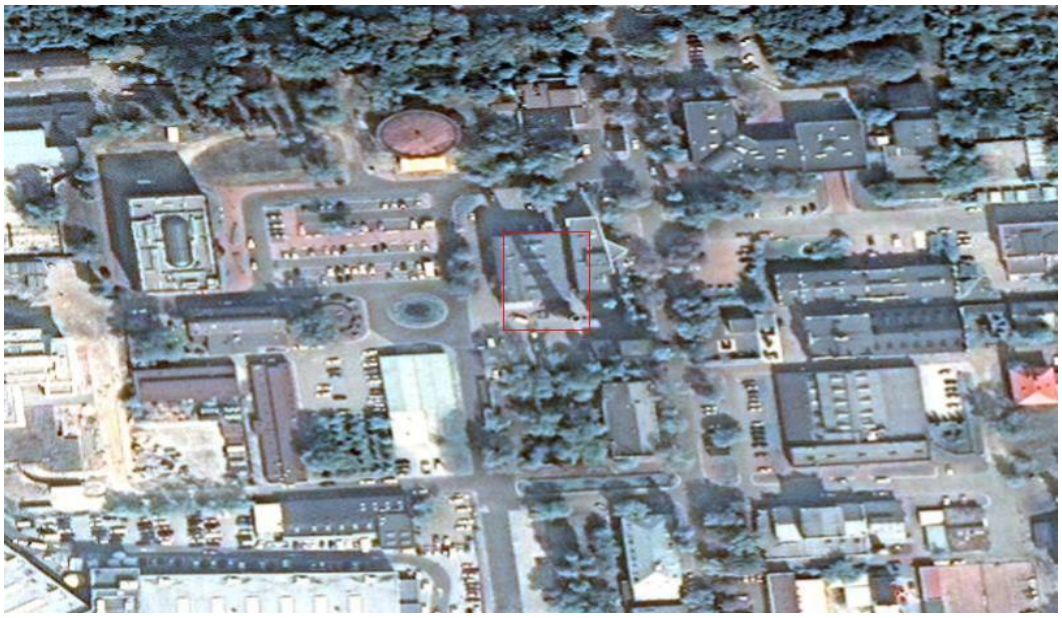


Figure 18a. Image of smokestack which was incorrectly labeled by all algorithms. The on-nadir imagery typically was not well predicted by the algorithm. Bounding box added.

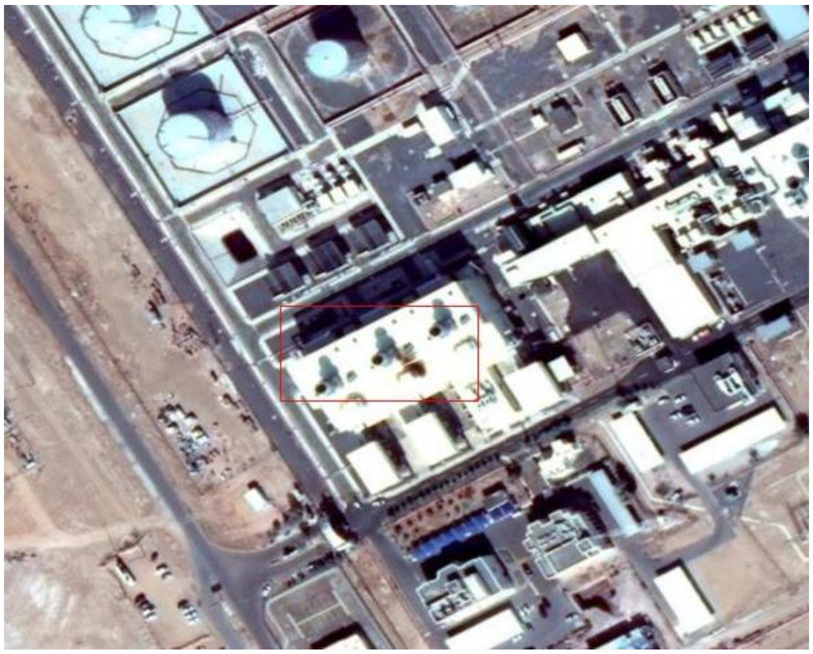


Figure 18b. Image of smokestack which was incorrectly labeled by all algorithms. The short shadows combined with near on-nadir imagery was incorrectly predicted. Bounding box added.

