NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

CONVOLUTIONAL NEURAL NETWORKS FOR DETECTION AND CLASSIFICATION OF MARITIME VESSELS IN ELECTRO-OPTICAL SATELLITE IMAGERY

by

Katherine Rice

December 2018

Thesis Advisor: Michael R. McCarrin
Co-Advisor: Lyn R. Whitaker

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 2018	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE CONVOLUTIONAL NEURAL NETWORKS FOR DETECTION AND CLASSIFICATION OF MARITIME VESSELS IN ELECTRO-OPTICAL SATELLITE IMAGERY	**5. FUNDING NUMBERS**
6. AUTHOR(S) Katherine Rice	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

The ability to locate and identify vessels of interest in satellite imagery plays a vital role in maintaining maritime security. Recent studies have demonstrated that convolutional neural networks can be used to automatically classify or detect ships in satellite images; however, this technique requires large amounts of training data and computational power that may not be readily available in an operational environment. We seek to show that the process of transfer learning can be used to adapt open source convolutional neural network architectures pre-trained on large datasets to Department of Defense-specific image classification and detection tasks. We test this hypothesis by retraining both the VGG-16 image classification architecture and a VGG-16 based Single Shot Detector on a dataset comprised of satellite images containing ships. We first examine the performance of these retrained networks on the single category task of classifying or detecting ships in satellite imagery. We then evaluate model performance on datasets in which a fraction of the images contains blur and noise to simulate degraded satellite imagery. Finally, we test the models' ability to distinguish between subcategories of ships. We show that transfer learning can be leveraged to reduce both the size of training set and the training time required to produce an effective classification or detection model to meet the Department of Defense's analysis needs.

14. SUBJECT TERMS deep learning, machine learning, image classification, object detection, transfer learning, satellite imagery, single shot detector, convolutional neural network	**15. NUMBER OF PAGES** 79
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

CONVOLUTIONAL NEURAL NETWORKS FOR DETECTION AND CLASSIFICATION OF MARITIME VESSELS IN ELECTRO-OPTICAL SATELLITE IMAGERY

Katherine Rice
Lieutenant, United States Navy
BS, Appalachian State University, 2008

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 2018

Approved by: Michael R. McCarrin
Advisor

Lyn R. Whitaker
Co-Advisor

Peter J. Denning
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The ability to locate and identify vessels of interest in satellite imagery plays a vital role in maintaining maritime security. Recent studies have demonstrated that convolutional neural networks can be used to automatically classify or detect ships in satellite images; however, this technique requires large amounts of training data and computational power that may not be readily available in an operational environment. We seek to show that the process of transfer learning can be used to adapt open source convolutional neural network architectures pre-trained on large datasets to Department of Defense-specific image classification and detection tasks. We test this hypothesis by retraining both the VGG-16 image classification architecture and a VGG-16 based Single Shot Detector on a dataset comprised of satellite images containing ships. We first examine the performance of these retrained networks on the single category task of classifying or detecting ships in satellite imagery. We then evaluate model performance on datasets in which a fraction of the images contains blur and noise to simulate degraded satellite imagery. Finally, we test the models' ability to distinguish between subcategories of ships. We show that transfer learning can be leveraged to reduce both the size of training set and the training time required to produce an effective classification or detection model to meet the Department of Defense's analysis needs.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

API	Application Program Interface
BOW	Bag of Words
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
cuDNN	CUDA Deep Neural Network Library
DoD	Department of Defense
FN	False Negative
FP	False Positive
FPR	False Positive Rate
GPU	Graphics Processing Unit
HOG	Histogram of Oriented Gradient
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
NPS	Naval Postgraduate School
R-CNN	Regions with CNN features
ReLU	Rectified Linear Unit
ROC	Receiver Operating Characteristics
SGD	Stochastic Gradient Descent
SSD	Single Shot Detector
SVD	Singular Value Decomposition
SVM	Support Vector Machine
TP	True Positive
VGG	Visual Geometry Group

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank a number of people who where instrumental in helping me complete this thesis. Thanks to my thesis advisors, Michael McCarrin and Dr. Lyn Whitaker, for helping me grow in my understanding of machine learning and computer vision, and for their guidance during the writing process. Thanks to Dr. Marcus Stefanou for helping me get this project started and for being my sounding board during the early stages of experimental design.

Thanks to Dr. Griffith Parks and Dr. Rajendar Deora for teaching me the fundamentals of the research process during my time at Wake Forest School of Medicine. I wouldn't be where I am today if they had not encouraged me to find a career that I could be passionate about. Although this work is very different from where I started in the field of Microbiology and Immunology, the skills I learned while working in their labs were invaluable during this process.

Thanks to all of the professors that I had during my time at the Naval Postgraduate School. I learned a tremendous amount from them.

Last, but not least, thanks to my friends and family for their support during this process and for their patience as they put up with all of my ramblings about convolutional neural networks over the past several months.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Electro-optical satellite imagery is a key resource used by the intelligence community to increase our National Security posture. However, due to large volumes of data produced it can be difficult for an intelligence analyst to both identify images of interest and conduct further in-depth analysis on these images in a timely manner. Recent advancements in Convolutional Neural Networks (CNN), may now provide a mechanism by which the Department of Defense (DoD) can address this workflow issue.

Competitions such as the ImageNet Large Scale Visual Recognition Challenge [1] have demonstrated the ability of CNNs to be trained to correctly classify or detect objects in photographic images from a wide variety of categories. In recent work at the Naval Postgraduate School (NPS), Paul [2] shows that the open source CNN software and frameworks developed as part of this competition can be adapted for use on DoD-related image sets through a technique known as transfer learning. This thesis demonstrates that these open source CNN resources for object classification and object detection can also be easily adapted for use with satellite imagery. We show that new machine learning models can be trained quickly and with a limited number of training examples through the technique of transfer learning. Applying this technique to DoD-related object classification and detection tasks will greatly reduce the amount of time required for analysts to identify satellite images of interest, allowing them more time to analyze the significance of what is contained within the imagery and to pass that information along quickly to those who need it. Additionally, due to the reduced resource requirements of transfer learning, this technique has the potential to provide operational units with a quick reaction capability to train new image classification and object detection models as the need arises.

A. PROBLEM STATEMENT

Remote sensing technology provides vital capabilities to the intelligence community through the collection of large amounts of data from geographic areas of interest. However, not all data gathered has tactical or strategic significance. The raw data that is collected by these sensors requires further analysis to locate unusual activity or the

presence of targets of interest in an area. In many cases, the extraction of this critical information is time sensitive and can be used to tip other remote sensors to collect data from an area, or to provide vital indications and warnings to both warfighters and decision makers. As technology continues to improve, it has become increasingly difficult for analysts to visually inspect the enormous amount of accumulated sensor data. This issue can be addressed through the development of machine learning algorithms optimized for detecting and classifying objects of specific interest to the DoD in satellite imagery. However, these techniques are often time and resource intensive, requiring large quantities of training data and state-of-the art GPUs to produce high performing models. In this thesis, we aim to show that the technique of transfer learning can be applied to open source CNN architectures pretrained on large publicly available datasets such as ImageNet [3] to train new CNNs for DoD-related satellite imagery detection and classification tasks. Use of this training technique will greatly reduce the resources needed to train these new CNNs.

B. RESEARCH QUESTIONS

To address the problem of reducing the resources required to train new CNNs for DoD use with satellite imagery we pose the following research questions. The first set of questions relates to the task of classifying objects in satellite imagery, while the second addresses whether the same techniques can also be applied to the task of object detection in satellite imagery.

1. How does a CNN trained initially on a large set of photographic images perform when the process of transfer learning is applied to retrain it for the task of classifying objects in satellite imagery?

    - What is the impact of reducing training set size on the recall and precision of a CNN retrained to classify objects in satellite imagery?

    - What effect does the presence of degraded satellite imagery and imagery of different resolutions have on the recall and precision of a CNN retrained via transfer learning?

- How does a retrained CNN perform when given the task of classifying sub-categories of ships in satellite imagery as measured by recall and precision?

2. How does a CNN based object detection model trained initially on a large set of photographic images perform when the process of transfer learning is applied to retrain it for the task of detecting objects of interest in satellite imagery?

    - How does the presence of degraded satellite imagery and imagery of different resolutions impact the recall and precision of the retrained object detection model?

    - What is the maximum recall and precision that can be achieved by the retrained object detection model when given the task of detecting sub-categories of ships in satellite imagery?

3. Does transfer learning reduce the required training time to a level where a GPU is no longer required to train new models for DoD use?

C. CONTRIBUTIONS

This thesis makes the following contributions:

- We demonstrate that we are able to apply the technique of transfer learning to open source CNN architectures to classify satellite imagery containing ships with a recall of 1.00 and a precision of 0.98.

- We demonstrate that the technique of transfer learning can be applied to train a CNN model to classify satellite imagery with as few as 200 training images and still produce recall and precision scores above 0.95.

- We demonstrate that the introduction of images containing noise and blur to our test set causes a reduction in recall and precision. Additionally, we show that the inclusion of noise and blur in the training dataset helps to mitigate this negative effect.

- We demonstrate that the transfer learning technique applied in this thesis can be used to train a CNN to classify satellite images containing sub-categories of ship images with an average recall of 0.83 and an average precision of 0.85.

- We demonstrate that transfer learning can also be applied to an open source object detection architecture to detect ships in satellite imagery with a recall of 0.88 and a precision of 0.95. We also show that the presence of noise and blur also have a negative impact on object detection performance, and that our object detection training method can be used to detect sub-categories of ships with an average recall of 0.82 and an average precision of 0.86.

- Lastly, we demonstrate that the use of transfer learning reduces the training resource cost to a point at which the use of a GPU is no longer required for the task of image classification.

D. THESIS ORGANIZATION

In Chapter II, we discuss key technical concepts related to the understanding of CNN architectures, the difference between object classification and object detection, transfer learning, a brief overview of the tools used in this thesis, and a description of the performance metrics we use for our analysis. In Chapter III, we describe previous work conducted in the fields of object classification and detection. Chapter IV outlines the methodology used for our experiments and we provide analysis of the results in Chapter V. Finally, in Chapter VI, we discuss how our results address the research questions outlined above, as well as possible future work in this research area.

II. TECHNICAL BACKGROUND

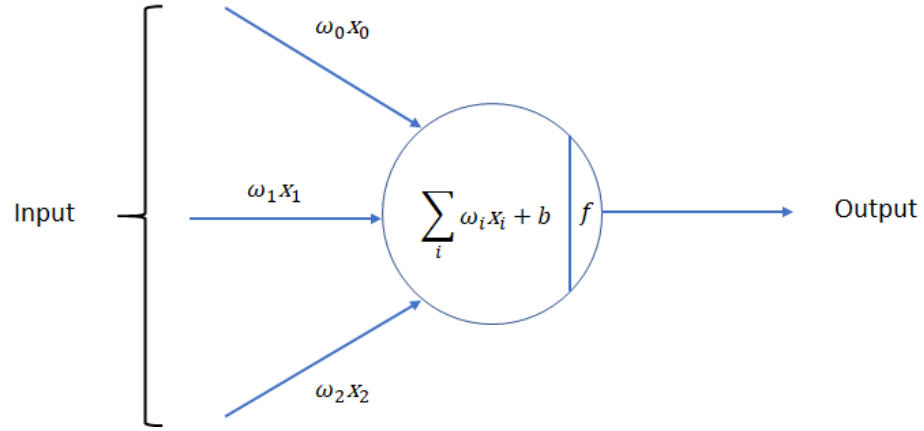
This chapter discusses several key technical concepts relevant to the experiments we conduct in this thesis as well as the open source software libraries we use to build our models. We first briefly describe the basic components of CNN architectures. This is followed by a discussion of the difference between object classification and object detection, and a description of the single shot detector (SSD) method of object detection. We then describe the concepts behind the process of transfer learning which is used heavily in our work. Finally, we discuss the open source tools used in this thesis and the performance metrics used to evaluate model performance in our experiments.

A. CONVOLUTIONAL NEURAL NETWORK CONCEPTS

Here, we give an overview of basic concepts necessary to understand the CNN architectures used in this thesis. We first describe the neurons that make up CNNs. We then discuss the types of layers used to construct a CNN: convolutional, pooling, and fully connected. Lastly, we briefly describe the hyperparameters used in our model.

1. Artificial Neurons

The basic building block that makes up a CNN is the neuron. As the name suggests, artificial neurons are inspired by the neurons that make up the human nervous system. Neurons take input signals from one or multiple sources and produce a single output based on these inputs. The output is produced by calculating the sum of each input to the neuron multiplied by its weight, adding the neuron's bias to it and then passing this value through an activation function (see Figure 1). Weights are real numbers that are used to express the relative importance of different input signals to that neuron. It is these weights, and the biases, that are updated during CNN training to create a model suited to a specific task such as the ship classification and detection tasks in this thesis. An activation function is then used to determine the final output of the neuron. Activation functions are discussed in further detail below. The single output from this activation function is then fed forward into the next layer of the CNN and the above process is repeated at each neuron in that layer [4].