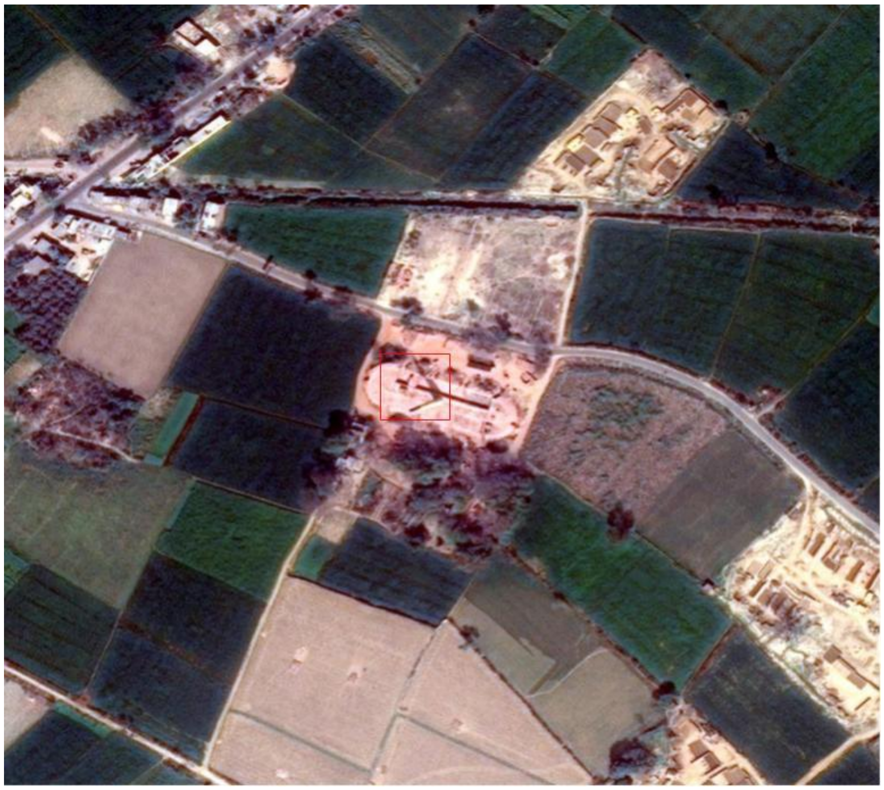


Figure 18c. Image of smokestack which was incorrectly labeled by all algorithms. The high amounts of surrounding environment and small smokestack profile/shadow did not receive correct predictions throughout the tests. Bounding box added.

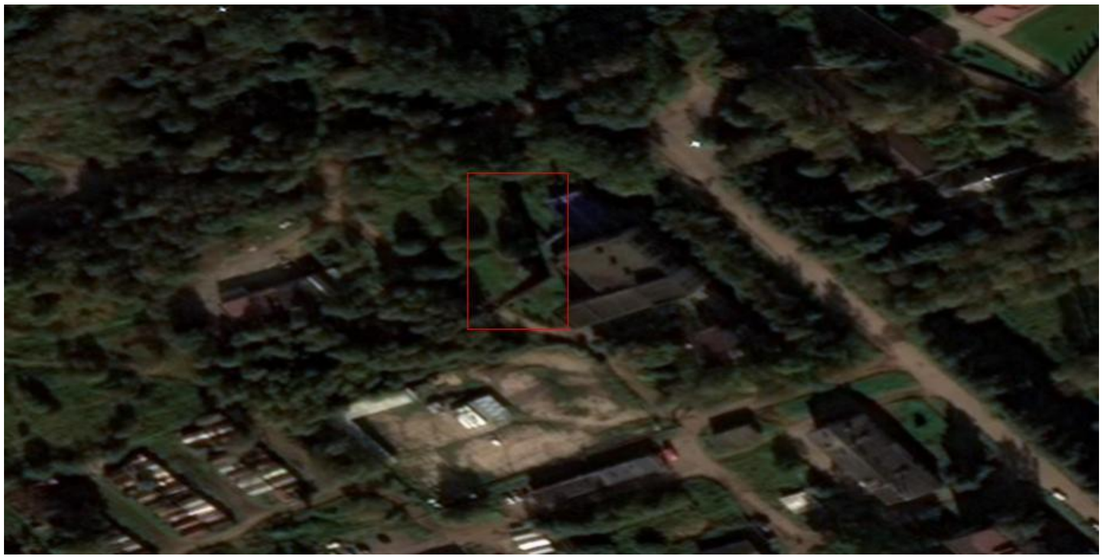


Figure 18d. Image of smokestack which was incorrectly labeled by all algorithms. The image is taken from the shadowed side, which makes the smokestack difficult to locate among other dark objects. Bounding box added.