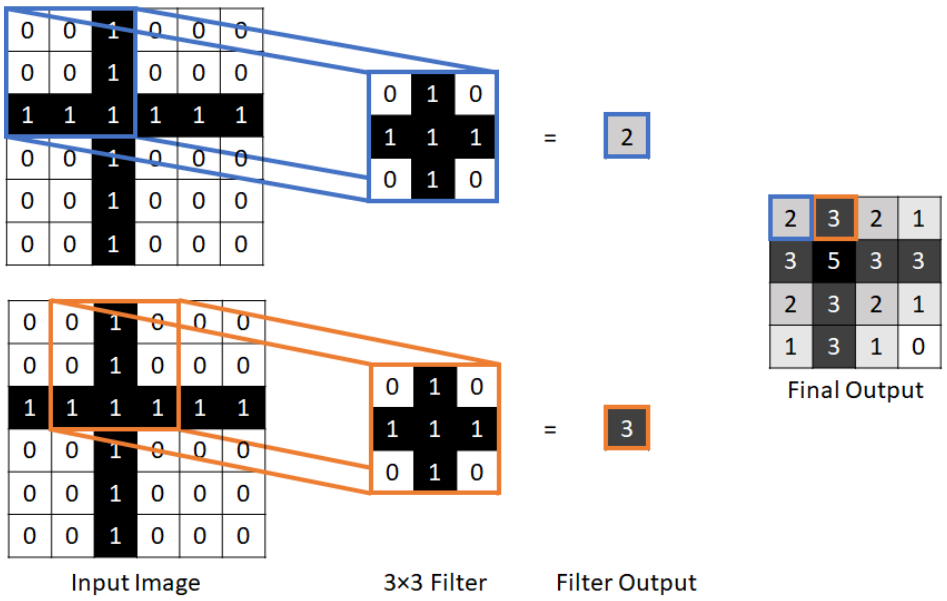
In this example, there are three inputs to the neuron denoted by x_0, x_1, and x_2. These inputs are then multiplied by their respective weights (ω_0, ω_1, and ω_2), and the sum of all three inputs is taken and added to the bias (b). This result is then passed through the activation function (f) to produce the final output.

Figure 1. Example of basic CNN neuron structure. Adapted from [5].

2. CNN Layers

In this section, we briefly cover the basic types of layers used to construct CNN architectures. We begin with a discussion of convolutional layers. We then discuss the function of max pooling layers and conclude with a description of the fully connected layers used during final classification determination. CNNs have become a popular machine learning method for use with image classification and detection tasks in part due to their ability to scale better than regular neural networks for large images. This scaling ability is due in part to the fact that the neurons in convolutional layers are connected only to neurons representing a nearby region of the image. In addition, all neurons in a convolutional layer share a common set of weights, which simulates the effect of applying a discrete convolutional filter to the input. This greatly reduces the number of parameters required during the early training layers and helps to reduce the potential for overfitting [5]. A simple example of this operation is shown in Figure 2. In this figure, each square of the image represents a pixel, with black pixels having a value of 1 and white pixels a value of 0. An example 3×3 filter of weights is then applied to this simple image and the resulting output is shown on the right (in practice, these weights are updated during CNN training). In this example, we use a stride of one, meaning that the filter is moved by one pixel each time it is applied to the image. Convolutional filters applied in this manner are able to detect

patterns in an image such as straight edges, curves, and other shapes. The filter in Figure 2 can be interpreted visually as two intersecting lines. When it is applied to the region of the image that most closely matches this pattern it produces its highest output score, indicating the presence of a similar shape in that region. Alternatively, when applied to a region of the image that is completely absent of any lines, the filter produces its lowest output score, indicating the absence of a similar shape to the filter.



Observe that the strongest response in the final output occurs when the pattern represented by the filter aligns with a matching pattern in the original input image.

Figure 2. Example showing a 3×3 convolution applied to a simple image with a stride of one.

The second major layer type found in CNN architectures is the pooling layer. Pooling layers are generally located between convolutional layers within the network architecture and are used to further reduce the spatial size of the network by summarizing regions of the image after they have been convolved with a filter as described above. This summarization is possible because the network does not need to know the exact location of a feature but is instead concerned with its relative location to other features in the image [4]. The ability to summarize feature locations in this way in turn reduces the number of parameters and required computations in subsequent layers. Two common strategies for

pooling are max pooling and average pooling. Of the two methods, max pooling has been found to produce better performance; however, average pooling was the favored technique in many early CNN architectures [5]. Examples of each of these pooling methods are provided in Figure 3. This example uses a 2×2 pooling filter with stride two. This means that either the maximum or the average value will be taken for each 2×2 area of the output matrix from the previous convolution. These values will then produce a new output matrix of smaller size for use as input to the next layer of the CNN. The VGG-16 CNN we use in our experiments implements the max pooling method [6]. The VGG-16 architecture is described in further detail in Chapter III.

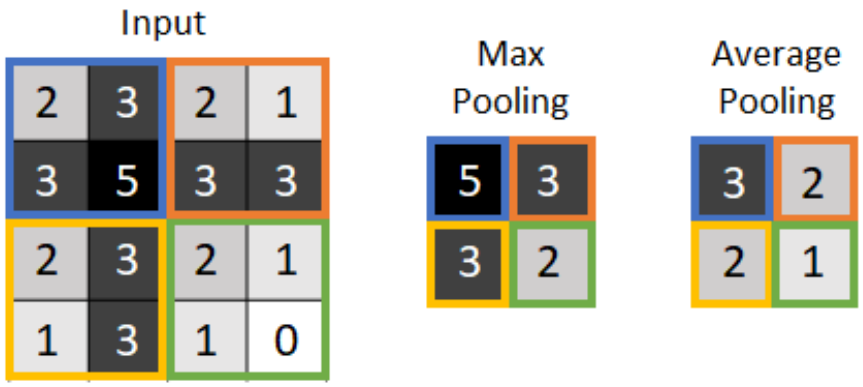


Figure 3. Example of max and average pooling conducted on the output of our convolution example from Figure 2 using 2×2 pooling and a stride of two.

The last of the major layer types found in CNNs is the fully connected layer. In this type of layer, each neuron in one layer is connected to every neuron in the subsequent layer. These fully connected layers are typically located after the convolutional and max pooling layers, and the final fully connected layer will contain a number of output neurons equal to the number of classes in the dataset the model is trained on [4]. A final activation function is then applied to the output from this last set of neurons to produce a predicted classification for the image that was passed through the CNN. Activation functions are discussed in further detail below.

3. CNN Hyperparameters

Here, we discuss the hyperparameters used in our CNN architecture: batch size, training epochs, dropout, activation functions, loss functions, and optimization functions. For CNNs batch size represents the number of training examples passed into the neural network at one time. The full training set is divided into these batches and then each batch is passed through the neural network. Weights are updated after a batch has been fed through the neural network. An epoch is considered to have occurred once the entire dataset (all of the batches) has been passed through the neural network. The number of epochs required for training varies depending on the dataset in use and if too high or too low can have negative impacts on model performance due to underfitting or overfitting [7]. Overfitting is caused when a CNN learns complicated relationships that are only present in the training set but not in the test set and can be limited through the use of dropout. Dropout refers to the process of temporarily dropping certain neurons from the CNN including all of their connections to other neurons. The neurons that are removed are selected at random and this helps prevent the CNN from learning relationships that are overly specific to the training data [8].

Activation functions are the parameter that is used to determine what values are output from neurons at one layer of a neural network and passed on as input to the next layer, or alternatively used to make the final classification determination [5]. The CNN architecture used in this thesis employs the Rectified Linear Unit (ReLU) activation function following each intermediate layer of the network, and the softmax classifier for the classification determination in the final layer [6]. The ReLU function is [5]

$$f(x) = max(0, x) . \quad (1)$$

This function has been found to greatly reduce the time required to train deep CNNs when compared to more traditional models such as the sigmoid and hyperbolic tangent activation functions [9].

For classification neural networks, the activation function used for the final layer is often the softmax activation function. This activation function outputs scores between zero and one that can be interpreted as probabilities that the final output belongs to each of the

final categories. These probability scores will always sum to one [5]. Specifically, when classifying the neural network input (e.g. image) as one of K categories, the last layer has K neurons. For j = 1,…, k, let f_j be the combined input to the j^{th} neuron; then the softmax activation function applied to the i^{th} neuron yields

$$\frac{e^{f_i}}{\sum_{j=1}^{K} e^{f_j}}. \tag{2}$$

We use these probability scores to examine performance at differing thresholds for two category classification tasks, and to examine both top-1 and top-2 performance of multi-category classification tasks as described in further detail in Chapter IV.

Loss functions are the parameter that measures how well the CNN is classifying the training data based on the output of the final activation function. Cross-entropy loss is a common loss function used to measure the difference between the true probability distribution (represented by correctly labeled classes) and the estimated distribution represented by predicted probabilities. Cross-entropy loss is logarithmic and increases as the predicted probability gets farther from the true classification of the training data [10]. It takes the form [5] of

$$L_i = -log\left(\frac{e^{f_i}}{\sum_{j=1}^{K} e^{f_j}}\right), \tag{3}$$

where the formula inside the parenthesis is the softmax function previously discussed above.

Optimization functions are the parameters that then determine how the weights and biases at each neuron should be updated to minimize the output of the loss function. Our work uses two optimization functions that build upon the stochastic gradient descent (SGD) method of optimization. Gradient descent calculates the gradient of the loss function with respect to the weights of the neuron and then updates the weights in the opposite direction until a local minimum is reached. SGD improves over traditional gradient descent by performing this parameter update after each training example [11]. We use the RMSProp optimization algorithm for our object classification experiments and the adam optimization algorithm for our object detection experiments. RMSProp speeds up the gradient descent

process by keeping an exponentially weighted average of the squares of past gradients. The learning rate is then divided by this average [12]. Adam is designed to combine the advantages of the RMSProp optimizer and the Momentum optimizer. In addition to following the procedure used for RMSProp, it also uses a "smooth" version of the gradient instead of the raw gradient vector and includes a bias correction mechanism [5].

B. OBJECT CLASSIFICATION AND OBJECT DETECTION

In this section, we explain the primary differences in the related tasks of object classification and object detection. We then briefly describe the SSD technique for object detection that we apply to the experiments in this thesis.

1. Object classification versus object detection

To understand the two categories of experiments we conduct as part of this thesis, it is important to understand the difference between the process of object classification and object detection. Object classification refers to the process of determining the category of an object present in an image. The performance of an object classification model is measured based on how well it correctly labels the object. It does not, however, attempt to identify the number of objects within an image or their relative locations. This is the task of object detection models. Object detection refers to the process of determining the classes and locations of multiple objects within the same image and drawing bounding boxes around them. While CNN architectures constructed using the components described above are able to successfully complete object classification tasks, additional computational steps must be added to CNNs in order for them to complete the more complicated object detection tasks.

2. Single Shot Detector Method for Object Detection

Here, we briefly describe the SSD method for object detection that we use in our second set of experiments. SSD is a technique developed by Lie et al. [13] in an effort to reduce the training time and computational resources required for accurate object detection in images. Unlike previous object detection techniques such as Regions with CNN features (R-CNN) [14], the SSD technique does not require resampling of pixels and is able to

perform object detection with a single pass through a CNN. This is accomplished by using a base network composed of a CNN designed for image classification such as the VGG-16 model we use in this thesis. The final classification layers are stripped from this base model and additional convolutional layers of decreasing size are added. These new layers allow for detection predictions at multiple scales. Predictions are made by evaluating a small set of default bounding boxes at each location in these new layers and producing shape offsets and confidence values for all object classes present in the dataset. These predictions are then compared with the ground truth boxes in the training set to determine model loss [13]. This thesis uses the SSD300 object detection model based on the VGG-16 architecture. A diagram of the SSD300 architecture is shown in Figure 4 and the VGG-16 base architecture is discussed in further detail in Chapter III.