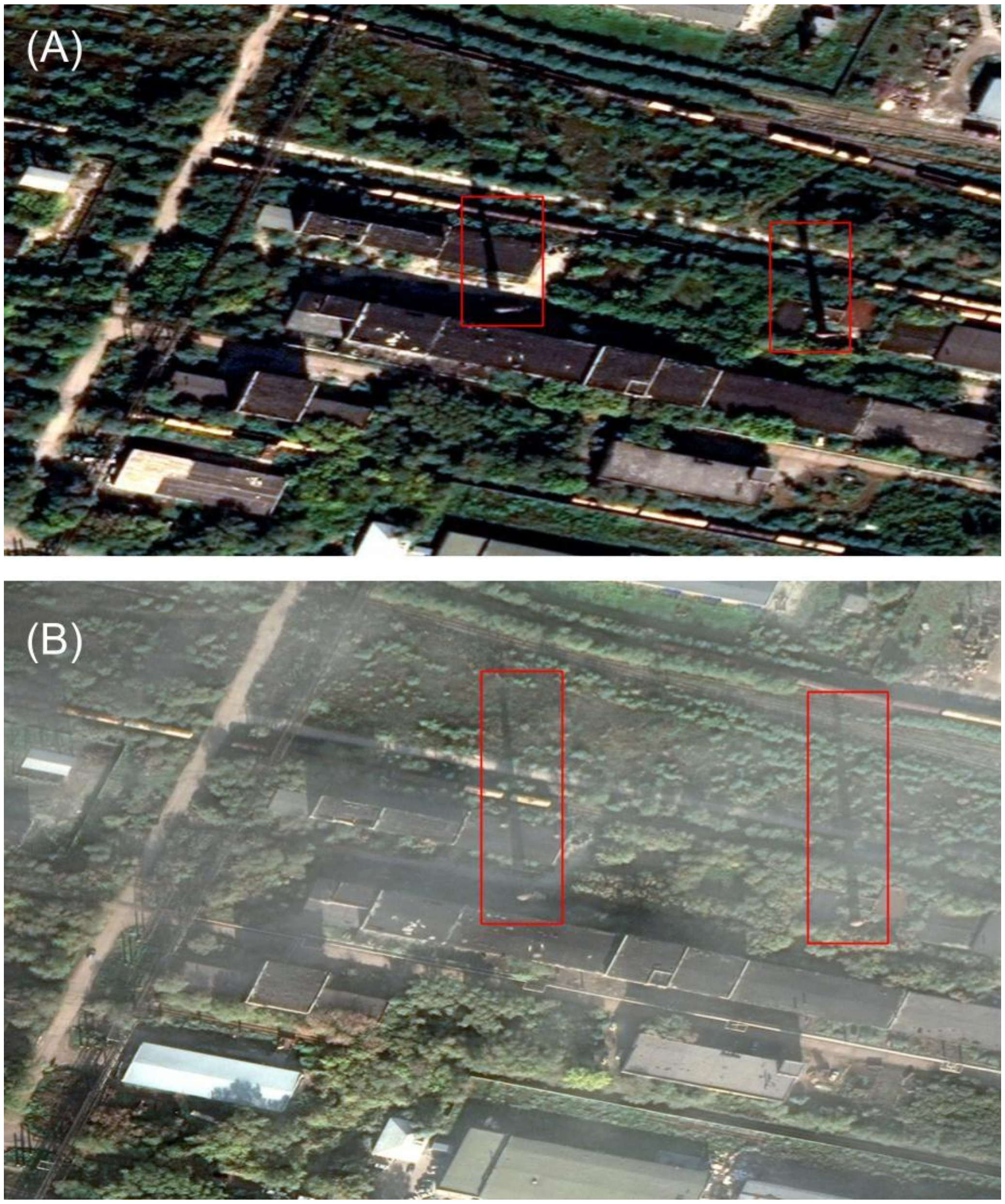


Figure 18e. Two smokestack images of the same location, but one photo is lower contrast, likely due to weather conditions (clouds) at the time of photo. Photo A received all correct labels, while photo B did not receive any correct labels. Bounding boxes added.

of correct smokestack predictions. Very small amounts of smokestacks are being mislabeled, while on the other side of the argument large numbers of non-smokestacks are being mislabeled as smokestacks; one run in particular saw 1118 (out of 1876) negative images predicted to be smokestacks. The runs that are not producing largely overfitted data are instead returning largely underfit smokestack data. Training an algorithm against one class of objects does not appear to be an effective strategy in detecting smokestacks in imagery. While the 3-class algorithm did perform better, the best use of data appeared to be a large two class dataset, consisting of a large negative class, with as many types of image classes as possible (simulated by adding 5% of the entire dataset). This was to be expected, however it was worth testing the viability of using smaller test groups; larger test groups with not much definition can sometimes return low accuracy due to confusion of the algorithm from overly complicated or vague outputs.

With adding more data and more specific outputs via the multiclass sets, the algorithm is able to properly guard against underfitting/overfitting on the data predictions. However at a certain point, it is seen that the outputs are so vast, the algorithm gets overwhelmed and the number of sparse FP/FNs begins to add up, lowering the F-score. Perhaps this is why the peak in the accuracy occurs at 5 classes. Much research has been done connecting the ‘overstimulation’ of an algorithm to its tendency to overfit data, and applying dropout layers to reduce the amount of inputs going into each training epoch. Utilizing dropout layers in their algorithms proved to reduce overfitting and error in the algorithm (Srivastava et al., 2014). Asking a deep learning algorithm to output too much or too little can lead to outputs that are difficult to discern against (giving high false positive/false negative values), or outputs that are too vague (usually returning extremely high false positive values) respectively. In structuring the output (binary output vs multiple outputs), the input data is also affected. Another issue that arises with input data is the quality of the data. Some images in the smokestack class (and presumably other classes) are difficult even for a human observer to identify the smokestack in the image. Images like these may only convoluted the algorithm’s understanding of smokestack image patterns.

Shown in the results here, the smokestacks class does not perform its best when run in the full dataset, but rather with a smaller dataset. In viewing the incorrectly labeled images from the smokestack class, it also stands to reason that the training images fed to the algorithm could be better selected, both in the smokestacks class and other classes. Poor image quality and general bad selection lead to a decrease in algorithm performance, as even a human observer may struggle to find the objects of interest in certain images.