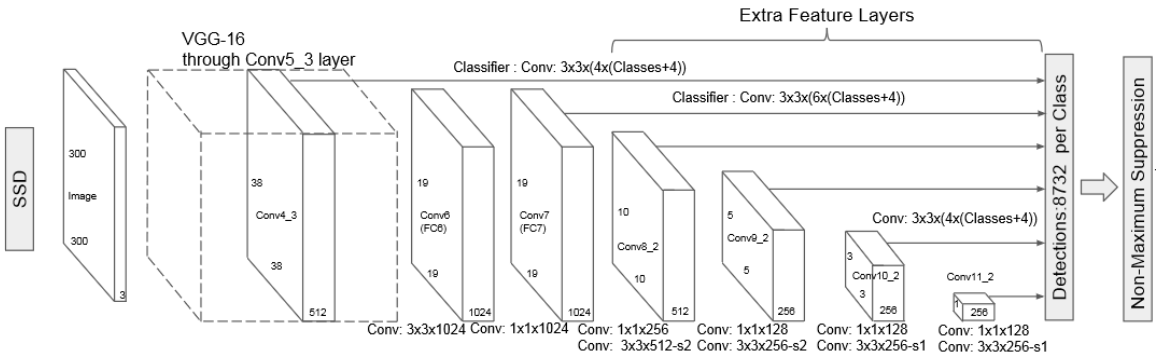


Figure 4. SSD300 model architecture. Source: [13].

C. TRANSFER LEARNING

Having covered some of the important concepts related to CNN architectures and the SSD method of object detection, we now briefly discuss the idea of transfer learning which is a primary focus of our work. The technique of transfer learning was developed as a means to address the issue of training CNNs on new tasks when there is a limited amount of training data available [15]. This is accomplished by using features extracted from a CNN trained on a large dataset such as ImageNet and then applying them to a new image classification or object detection tasks. Specifically, features produced by passing training

images through the pretrained network are extracted from the penultimate layer, the layer just prior to the last fully connected classification layer. These extracted features are then fed into new fully connected layers to train a classifier for the new task [15]. This allows for successful training on smaller datasets because low and mid-level features, which are common across different datasets, are learned on the large initial dataset, while the small dataset is used only to associate these features with the specific categories in the new classification task.

A second benefit to transfer learning is the ability to reduce the required training time for the model through the use of what is commonly referred to as bottleneck features [16]. The term bottleneck features refers to the process of running training images through the pretrained CNN layers only once and saving the extracted features into an array. These saved features are then applied to the new fully connected classification layers during each training epoch. This results in each image in the training set only needing to pass through the full CNN architecture once [16]. We show in our results that this reduction in training time is enough to allow for training of effective models for DoD use on a CPU only.

D. TOOLS

In this section, we briefly describe the two open source machine learning software libraries we use to build our object classification and detection models. The first library we use is TensorFlow. It is an open-source software library designed by Google that provides support for tasks requiring high performance numerical computation such as machine learning [17]. We select TensorFlow as the backend software library for our work because of its previous use in Paul's thesis as discussed in Chapter III [2]. While his work examined the use of TensorFlow with DoD related image categories, we aim to show that it can also be used to retrain CNN architectures for use with satellite imagery containing objects of interest to the DoD.

We use the Keras deep learning library for Python to simplify the coding requirements of building our models. Keras is a high-level application program interface (API) that runs on top of TensorFlow and contains numerous built in functions for building and testing machine learning models [18]. Additionally, Keras provides access to deep

learning models pretrained on the ImageNet dataset [19]. This greatly simplifies the Python code required to conduct our transfer learning tasks. We obtain satellite imagery from Planet Labs using their Planet Explorer imagery exploration tool [20]. Available satellite imagery was taken by the PlanetScope satellite constellation, which is capable of producing four-band images (blue, red, green, and near infrared) with an approximate ground sample distance of 3m [21].

E. PERFORMANCE METRICS

Performance for two-level classification is measured by counting the true positives (TP), false positives (FP), and false negatives (FN) where an input is classified as positive if its probability score is greater than a threshold. We evaluate thresholds ranging from 0.0 to 1.0 and incrementing by 0.1. Our definitions of TP, FP, and FN for our tasks will be discussed in further detail in Chapter IV. This information is then used to calculate recall, precision, false positive rate (FPR) and F-score for each threshold as defined in equations 4, 5, 6, and 7, respectively. The F-score is the harmonic mean of both precision and recall and can be used to determine the optimal threshold for a machine learning classifier [2]. Recall and FPR values are then used to plot a receiver operating characteristics (ROC) curve, which shows the tradeoffs between number of true positives versus number of false positives in a model [22]. These results are discussed in Chapter V.

Equations for recall, precision, FPR [22], and F-score [23] are given below.

$$recall = \frac{True\ Positives}{Positive\ Examples} \tag{4}$$

$$precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \tag{5}$$

$$FPR = \frac{False\ Positives}{Negative\ Examples} \tag{6}$$

$$F - score = \frac{(1+1)*precision*recall}{1*precision+recall} \tag{7}$$

III. RELATED WORK

Much of the prior work in object classification and detection focuses on improving CNN performance through training on large repositories of photographs such as the ImageNet database. This thesis builds upon that work by leveraging the VGG-16 CNN architecture previously trained on ImageNet in order to reduce the overhead required to train new models for use with satellite imagery. We briefly discuss a previous NPS thesis by Paul that demonstrates the potential for open source deep learning frameworks to be adapted for use with DoD related image sets. We then describe the composition of the ImageNet database, the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), and the development of the VGG-16 CNN architecture. We next discuss some of the challenges presented by working with satellite imagery explored in this thesis: image pre-processing requirements and the impact of degraded data on model performance. Lastly, we discuss related research that explores the use of CNNs for the task of ship classification and detection in satellite imagery.

A. OPEN SOURCE FRAMEWORKS FOR USE IN DOD RELATED IMAGE SETS

Paul [2] explores the idea that open-source deep learning frameworks can be leveraged by the DoD for use with intelligence related image sets. He uses the open-source software TensorFlow to test the performance of deep learning algorithms as a means for detecting AK-47s, ships, and screenshots [2]. This performance was compared with the previously used methods of object detection performed on similar data sets [24], [25], [26].

Paul [2] shows that TensorFlow-based CNN architectures provide a performance increase for classification tasks involving both AK-47 and ship images when compared with previous methods used by Jones [24] and Camp [25]. His screenshot detection experiments yield similar performance as those conducted by Sharpe. However, her method requires human-selected feature extractors, while CNNs do not require direct human input to determine which features to evaluate [2]. While Paul's work focuses on the detection and classification of DoD related objects in standard images, we expand upon

this work to examine the performance of TensorFlow and the publicly available VGG-16 architecture to the classification and detection of objects from satellite imagery.

B. IMAGENET, ILSVRC, AND THE DEVELOPMENT OF VGG-16

One of the challenges relating to CNNs is the requirement for large datasets and increased computing power when compared with legacy object classification and detection methods [2]. For this reason, our work focuses on the use of the technique of transfer learning, allowing us to make use of publicly available CNNs pretrained on the ImageNet database. Here, we briefly discuss the composition of ImageNet, the ILSVRC, and the VGG-16 architecture used in our work.

1. ImageNet Database

ImageNet, developed in 2009 by Deng et al. [3], represents an effort to organize the vast amount of multimedia data presently found on the Internet. At the time of its initial release ImageNet contained 3.2 million images and has since grown to over 14 million images. These images are organized based on the hierarchical method developed for use with WordNet. The WordNet hierarchy is based on the idea of synonymy, meaning that images labeled with similar terms such as “car and automobile” are grouped together into categories known as synonym sets or synsets. Synsets are then linked into a hierarchical structure based on super-subordinate relationships such that more general synsets are linked to more specific subcategories of that type of object [27]. Figure 5 demonstrates how ImageNet applies this hierarchical structure to organize images into increasingly specific synsets. ImageNet currently contains 27 high-level categories divided into a total of 21,841 synsets with an average of 743 images in each synset [19].

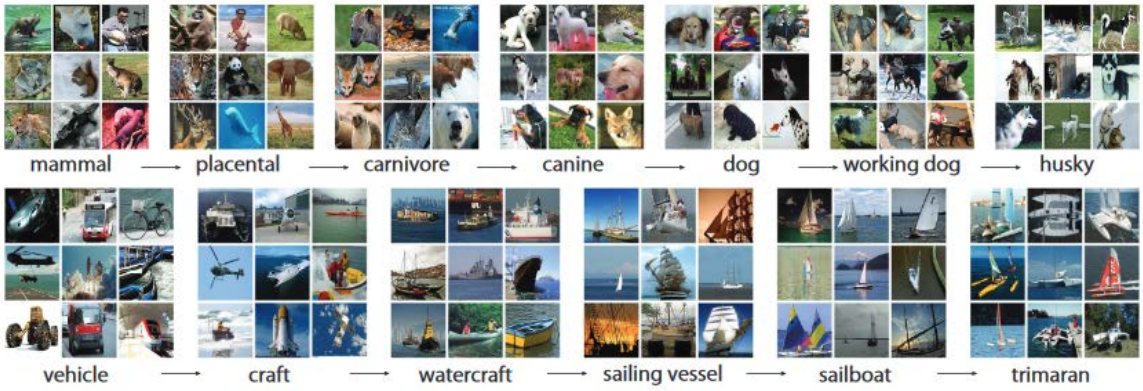


Figure 5. A snapshot of two ImageNet categories showing the hierarchical structure of synsets. Source: [3].

The images contained within the ImageNet database were obtained by conducting searches on image search engines using the set of WordNet synonyms that makes up that particular synset. Queries were also expanded by adding in terms from the parent synset as well as conducting queries in multiple languages. The task of verifying that all images are labeled correctly was conducted manually by a group of individuals selected using Amazon Mechanical Turk, a platform which allows users of the system to be paid for completing customer-defined tasks. Users were tasked to confirm that candidate images of a synset matched the definition of the target synset that they were given. To further ensure accuracy of image categorization, multiple users are tasked to independently review each image and an image is only considered an acceptable example of a synset if the majority of users vote that it belongs in that synset. Using this methodology, an average of 99.7% precision was obtained across 80 synsets, as determined through evaluation of random samples by an independent group of expert raters with label verification [3].

2. ImageNet Large Scale Visual Recognition Challenge

The large number, accuracy, and diversity of images contained within the ImageNet database led to the development of the ILSVRC which was run annually from 2010 to 2017 [1], [28]. The ILSVRC provides the computer vision research community with a publicly released dataset containing annotated training images as well as unlabeled test images. Teams wishing to participate in the challenge would train their model using the training

dataset and submit their automatically annotated test images to an evaluation server for scoring. Two categories of annotations were provided for training and scoring: image-level annotation and object-level annotation. Image-level annotations indicate the presence or absence of a particular category in an image and can be used to train for object classification, while object-level annotations include bounding boxes around a specific object within an image. This type of annotation is used for training object detection architectures [1].

The ILSVRC dataset contains a subset of 1,000 synsets selected from the ImageNet database for the image classification training set and an additional subset of 200 synsets for use in the object detection data set. In both cases, the synsets are selected such that there is no overlap between them; meaning that one is not the parent of the other in the ImageNet hierarchy. Table 1 includes details of the datasets provided as part of the ILSVRC [1]. Several notable CNN architectures have been produced as a result of this annual competition including: AlexNet [9], Inception [29], VGG [6], and ResNet [30]. The transfer learning conducted as part of our work used the VGG-16 architecture pretrained on the ILSVRC2012 classification data set and an SSD object detection model built on the VGG-16 architecture and pretrained on the ILSVRC2014 object detection data set.

Table 1. Summary of ILSVRC image classification and object detection datasets. Adapted from [1].

Year	Train Images	Validation Images	Test Images
Image Classification (1000 classes)			
ILSVRC2010	1,261,406	50,000	150,000
ILSVRC2011	1,229,413	50,000	150,000
ILSVRC2012-17	1,281,167	50,000	150,000
Object Detection (200 classes)			
ILSVRC2013	395909	21121	40152
ILSVRC2014-17	345854	21121	40152

3. VGG-16 Architecture

VGG-16 is a deep CNN architecture developed for ILSVRC-2014 by the Visual Geometry Group at the University of Oxford [6]. The VGG model sought to improve CNN performance on the ImageNet dataset by increasing the depth of the network architecture. This is accomplished by increasing the number of convolutional layers, while reducing their size to small (3×3) filters (compared to the 11×11 and 7×7 filters used by other architectures) to decrease the number of parameters without making the increased number of convolutional layers computationally infeasible. The 3×3 size is selected because it is the smallest filter size that is still able to recognize the ideas of center, up versus down, and left versus right when stepping across an image during the convolution process. Additionally, a stride of one is used such that the filter is convolved with the input at each pixel. The convolutional layers are interspersed with max pooling layers at regular intervals. These are followed by three fully connected layers, and a final softmax layer. The top two performing architectures from this work have 16 and 19 total weight layers (VGG-16 and VGG-19, respectively) [6]. This architecture was awarded second place in the object classification category of ILSVRC-2014 with a top-5 error percentage of 7.3% [1]. We use the VGG-16 version of this architecture which is diagramed in Figure 6.

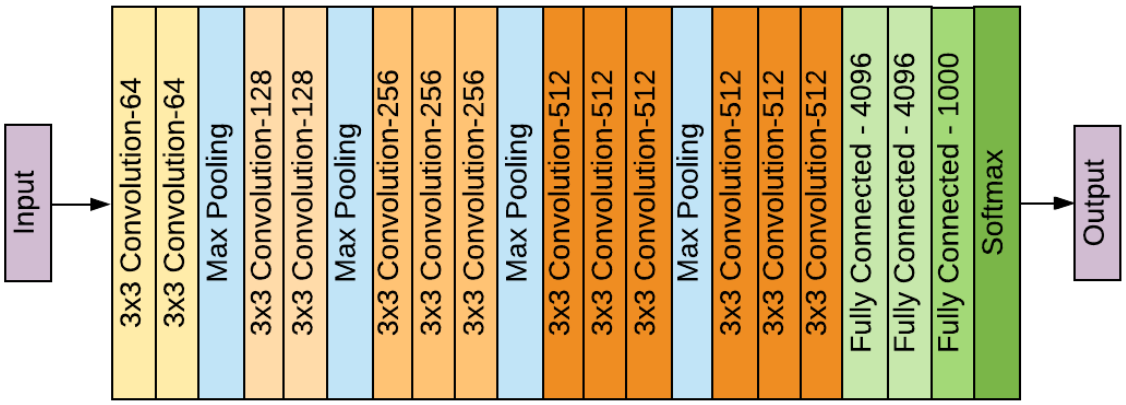


Figure 6. VGG-16 model architecture. Adapted from [6].

C. CHALLENGES PRESENTED BY THE USE OF SATELLITE IMAGERY

The use of object detection and classification models with satellite imagery is not without challenges. Here, we focus on two potential challenges that are addressed by our work. We briefly discuss the requirement for image pre-processing in previously explored methods of vessel classification and the potential impact of degraded satellite imagery on model performance.

1. Image Pre-processing Requirements

Much of the previous work conducted on object detection and classification in satellite imagery relies on human-selected feature extractors that are sensitive to differences in rotation and scale. Yao et al. suggests that this creates a requirement for candidate images to be rotated and resized before any features can be extracted [31]. Additionally, Rainey et al. explores the use of a Bag of Words (BOW) feature extraction algorithm to avoid this issue but found that other algorithms that required pre-processing were more effective. This pre-processing is conducted by manually rotating, cropping, aligning, and resizing each image prior to feature extraction and classification [32].

The use of CNNs for object detection and classification automates the feature generation process and is capable of producing effective classification results despite the presence of images in a variety of orientations. This feature of CNNs has also been used to augment small datasets when a large database such as ImageNet is not available. Yao et al. begins with a data set of only 36 images containing 1411 ships. Using a combination of rotations and cropping they scale their training data set up to a total of 22,683 ships of different scales and orientations for use in their ship detection network [31]. Although, their methodology produces a successful ship detection model (resulting in a recall of 0.92 and a precision of 0.78), the need to scale up their dataset to an effective training size still requires image pre-processing that increases the initial overhead of training the model. It does, however, improve on previous methods in that this image pre-processing is only required for initial training/testing, and not for applying the trained classifier as with previously used feature extraction methods. We show that this type of pre-processing is not required for small training sets if the technique of transfer learning is applied.

2. Impact of Degraded Satellite Imagery on Model Performance

Another challenge to working with satellite imagery is the potential variation in images collected from different sensors or during a range of collection windows. Imagery from different collection assets may be lower or higher in resolution, the image may be degraded by the presence of noise, or the image may be partially occluded by cloud cover. Rainey et al. demonstrate that the presence of these types of degradation may have negative impact on model performance if not accounted for during the training process [33]. They compare the effect of resolution difference, noise, and occlusion on the ability of a BOW feature extractor and support vector machine (SVM) classifier to correctly classify ships of different types in satellite images. Their results demonstrate that their model accuracy is much higher when the same degree of image degradation was present in both the testing set and the training set. These results indicate that when building a classifier, it is important to include examples of all variations you expect your model to encounter during use [33].

Our work aims to determine whether image degradations have a similar impact on CNN-based classifiers. These findings also demonstrate a potential problem with using available open source images for training DoD-focused CNN networks and provide another example showing that ability to conduct training on small data sets via transfer learning may be vital to this problem area. Although, resources such as the UCMerced data set [34] and datasets created for the DeepGlobe 2018 [35], DIUX xView [36], and Airbus Ship Detection Challenge [37] provide potential training sets, if the images are at different resolutions or have significantly different amounts of noise and occlusion examples, a model trained using them may perform poorly when used for images collected via DoD sensor platforms. Development of a model that can be quickly trained on a small dataset will allow analysts to train models tailored specifically for their classification and detection requirements.

D. RELATED RESEARCH ON THE USE OF CNNS FOR SHIP CLASSIFICATION AND DETECTION

In this section, we examine previously published research that also demonstrates the ability of CNNs to be used for the task of ship classification and detection in satellite

imagery. Rainey et al. examines whether a CNN can be trained to recognize different classes of ships in satellite imagery [38]. They use a moderately sized CNN architecture containing four convolutional layers and train their model on 21,934 images containing ships from four different ship classes as well as non-ship objects. Instead of training a single multi-category classification system, a separate binary classification model is trained to recognize each different ship class. Their results demonstrate that their model is able to correctly classify ship sub-categories with an average recall of 0.74 and an average precision of 0.25 across all categories [38]. In this thesis, we demonstrate that better performance can be obtained by applying transfer learning to the VGG-16 architecture pretrained on the ImageNet database.

Yao et al. and Nie et al. explore two methods for leveraging CNNs for the task of ship detection in satellite imagery. Yao et al. use a CNN to extract feature maps from satellite images and then apply a region proposal network to determine the precise location of each ship [31]. As mentioned previously, they use a dataset containing 22,683 ships for training and focus only on those ships larger than 20 pixels. Additionally, they do not label small ships with long wakes as positive examples. This method produced higher performance (see above) than previously conducted research using non-CNN based methods such as Histogram of Oriented Gradients (HOG) [39] and Singular Value Decomposition (SVD) networks [40].

Nie et al. explore the use of an SSD object detection model that uses a pretrained VGG-16 network as its base [41]. They train their model using a dataset containing satellite images of 3898 ships obtained from Google Earth with a 0.54m ground sample distance. They achieve a maximum F-score of 0.843 at a recall rate of 0.90 demonstrating that this method is able to achieve high performance levels [41]. This thesis expands upon their work by exploring model performance on images with lower ground sample distance, the addition of degradations to selected images in the dataset, and the ability of the model to detect sub-categories of ships in satellite imagery.

IV. METHODOLOGY

This chapter outlines the methods and techniques used in our research. We first describe the hardware and software used to conduct our experiments and discuss how we constructed our dataset. We then describe the methods we use in our first set of experiments, which focus on the task of ship classification in satellite imagery. In Experiment 1 we verify the baseline performance of our ship classification model. Next, Experiment 2 examines the impact of training dataset size on model performance. Experiment 3 then explores the impact of noise and blur on model performance, and Experiment 4 tests whether our ship classification model can also be trained to classify sub-categories of ships. Lastly, we describe the techniques we use in our second set of experiments, which focus on the task of ship detection in satellite imagery. In Experiment 5, we verify the baseline performance of our ship detection model. Experiment 6 then examines the impact of noise and blur on model performance, and Experiment 7 tests whether our ship detection model can also be trained to correctly detect and classify ship sub-categories.

A. HARDWARE AND SOFTWARE

Our CNNs are trained using a Windows 10 desktop computer with a NVIDIA GTX 1080 GPU and i7-7700 CPU. Full specifications are located in Table 2. Tensorflow version 1.11.0 is used as the backend for our neural network architectures and interfaces with the GTX 1080 GPU through use of NVIDIA's CUDA 9.0 API and CUDA Deep Neural Network library (cuDNN) 7.3. All code is written in Python 3.6.3 using the Keras library.

Table 2. Hardware specifications

GPU	**Clock Speed**	**Cores**	**Memory (GB)**	**Memory Interface**
GTX 1080	1.607 GHz	2,560	8	GDDR5X
CPU				
i7-7700	3.60 GHz	4	32	DDR4

B. DATASETS

To retrain the VGG-16 architecture to classify and detect objects in electro-optical satellite imagery, we obtain overhead images from Planet Labs using their Planet Explorer imagery exploration tool [20]. To build the dataset, we manually examined areas near major shipping lanes for images containing ships, which we then clipped out as smaller images of 100×100 and 300×300 pixels for classification and object detection experiments, respectively. In addition to ships, the dataset includes images containing clouds, open ocean, islands, and land to provide negative training and testing examples. A description of the full dataset is provided in Table 3 and example images from each subcategory are shown in Figure 7. This dataset is then randomly divided into training, validation, and test subsets as described in the task-specific sections below.

Table 3. Datasets for ship classification and detection.

	Ship Classification	**Ship Detection**
Ship	**1100**	**826**
Barge	220	202
Fast	220	235
Merchant	660	389
No Ship	**1100**	**437**
Cloud	287	220
Island	299	217
Land	220	N/A
Ocean	294	N/A

The numbers for ship classification represent 100x100 pixel satellite images. The numbers for ship detection represent the number of bounding boxes drawn for each category within a set of 484 300x300 pixel satellite images.

Figure 7. Example images from the 100×100 pixel ship classification dataset.

C. SHIP CLASSIFICATION TASKS

In this section we test the performance of our VGG-16 based ship classification model on a variety of ship classification tasks. We first verify baseline model performance. We then examine the impact of decreasing training size and the addition of noise and blur to our dataset. Lastly, we test whether our model can also be trained to classify ship sub-categories.

1. Experiment 1: Ship Classification Baseline

In Experiment 1, we verify whether our retrained VGG-16 model is able to correctly classify ships in satellite imagery. For this experiment, the ship classification dataset from Table 3 is randomly divided into training, validation, and test subsets containing 1800, 200, and 200 images, respectively. Each subset is evenly divided between images containing ships and those not containing ships. This dataset is then used to retrain the VGG-16 architecture described in Chapter III via transfer learning on a model previously trained on the ImageNet ILSVRC2012 dataset described in Table 1 [1]. To accomplish this task, we modify publicly available code to meet our requirements [42], [43]. Our modifications adjust the input size and number of categories to match our dataset,

change the final activation function to softmax, and adjust the output metrics to produce a list of predicted image classes and their respective probabilities. These alterations allow us to produce the ROC graph described in Chapter II as a measure of model performance.

Re-training the VGG-16 CNN is a two-step process (see Figure 8). We first obtain "bottleneck" features by running our images through the convolutional and max pooling layers of a VGG-16 network pretrained on the ImageNet database. The fully-connected classifier layers are not used in this process. Instead, the activation maps from the last max pooling layer for the training and validation datasets are saved in numpy arrays. This step reduces the computational requirements of retraining the network, because each image only has to pass through the main layers of the VGG-16 network once. These bottleneck features are then used to train our own fully-connected layers on the specific categories we are attempting to classify [42]. The model is trained using the following hyperparameters:

- Batch size = 16
- Number of Epochs = 50
- Activation = relu, softmax (final layer)
- Optimizer = RMSprop
- Loss = cross-entropy
- Dropout = 0.5

We then calculate TP, FP, and FN at probability thresholds ranging from zero to one and incrementing by 0.1. In our ship classification experiments, a TP is defined as an image that contains at least one example of a ship and is also classified as "ship". A FP is defined as an image that does not contain an example of a ship but is misclassified as "ship." Lastly, a FN is defined as an image that contains at least one example of a ship but is classified as "no ship." We do not analyze true negative images as there are large expanses of empty ocean collected by remote sensing platforms and we are only concerned with the ability to correctly classify those images that contain an object of interest, in this case a ship. We use these TP, FP, and FN values to calculate recall, precision, FPR, and F-

score. We then produce ROC curves from the calculated recall and FPR. These results are discussed in Chapter V.