The Adam optimizer appears to offer slightly higher accuracy at the penalty of slightly longer training times. The longer training times are likely attributed to the Adam optimizer adapting its own momentum values based on the previous epoch, a function within the optimizer which takes extra time to run. But these self-adapted and fluid momentum values allow the Adam optimizer to return higher accuracy values. In deep learning algorithms it would appear that self regulating, and therefore fluid, hyperparameters return the best values, as the training process is not a static process, suggesting that the hyperparameters should not be static either. This nature is somewhat seen in the Hydra research. The Hydra network uses on-line augmentation, creating a newly augmented dataset for each epoch (Minetto et al., 2019). The use of online augmentation, as opposed to augmenting data before the training, begins allows for a dynamic training set, which gives the algorithm a more complete understanding of what it is trying to learn through the data.

Using more epochs in the training session did not create a more accurate algorithm, negating the need for longer more intense training for the problem addressed in this thesis. Potential reasons for this are the algorithm being pre-trained on ImageNet and already being near max efficiency of image recognition. The other train of thought would be that the algorithm has been pre-trained too far on the ImageNet data, and therefore cannot change the weights within the body of the network enough to further maximize its efficiency (see figure 19, displaying the body vs the head argument). Pre-training and the concept of transfer learning is used not only in remote sensing, but in other instances of limited data (in the medical field, (Lévy & Jain, 2016), general object recognition, (Schmitz et al., 2014)). It has been studied that algorithm performance can not only be increased by modifying the training process, but also by modifying the pre-training process (Guo et al., 2016). This, along with additional dropout layers, is something that should be considered to increase the algorithm accuracy, should this research be continued.

There were several limitations to the study that should be addressed. While the results showed that the five class dataset returned the highest value, this result may only be applicable to this dataset, i.e. to state that a 5 class dataset would return the highest smokestack F-score for every dataset would be an improper conclusion. The results here are highly dependent on the distribution of images between the classes,