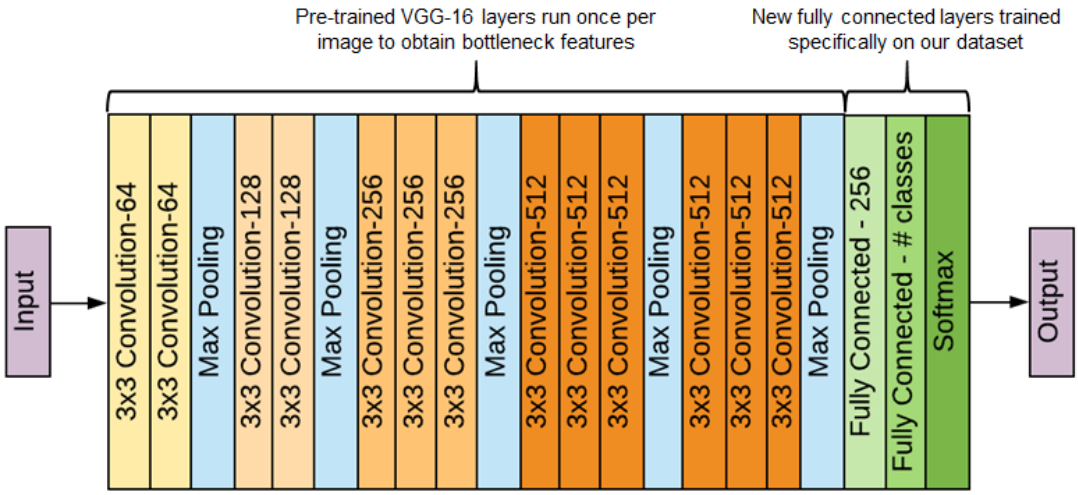


Figure 8. Diagram of VGG-16 model adapted for transfer learning. Adapted from [6], [42].

2. Experiment 2: Impact of Training Dataset Size on Model Performance

As discussed in Chapter II, the process of transfer learning provides a method to train a high-performance CNN without the need for a large dataset of labeled images. In this section we examine the performance of the previously described VGG-16 transfer learning process when trained on datasets of decreasing size. To accomplish this, subsets of ship and non-ship images are produced by starting with the 2000 image dataset used in Experiment 1 and randomly selecting 200 images for removal each time a new subset is constructed. This process produces a series of subsets ranging in size from 2000 to 200 and decreasing by steps of 200. Prior to re-training the VGG-16 network, these datasets are further subdivided such that 10% of the training images are randomly selected for use in model validation during training. The test set is the same as that used in Experiment 1 and remains constant across each newly trained model. Each model is trained using the same parameters as in previous tasks and we evaluate them with the same performance metrics

of recall, precision, FPR, and F-score. We repeat this process three times to control for potential impact of variations in the randomly selected training sets of reduced size. We produce ROC graphs from the average recall and FPR.

3. Experiment 3: Impact of Noise and Blur on Model Performance

Rainey et al. demonstrate that the effect of image degradation due to noise and resolution differences on SVM performance can be mitigated by including images of these types in their training set [33]. We aim to build upon this work by comparing retrained VGG-16 performance on datasets where a percentage of images contain salt and pepper noise. We test the effect of replacing 25%, 50%, or 75% of either the training set, test set, or both with images containing salt and pepper noise. Images are randomly selected from the ship classification dataset in Table 3 and noise is added by selecting random pixels to be set to either black or white values.

We then repeat this process with the addition of a Gaussian blur to 25%, 50%, or 75% of either the training set, test set or both. This is to simulate the presence of images of lower resolution in the dataset which may occur if imagery analysts are working with data coming from different overhead sensor platforms. We train each model using the same parameters as in previous tasks and we evaluate them using the same performance metrics of recall, precision, FPR, and F-score. We produce ROC curves from the calculated recall and FPR.

4. Experiment 4: Model Performance when Trained on Sub-categories of Ship and Non-ship Images

In our fourth experiment, we explore whether the VGG-16 architecture can be retrained to classify more specific categories than “ship” or “no ship.” For this task, we create a subset of the ship classification dataset in Table 3 by randomly sampling 220 images from each sub-category. Twenty images are randomly selected from each category to serve as test images and an additional 20 images are randomly split from the training set to be used for validation during the training process. This new data set is summarized in Table 4. The VGG-16 transfer learning architecture and parameters are the same as in previous experiments. Performance is measured by counting the number of correctly and

incorrectly classified images in each category. This data is then used to create a multi-class confusion matrix and calculate recall, precision, FPR, and F-score as done previously in Experiments 1-3.

Table 4. Dataset breakdown for ship category classification model

	Ship Classification
Ship	**660**
Barge	220
Fast	220
Merchant	220
Non-Ship	**880**
Cloud	220
Island	220
Land	220
Ocean	220

D. OBJECT DETECTION TASKS

In our second set of experiments, we aim to expand upon previous experiments by moving from image classification to object detection. We retrain an SSD model that is built upon the VGG-16 architecture and previously trained on the ILSVRC2014 object detection dataset. As discussed in Chapter III, this method has previously been used by Nie et al. for detection of ships in satellite imagery [41]. We replicate these results in Experiment 5 using our own dataset and then build upon them in Experiments 6 and 7 by testing the effect of noise and blur additions to our dataset and by examining model performance on subcategories of ship images.

In Experiment 5, we verify whether our object detection model is able to correctly detect and classify ships in satellite imagery. To conduct our training, we use a Keras adaptation of the SSD model described in Chapter II [44]. We modify this model for the number of ship-related categories (one or three) depending on the current task. The model is then trained on the ship detection dataset described in Table 3, after we subdivide it by randomly selecting 10% of the images to be used in validation, and 10% to be used in testing. We use the following parameters in training:

- Batch size = 8
- Number of Epochs = 50
- Step per Epoch = 100
- Activation = relu, softmax (final layer)
- Optimizer = adam
- Learning Rate = 0.001

We then examine the effect of noise and blur on SSD model performance in Experiment 6 using the same procedure as outlined above for adding salt and pepper noise and Gaussian blur 75% of the training set, test set, or both. We elect to manipulate 75% of the images as that is the percentage where we see the greatest impact on ship classification performance. Finally, in Experiment 7, we test SSD model performance on the task of identifying specific subcategories of ships. Model performance is once again evaluated by counting TP, FP, and FN. This information is then used to calculate recall, precision, FPR, and F-score for single category tasks and to construct a multi-class confusion matrix for the task of detecting ship subcategories. For this set of experiments, a detection is considered a TP if the bounding box contains any portion of the ship and correctly identifies the ship. This is because the premise of our thesis is to develop a model that can be used to help analysts quickly find ships in satellite imagery, so the bounding boxes do not need to be 100% accurate as long as they correctly flag the object. A detection is considered a FP if a bounding box is drawn around a region that does not contain a ship. Lastly, a FN is defined as a region containing a ship that is not detected by the model. We do not count TN objects as that category would include all background objects found in each image.

V. RESULTS AND ANALYSIS

This chapter discusses the results obtained from the experiments outlined in Chapter IV. In our first set of experiments, we find that a retrained VGG-16 network provides an accurate model for the task of object classification even when trained on datasets with as few as 200 samples. In our second set of experiments, we find that a retrained SSD model built upon the VGG-16 architecture can be used for the object detection task as well. As part of our analysis, we provide ROC curves and F-scores for single category classification tasks and confusion matrices for multi-category classification and tasks. For each experiment, we also examine which images the model failed to properly classify or detect in an effort to better understand what characterizes hard cases. Lastly, we compare the training time required for models trained on a CPU to time required when training is conducted on a GPU.

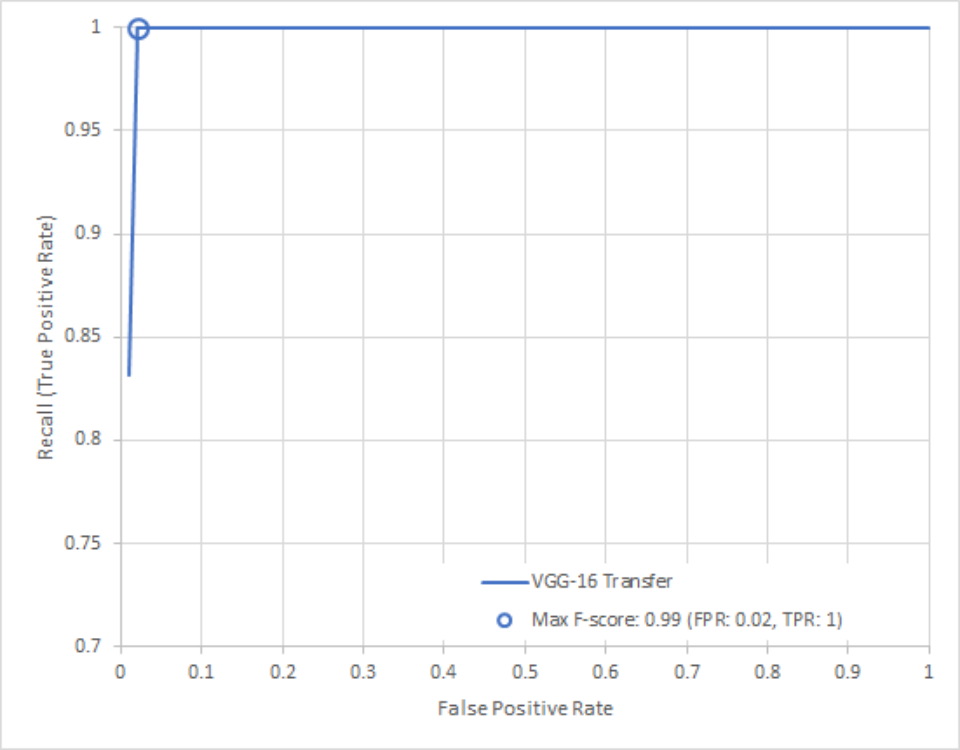
A. OBJECT CLASSIFICATION TASKS

This section discusses the results we obtained when testing our model on a variety of tasks related to object classification. We first evaluate the performance of a retrained VGG-16 model to establish a baseline for ship classification tasks. Following this, we examine the performance of our retrained VGG-16 network when it is trained on datasets of decreasing size. Next, we examine the effect that the addition of images containing noise or blur has on retrained VGG-16 model performance. Finally, we show that our retrained VGG-16 model can be trained to distinguish between sub-categories of ships as well as the overall class.

1. Experiment 1: Ship Classification Baseline

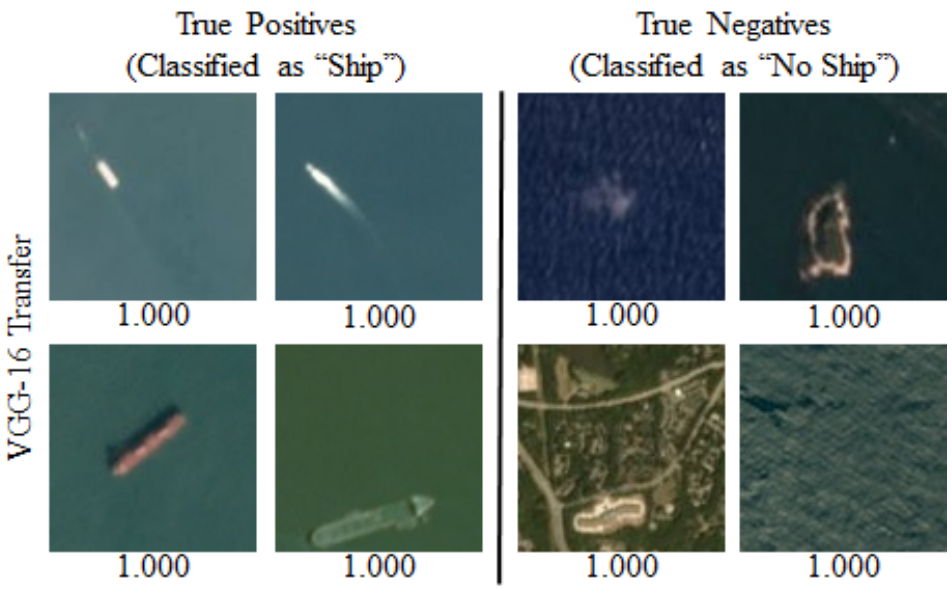
The retrained VGG-16 network demonstrates high performance resulting in only two false-positives and no-false negatives at the highest performing threshold. Figure 9 depicts the ROC curve for the models when presented with a test set of 200 images containing examples of both the “ship” and “no-ship” categories. Figures 10 and 11 depict examples of images that were correctly and incorrectly classified by our model, respectively. Both false positive images shown in Figure 11 contain rectangular shapes of

a similar size to many of the ships in our dataset. It is these regions in each image that likely caused our model to misclassify them as ships. In contrast to this, the barge that is misclassified as “No Ship” is slightly blurry and has similar characteristics to the small cloud shown as a true negative example in Figure 10.