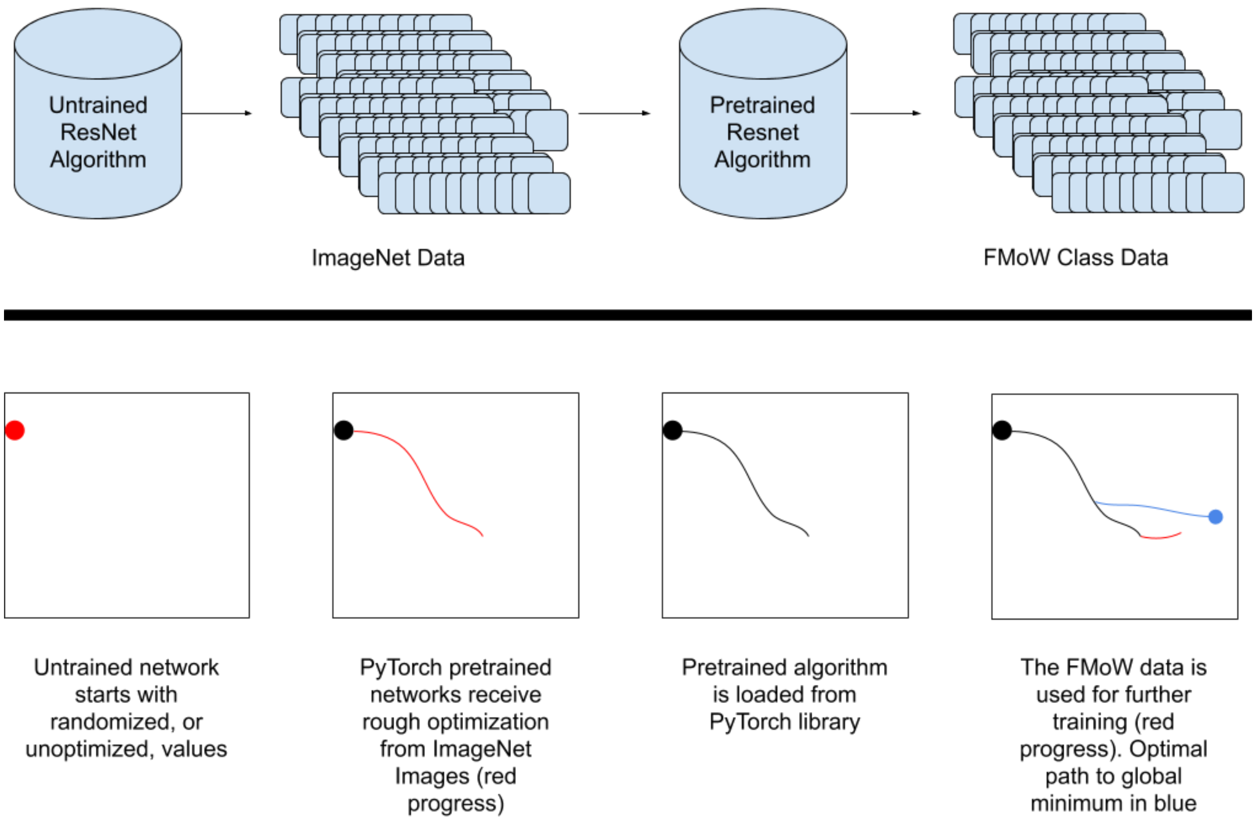

Untrained ResNet Algorithm
ImageNet Data
Pretrained Resnet Algorithm
FMoW Class Data
Untrained network starts with randomized, or unoptimized, values
PyTorch pretrained networks receive rough optimization from ImageNet Images (red progress)
Pretrained algorithm is loaded from PyTorch library
The FMoW data is used for further training (red progress). Optimal path to global minimum in blue

Figure 19. Potential effects of pre-training. As the randomized network progresses through pre-training, it is optimized towards the global minimum of the pre-training set. It is conceivable that the network is trained away from the optimal path to the global minimum of the training set (shown in blue). This graph is for conceptualization only; the axes have no value, the paths are arbitrary. They are only meant to visualize the stray from the global minimum.

and the images within the classes themselves. A multi-class dataset could very well have an imbalanced distribution of images, with one class having significantly less images than the others, which affects how well the algorithm is able to learn the defining characteristics. Similarly, should a new dataset be applied, one where smokestacks are tested against different classes than ones provided in the fMoW set, the smokestacks could require more, or less images to achieve the highest F-score. Another limitation is the photos themselves. Photos which exhibit characteristics from multiple classes (for example, a windmill that stands over farmland) creates confusion for the algorithm. It is likely the algorithm would select the feature that is most largely depicted. This then means that images of certain features must be taken so that the feature fills the image. As seen with many of the example images, this is not that case typically.

7. Conclusion

The highest scoring training data, in the instance of the smokestacks class tested against fMoW data, is a five class training/output set; this data arrangement sharpens the non-smokestack definition enough to guard against overfitting, but does not overload the algorithm with output classes, and creating mass amounts of false positives/false negatives within the smokestack class. While this research is applied to smokestacks specifically, it shows the general concept of why it is important to consider not only the training inputs for a machine learning algorithm, but how the inputs are structured as well. In implicating a deep learning algorithm, it is important to consider how the training data is gathered and structured. In searching for smokestack images among thousands of images from other classes, the makeup of the non smokestack classes directly impacts the performance of the algorithm, that being its ability to correctly label smokestack images and not overfit the data to include non-smokestacks as positive predictions. It is seen here that continuing to add classes only decreases the accuracy of the smokestacks class, yet diluting the training data to a simple smokestacks vs non-smokestacks situation also does not provide the highest accuracy values.