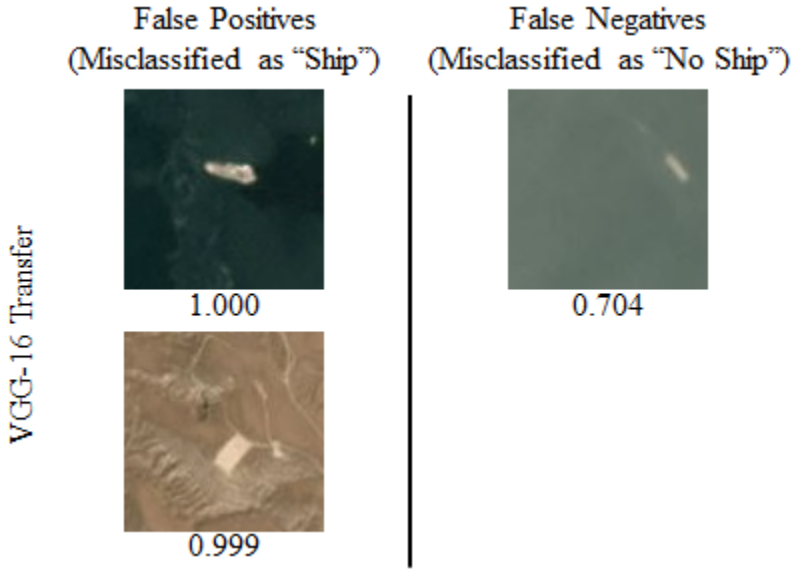
The model is evaluated on a test set containing 200 images from both the “ship” and “no ship” categories after 50 epochs of training. The retrained VGG-16 demonstrated high performance with a maximum F-score of 0.99.

Figure 9. Ship classification performance for the VGG-16 model retrained on our dataset using transfer learning



The number displayed below each image represents the predicted probability that the image is in the correct category.

Figure 10. Examples of images correctly classified by the VGG-16 model retrained using transfer learning.

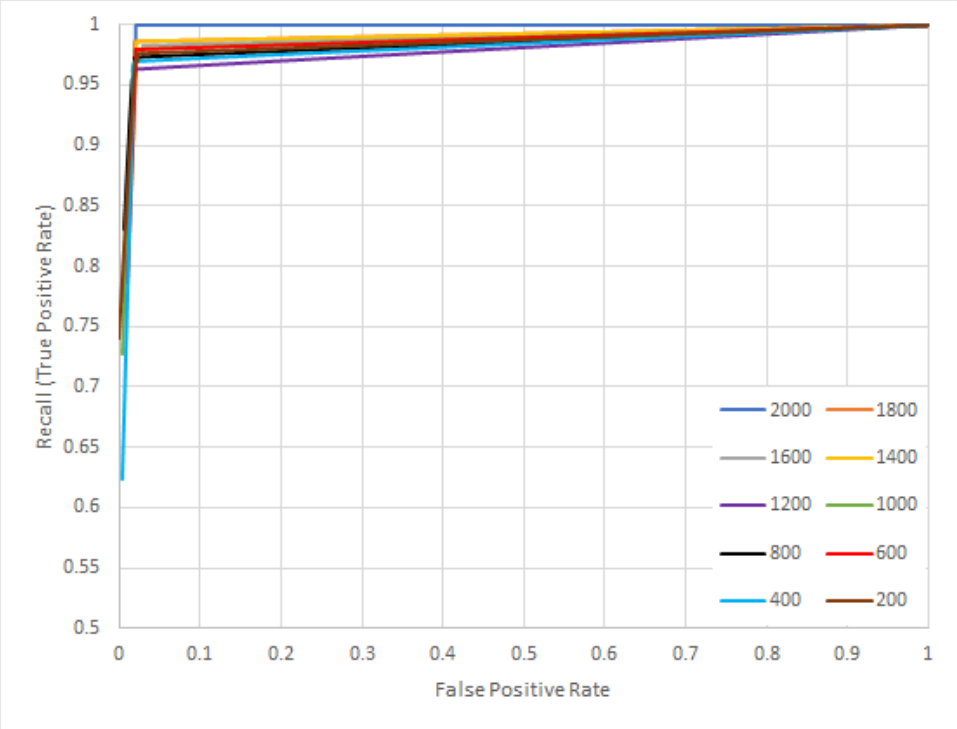


The number displayed below each image represents the predicted probability that resulted in the image being misclassified.

Figure 11. Examples of images misclassified by the simple CNN model and the VGG-16 retrained using transfer learning.

2. Experiment 2: Impact of Training Dataset Size on Model Performance

Figure 12 displays the ROC curves for each of the models and Table 5 contains the corresponding performance metrics at the best F-score threshold for each model. Although there is slight variation in the maximum F-scores for each model, all models perform well, with the worst performer having an F-score of 0.973. This indicates that VGG-16 can be retrained to fit new DoD related imagery classification tasks even in cases where a large amount of previously obtained satellite imagery is not available for use in model training.



All models are evaluated on the same test set containing 200 images from both the "ship" and "no ship" categories. Performance metrics at the threshold producing the highest F-score are displayed in Table 7.

Figure 12. Ship classification performance for the VGG-16 model retrained on datasets of decreasing size using transfer learning.

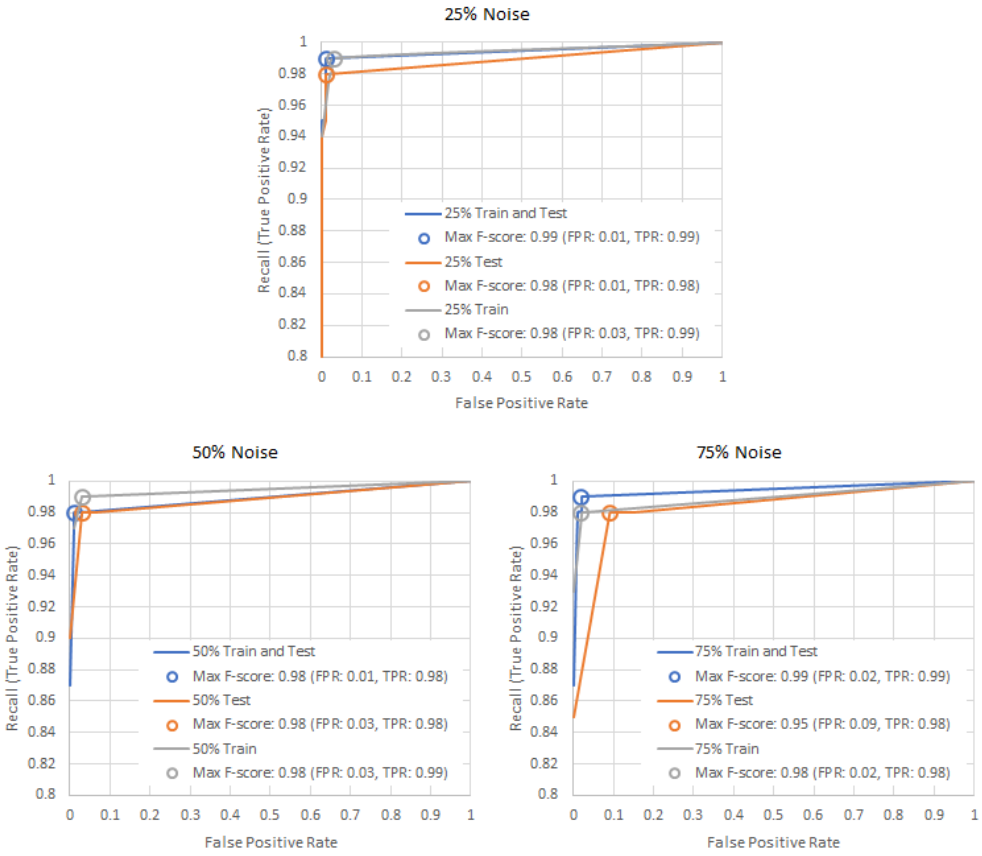
Table 5. Performance metrics for VGG-16 models trained on datasets of decreasing size using transfer learning.

Training Set Size	Recall (TPR)	Precision	F-score	FPR
2000	0.990	0.980	0.985	0.020
1800	0.987	0.980	0.983	0.020
1600	0.983	0.971	0.977	0.030
1400	0.987	0.980	0.983	0.020
1200	0.963	0.983	0.973	0.017
1000	0.973	0.980	0.977	0.020
800	0.973	0.977	0.975	0.020
600	0.980	0.980	0.980	0.020
400	0.967	0.983	0.975	0.017
200	0.977	0.977	0.977	0.023

These metrics represent performance at the threshold producing the highest F-score. All subsets of data show high performance with a minimum F-score of 0.973 indicating that new models can be trained even in the absence of large amounts of training data.

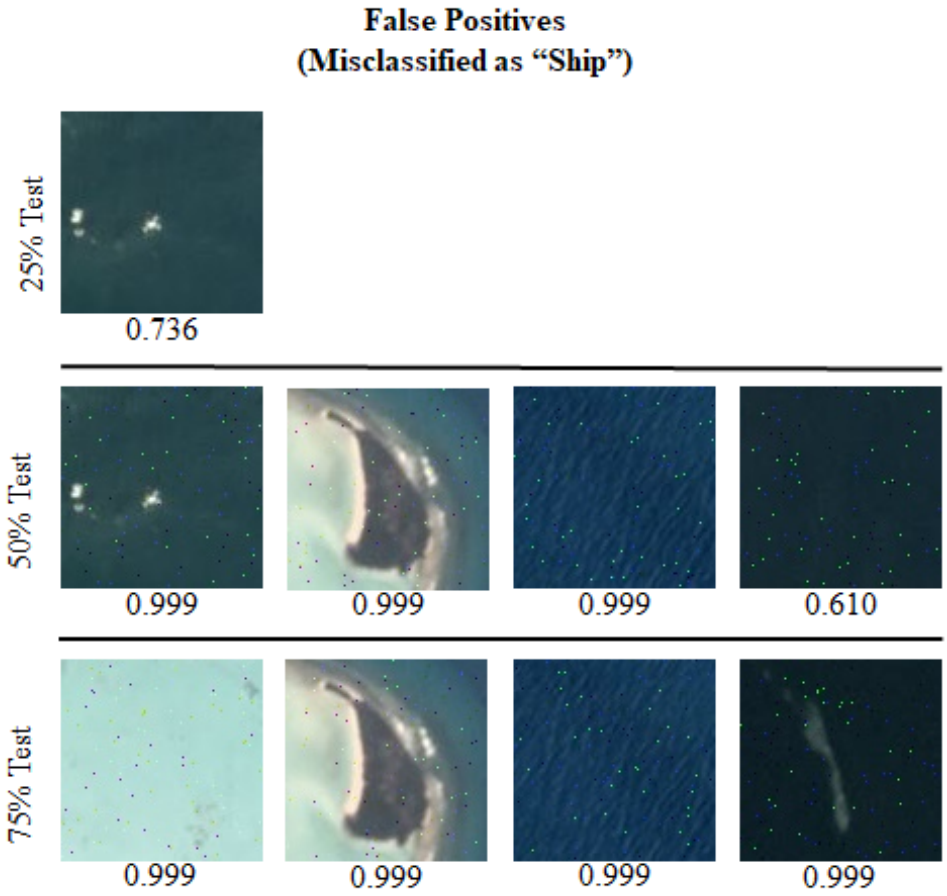
3. Experiment 3: Impact of Noise and Blur on Model Performance

Figure 13 displays ROC curves for models trained and tested on datasets containing noise in either 25%, 50%, or 75% of their training data, test data, or both data sets. We find that models trained on datasets devoid of noise examples produce a higher rate of false positives when presented with a test set containing noise. Figure 14 displays examples of images that are misclassified with the addition of noise. Note that the image misclassified in the 25% Test set did not contain noise. Therefore, the addition of noise to 25% of the test images had no measurable impact on our results. However, when noise is added to the image, as can be seen in the 50% Test experiment, the image is misclassified with a higher predicted probability.



In all three cases, models that are only exposed to noise during testing demonstrated the worst performance.

Figure 13. Ship classification performance for the VGG-16 model when presented with training and test data with 25%, 50%, or 75% of images containing salt and pepper noise.