References

Ball, J. E., Anderson, D. T., & Chan, C. S., Sr. (2017). Comprehensive survey of deep learning in remote sensing: theories, tools, and challenges for the community. *Journal of Applied Remote Sensing*, *11*(4), 042609. https://doi.org/10.1117/1.JRS.11.042609

Bera, S., & Shrivastava, V. K. (2020). Analysis of various optimizers on deep convolutional neural network model in the application of hyperspectral remote sensing image classification. *International Journal of Remote Sensing*, *41*(7), 2664–2683. https://doi.org/10.1080/01431161.2019.1694725

Bertazzi, P. A., Bernucci, I., Brambilla, G., Consonni, D., & Pesatori, A. C. (1998). The Seveso studies on early and long-term effects of dioxin exposure: a review. *Environmental Health Perspectives*, *106 Suppl 2*, 625–633. https://doi.org/10.1289/ehp.98106625

Birnbaum, L. S. (1994). The mechanism of dioxin toxicity: relationship to risk assessment. *Environmental Health Perspectives*, *102 Suppl 9*, 157–167. https://doi.org/10.1289/ehp.94102s9157

Christie, G., Fendley, N., Wilson, J., & Mukherjee, R. (2018). Functional map of the world. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6172–6180. http://openaccess.thecvf.com/content_cvpr_2018/html/Christie_Functional_Map_of_CVPR_2018_paper.html

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248–255. https://www.researchgate.net/profile/Li_Jia_Li/publication/221361415_ImageNet_a_Large-Scale_Hierarchical_Image_Database/links/00b495388120dbc339000000/ImageNet-a-Large-Scale-Hierarchical-Image-Database.pdf

Durand, N., Derivaux, S., Forestier, G., Wemmert, C., Gancarski, P., Boussaid, O., & Puissant,

A. (2007). Ontology-Based Object Recognition for Remote Sensing Image Interpretation. *19th IEEE International Conference on Tools with Artificial Intelligence(ICTAI 2007)*, *1*, 472–479. https://doi.org/10.1109/ICTAI.2007.111

Fellbaum, C. (1998). *WordNet: An Electronic Lexical Database*. MIT Press. https://play.google.com/store/books/details?id=Rehu8OOzMIMC

Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., & Lew, M. S. (2016). Deep learning for visual understanding: A review. *Neurocomputing*, *187*, 27–48. https://doi.org/10.1016/j.neucom.2015.09.116

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778. http://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4700–4708. http://openaccess.thecvf.com/content_cvpr_2017/html/Huang_Densely_Connected_Convolutional_CVPR_2017_paper.html

Huang, Z., Pan, Z., & Lei, B. (2017). Transfer Learning with Deep Convolutional Neural Network for SAR Target Classification with Limited Labeled Data. *Remote Sensing*, *9*(9), 907. https://doi.org/10.3390/rs9090907

Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. In *arXiv [cs.LG]*. arXiv. http://arxiv.org/abs/1412.6980

Kohlbrenner, D. A., Jamison, T. A., & Kedzierski, L. M. (2009). Automated extraction of vertical obstructions from Lidar data. *ASPRS/MAPPS 2009 Fall Conference*. http://www.asprs.org/a/publications/proceedings/sanantonio09/Kohlbrenner.pdf

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 25* (pp. 1097–1105). Curran Associates, Inc. http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

Kussul, N., Lavreniuk, M., Skakun, S., & Shelestov, A. (2017). Deep Learning Classification of Land Cover and Crop Types Using Remote Sensing Data. *IEEE Geoscience and Remote Sensing Letters*, *14*(5), 778–782. https://doi.org/10.1109/LGRS.2017.2681128

Le, Q. V., Ngiam, J., Coates, A., Lahiri, A., Prochnow, B., & Ng, A. Y. (2011). On optimization methods for deep learning. In *ICML*. https://openreview.net/pdf?id=Sk4lD3W_bB

Lévy, D., & Jain, A. (2016). Breast Mass Classification from Mammograms using Deep Convolutional Neural Networks. In *arXiv [cs.CV]*. arXiv. http://arxiv.org/abs/1612.00542

Ma, L., Liu, Y., Zhang, X., Ye, Y., Yin, G., & Johnson, B. A. (2019). Deep learning in remote sensing applications: A meta-analysis and review. *ISPRS Journal of Photogrammetry and Remote Sensing: Official Publication of the International Society for Photogrammetry and Remote Sensing* , *152*, 166–177. https://doi.org/10.1016/j.isprsjprs.2019.04.015

Mayer, H. (1999). Automatic Object Extraction from Aerial Imagery—A Survey Focusing on Buildings. *Computer Vision and Image Understanding: CVIU*, *74*(2), 138–149. https://doi.org/10.1006/cviu.1999.0750

Minetto, R., Pamplona Segundo, M., & Sarkar, S. (2019). Hydra: An Ensemble of Convolutional Neural Networks for Geospatial Land Classification. *IEEE Transactions on Geoscience and Remote Sensing: A Publication of the IEEE Geoscience and Remote Sensing Society*, *57*(9), 6530–6541. https://doi.org/10.1109/TGRS.2019.2906883

Pesatori, A. C., Zocchetti, C., Guercilena, S., Consonni, D., Turrini, D., & Bertazzi, P. A. (1998). Dioxin exposure and non-malignant health effects: a mortality study. *Occupational and Environmental Medicine*, *55*(2), 126–131. https://doi.org/10.1136/oem.55.2.126

Schmitz, A., Bansho, Y., Noda, K., Iwata, H., Ogata, T., & Sugano, S. (2014). Tactile object recognition using deep learning and dropout. *2014 IEEE-RAS International Conference on Humanoid Robots*, 1044–1050. https://doi.org/10.1109/HUMANOIDS.2014.7041493

Shanmuganathan, S. (2016). Artificial Neural Network Modelling: An Introduction. In S. Shanmuganathan & S. Samarasinghe (Eds.), *Artificial Neural Network Modelling* (pp. 1–14). Springer International Publishing. https://doi.org/10.1007/978-3-319-28495-8_1

Song, J., Gao, S., Zhu, Y., & Ma, C. (2019). A survey of remote sensing image classification based on CNNs. *Big Earth Data*, *3*(3), 232–254. https://doi.org/10.1080/20964471.2019.1657720

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research: JMLR*, *15*(1), 1929–1958. https://dl.acm.org/doi/abs/10.5555/2627435.2670313

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1–9. https://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Szegedy_Going_Deeper_With_2015_CVPR_paper.html

Zeiler, M. D., Taylor, G. W., Fergus, R., & Others. (2011). Adaptive deconvolutional networks for mid and high level feature learning. *ICCV*, *1*, 6. https://www.matthewzeiler.com/mattzeiler/adaptivedeconvolutional.pdf

Zhang, L., Zhang, L., & Du, B. (2016). Deep Learning for Remote Sensing Data: A Technical Tutorial on the State of the Art. *IEEE Geoscience and Remote Sensing Magazine*, *4*(2), 22–40. https://doi.org/10.1109/MGRS.2016.2540798

Zhu, X. X., Tuia, D., Mou, L., Xia, G., Zhang, L., Xu, F., & Fraundorfer, F. (2017). Deep Learning in Remote Sensing: A Comprehensive Review and List of Resources. *IEEE Geoscience and Remote Sensing Magazine*, *5*(4), 8–36. https://doi.org/10.1109/MGRS.2017.2762307

Zou, Q., Ni, L., Zhang, T., & Wang, Q. (2015). Deep Learning Based Feature Selection for Remote Sensing Scene Classification. *IEEE Geoscience and Remote Sensing Letters*, *12*(11), 2321–2325. https://doi.org/10.1109/LGRS.2015.2475299

Vita

Kenny Moss attended Auburn University, graduating in 2017 with a bachelor's degree in sciences, in the geography department. He began school in 2017 at the University of Tennessee-Knoxville in pursuit of a master's degree in geology, switching to a master's degree in geography. He did multiple internships with the National Geospatial-Intelligence Agency through his time in Knoxville, and completed his thesis in partnership with Oak Ridge National Laboratory.