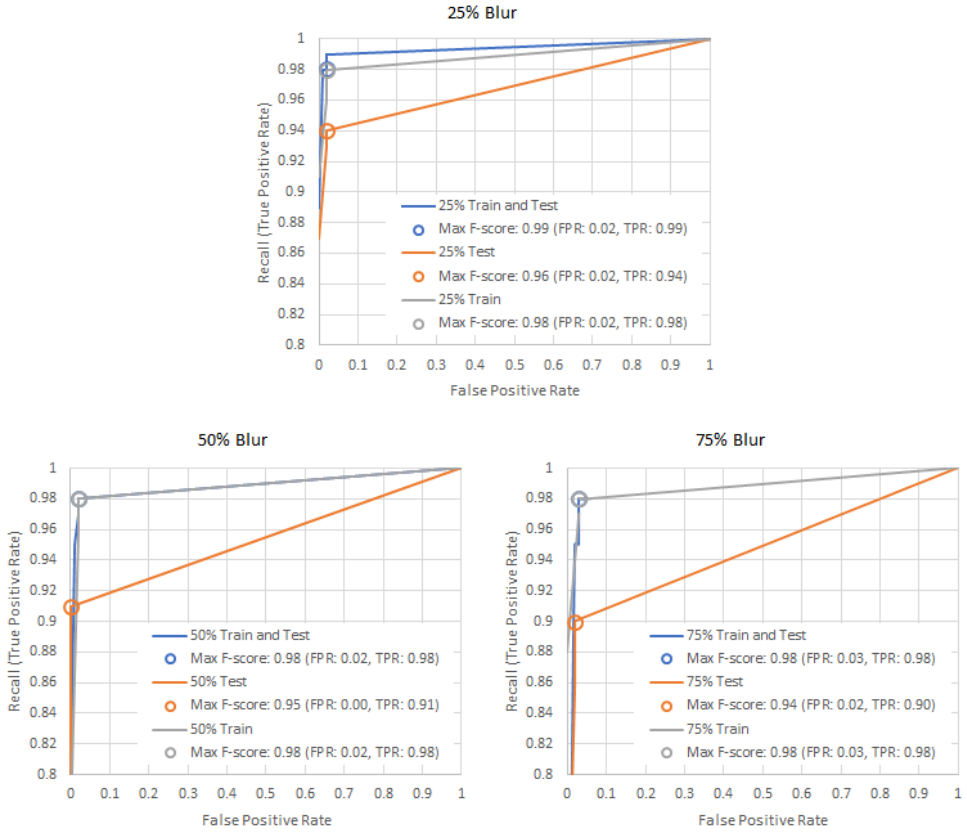


Images selected for inclusion are those with the highest probability score in the incorrect category. This probability score is displayed below each image. The first image shown in the 25% and 50% Test categories is also misclassified in the 75% Test category, however, it does not score high enough for inclusion in the figure.

Figure 14. Examples of images misclassified as ships when noise is added to the test set only.

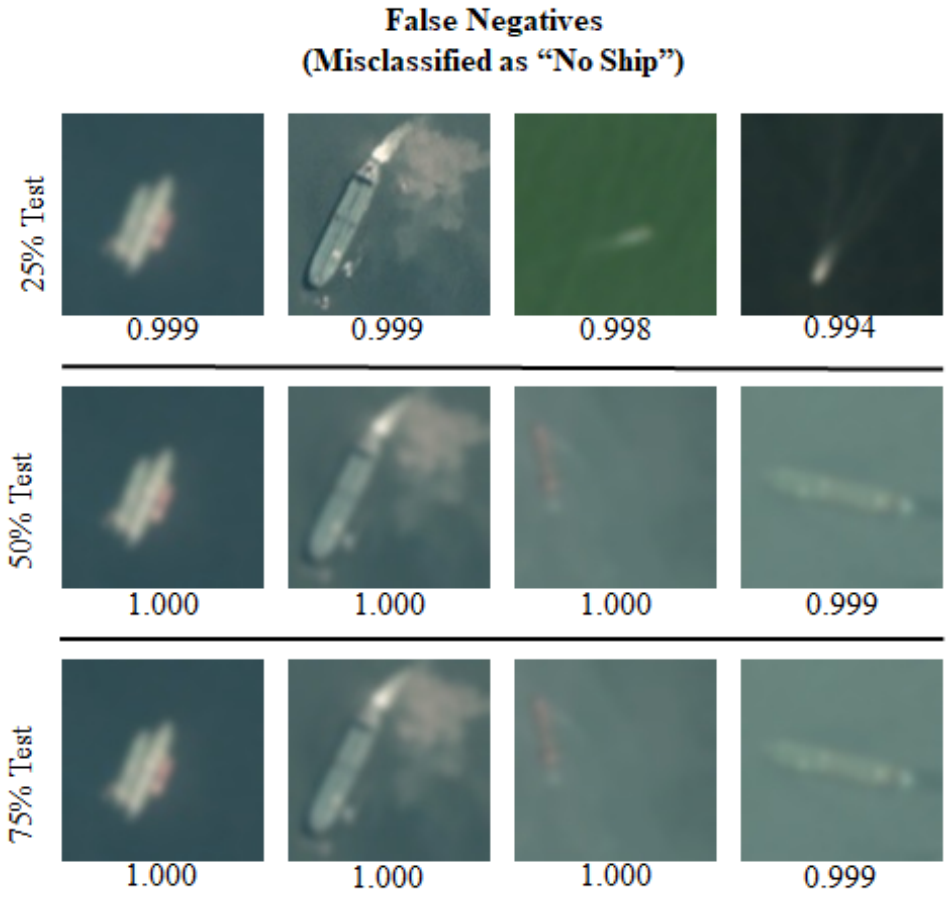
Figure 15 displays ROC curves examining model performance when blur is added to 25%, 50%, or 75% of the training set, test set, or both image datasets. We find that models which had no blurred images in training perform worse when tested on datasets that include blurred images than those trained on both blurred and not blurred images. This decrease in performance is the result of a higher number of false negative classifications. Examples of the highest scoring false-negatives for the models exposed to blurred images during testing only are shown in Figure 16. There is not a noticeable impact due to the presence of blurred images in the creation of false-positive images. This result indicates

that a dataset trained on higher resolution images will likely perform poorly when attempting to classify images obtained from a lower resolution sensor.



In all three cases, models that are only exposed to blurred images during testing are the worst performers.

Figure 15. Ship classification performance for the VGG-16 model when presented with training and test data in which 25%, 50%, or 75% of images are blurred.



Images selected for inclusion are those with the highest probability score in the incorrect category. This probability score is displayed below each image.

Figure 16. Examples of images misclassified as ships when blurred images are added to the test set only.

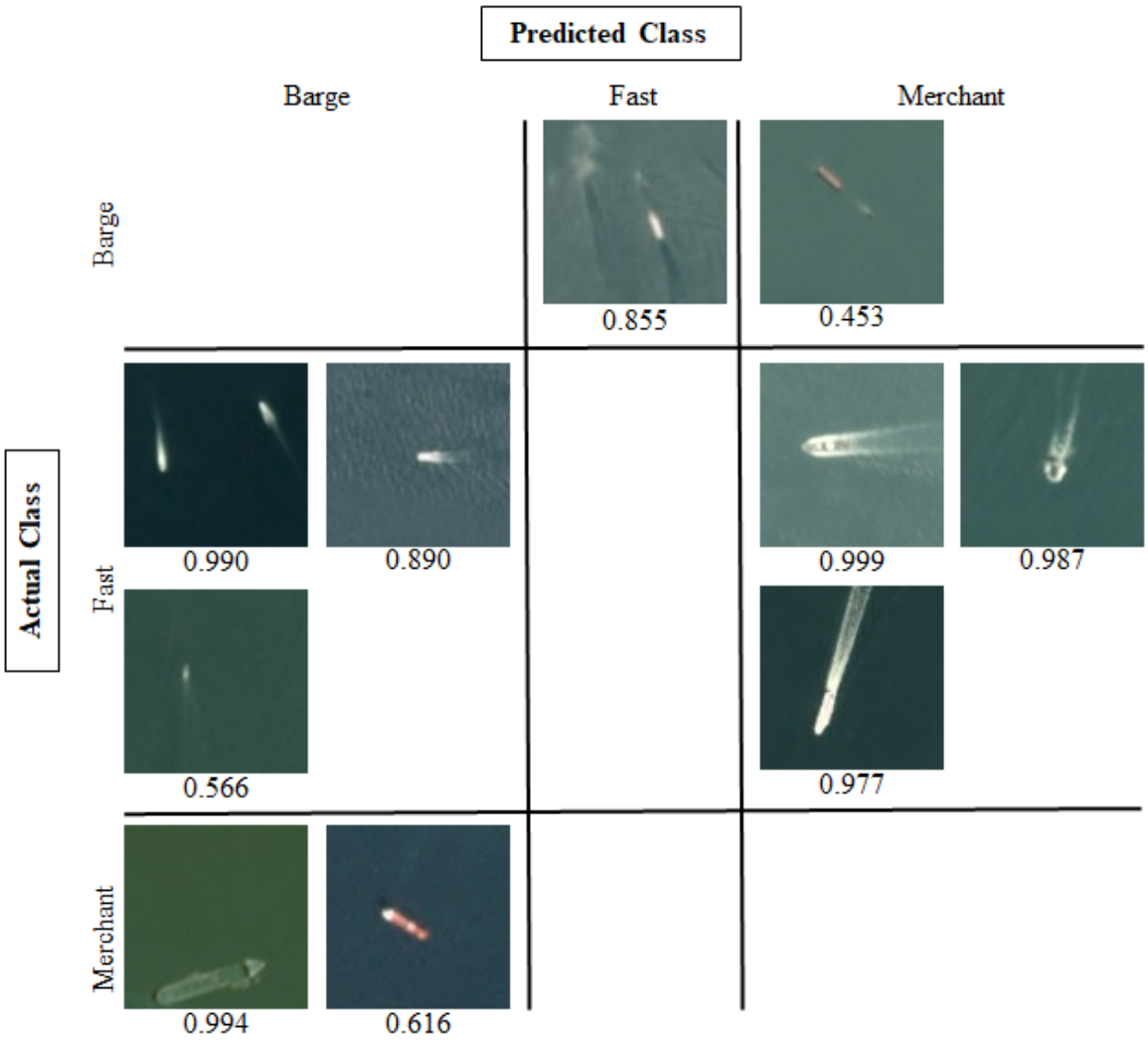
We find that the presence of noise and blur in the test data only produce opposite effects. The presence of noise causes an increase in the number of FP image classifications, while the presence of blur causes an increase in the number of FN classifications. We hypothesize that this occurs because the model is learning to associate sharp gradients in an image with the presence of a ship. Thus, the presence of additional sharp gradients in the form of salt and pepper noise causes an increase in FPs and the reduction of sharp gradients created by the application of Gaussian blur causes an increase in FNs. Additionally, we demonstrate that these negative effects can be mitigated by the presence of images with noise and blur in the training data.

4. Experiment 4: Model performance when trained on sub-categories of ship and non-ship images

Table 6 represents a confusion matrix showing predicted versus actual classes for each image sub-category. Figure 17 displays the ship images that are misclassified and the probability score in their predicted category. Based on the low probability scores for some of the misclassified images we also examine the model's top-2 performance on the ship subcategory classification task. This means that we count an image as properly classified if its true category is one of the top two predicted categories. We adapt this method from the top-5 metric for measuring model performance that is used to evaluate ILSVRC entries [28]. We elect to examine only the top two scores instead of the top five due to the small number of categories in our dataset. Top-2 performance results are displayed in Table 7.

Table 6. Confusion matrix representing top-1 performance on the ship subcategory classification task.

		Predicted Class - Top 1							
		Ship			No Ship				
		Barge	Fast Ship	Merchant	Cloud	Island	Land	Water	Recall (TPR)
Actual Class: Ship	Barge	18	1	1					0.90
	Fast ship	3	14	3					0.70
	Merchant	2		18					0.90
Actual Class: No Ship	Cloud				20				1.00
	Island				1	18	1		0.90
	Land					2	17	1	0.85
	Water					1		19	0.95
	Precision	0.78	0.93	0.85	0.95	0.86	0.94	0.95	



The number located below each image represents the probability score that it belongs in the indicated category.

Figure 17. Examples of images misclassified during the ship category classification task.

Table 7. Confusion matrix representing top-2 performance on the ship subcategory classification task.

		Predicted Class - Top 2							
		Ship			No Ship				
		Barge	Fast Ship	Merchant	Cloud	Island	Land	Water	Recall (TPR)
Actual Class	Ship: Barge	19		1					0.95
	Ship: Fast ship		19	1					0.95
	Ship: Merchant	1		19					0.95
	No Ship: Cloud				20				1.00
	No Ship: Island					20			1.00
	No Ship: Land					1	19		0.95
	No Ship: Water					1		19	0.95
	Precision	0.95	1.00	0.90	1.00	0.91	1.00	1.00	

We find that while the model mis-categorizes several images in each subcategory, it is still able to correctly identify between images containing ships and those containing non-ship objects. The model has the most difficulty recognizing the category we characterize as “fast” ships. We hypothesize that this may be due to their small size and the presence of wakes in some “barge” and “merchant” images that may be confused with the “fast” ship characteristic. Additionally, barges are composed of two pieces (the tug boat and the barge it is towing) which may be confused with the presence of multiple fast ships in an image.

B. OBJECT DETECTION TASKS

This section examines the performance of the SSD300 object detection model on the task of ship detection. We begin by confirming that our model is able to correctly detect ships present in satellite imagery. We then explore the effect that the addition of noise and blur has on our retrained SSD300 object detection model. Finally, we show that our model is able to distinguish between subcategories of ships during the detection process, similar to what we observe during testing of the object classification tasks.

1. Experiment 5: Object Detection Baseline

Here, we show that an SSD300 object detection architecture pretrained on the ImageNet dataset can be retrained to detect ships present in satellite imagery. Although,

performance was lower than with the easier ship classification task, our model achieves a maximum F-score of 0.91, outperforming the maximum F-score of 0.84 reported by Nie et al. [41]. We note that although this metric is encouraging, it may be influenced by variations in our datasets. A fair comparison would require that we test against their dataset, a task we leave for future work. A ROC curve of our results is displayed in Figure 18 and examples of correctly classified images are presented in Figure 19. Examples of both misclassified and un-detected images are presented in Figure 20. Four objects are misclassified. However, only two of them receive a probability score above 0.5. These objects were sandy areas of larger islands which were incorrectly classified as ships. Small islands are also included in the training set and are correctly not detected as ships. The majority of the undetected ships are fast ships. Barges proved second most difficult to detect and only a few small merchants were undetected. Additionally, many of the fast ships were only detected at thresholds below 0.5 preventing us from raising the detection threshold to reduce the number of false positives.

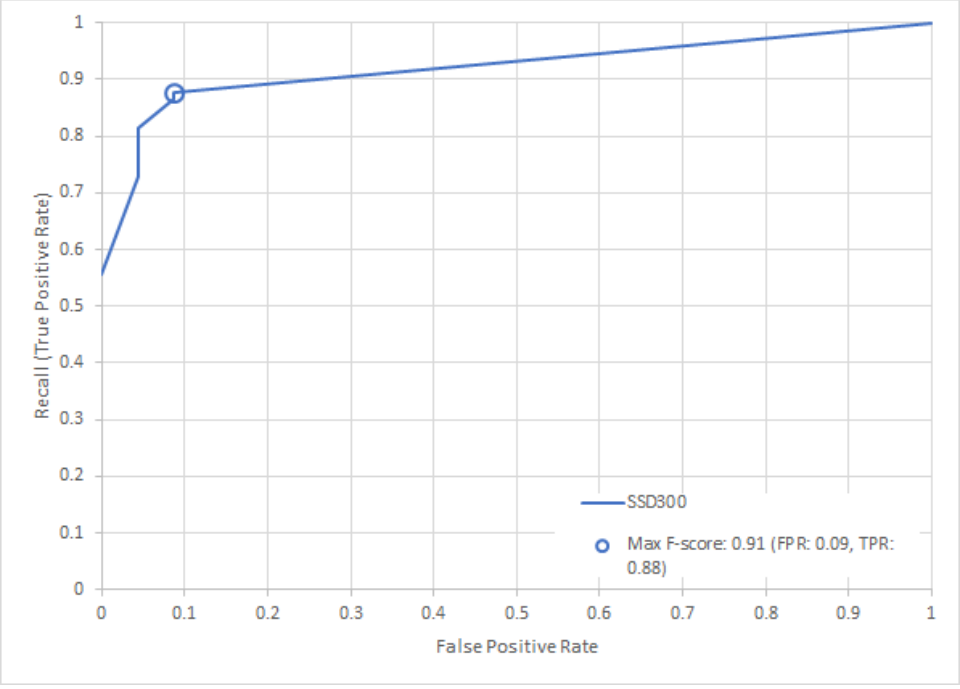
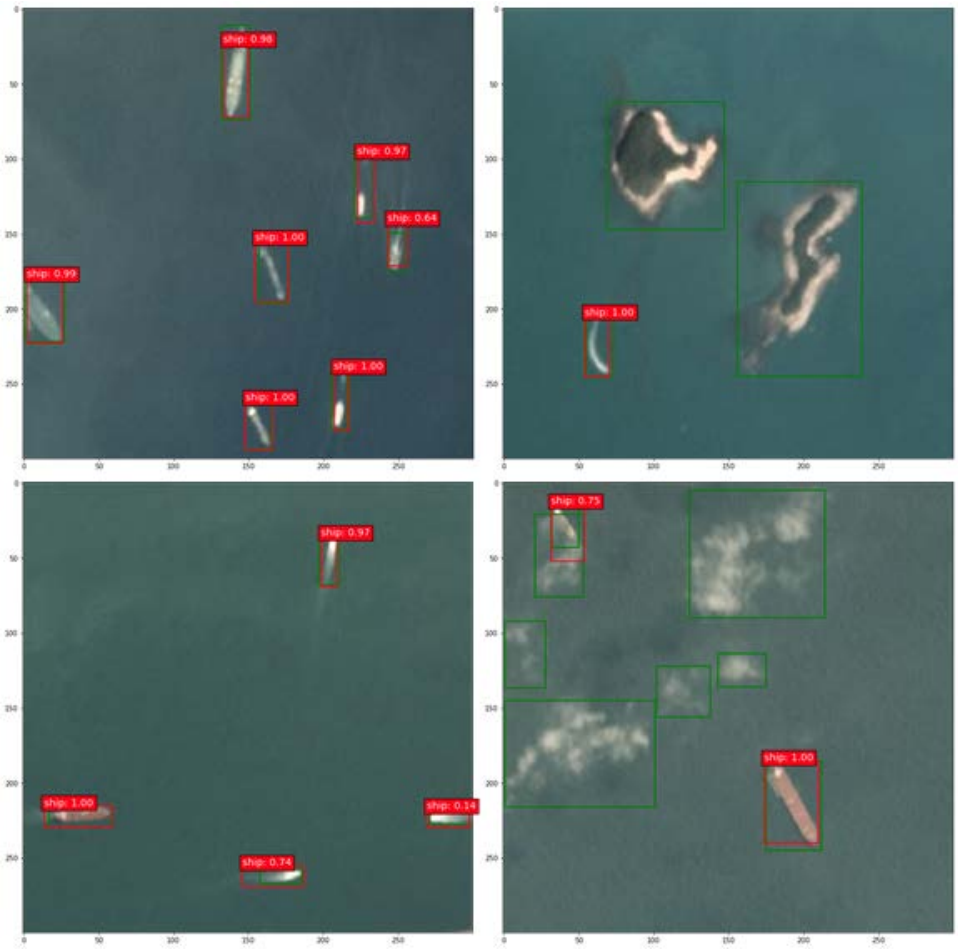
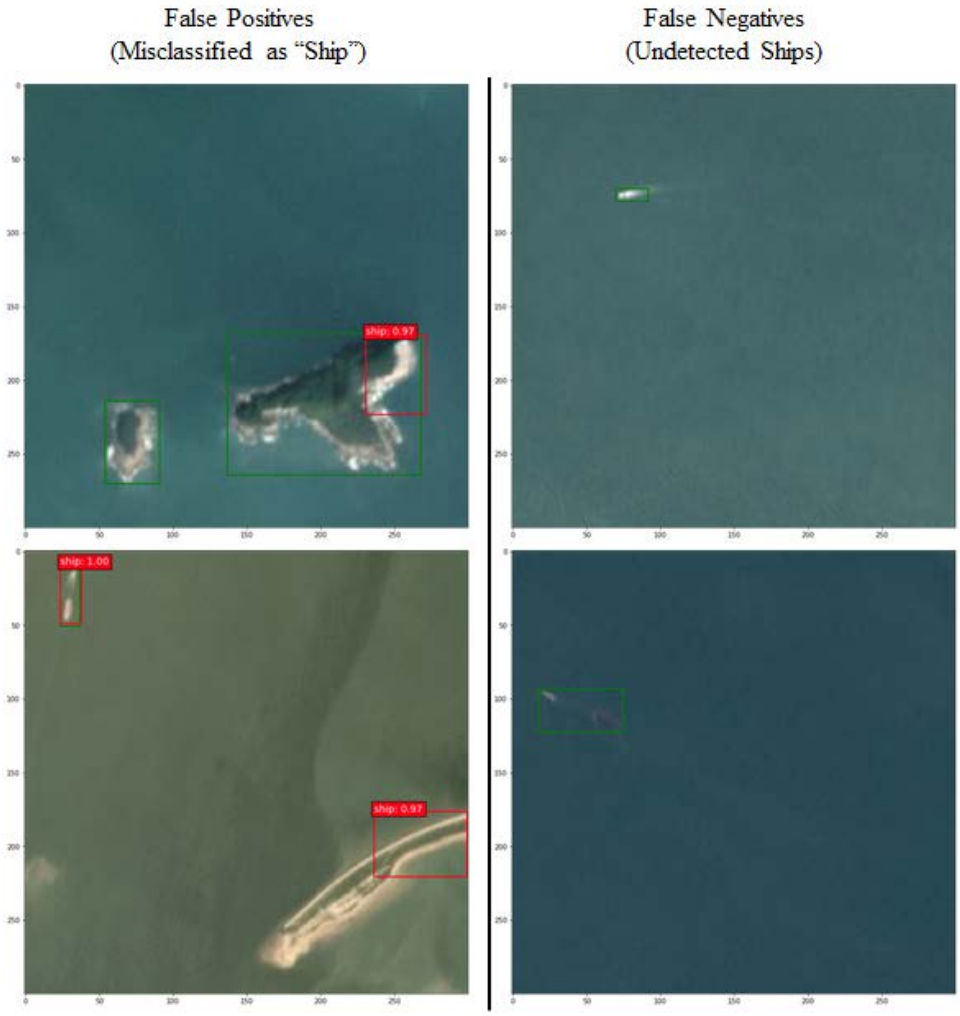


Figure 18. Performance of the SSD300 object detection architecture retrained on our dataset using transfer learning.



Green bounding boxes were hand drawn around both ship and non-ship images during construction of the dataset. Red bounding boxes indicate the predictions made by the classifier.

Figure 19. Examples of images that are correctly detected and classified by the retrained SSD300 object detection model.



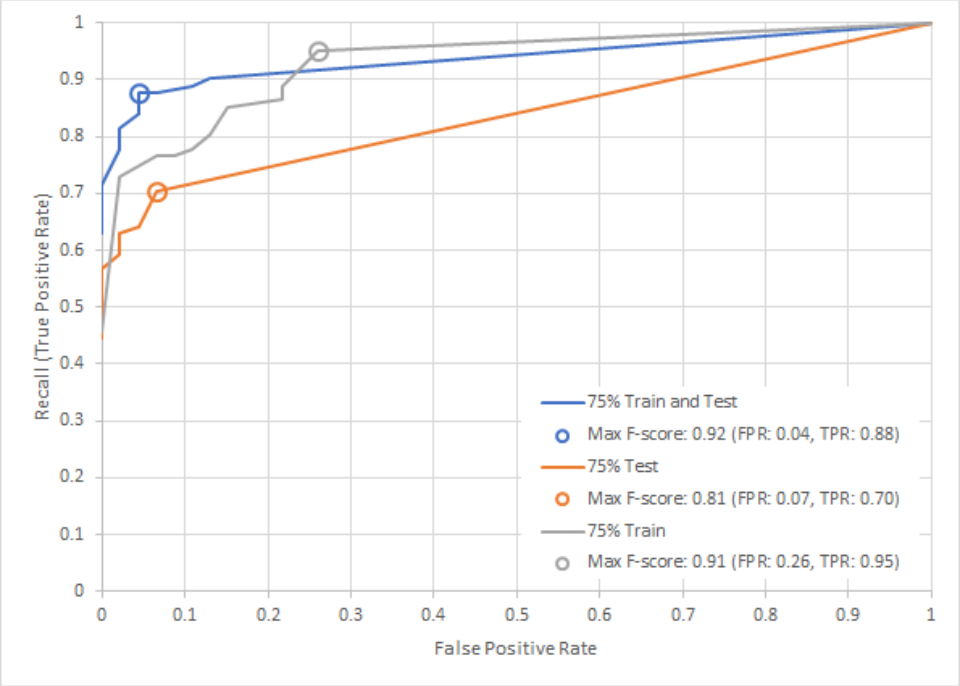
Green bounding boxes were hand drawn around both ship and non-ship images during construction of the dataset. Red bounding boxes indicate the predictions made by the classifier.

Figure 20. Examples of images that are either misclassified or undetected during the ship detection task.

2. Experiment 6: Impact of Noise and Blur on Model Performance

We find that both types of image degradation have a different effect on model performance than what we find in the ship classification task discussed above. Figure 21 displays the ROC curve for the ship detection task when noise is added to 75% of test images, training images, or both data sets. The addition of noise to the test set only again results in the worst model performance, similar to the results previously discussed for ship classification. However, while for ship classification the poor performance is due to an

increased number of false positives, for ship detection the poor performance is due to a larger number of false negatives. At the highest performing threshold there are 24 undetected ships compared to only 10 in the original ship detection task. Additionally, the model presented noise only during training exhibited poorer performance but in the form of additional false-positives. There is an increase in misclassified objects from four in the original ship detection task to 12 following training on a dataset containing noise.

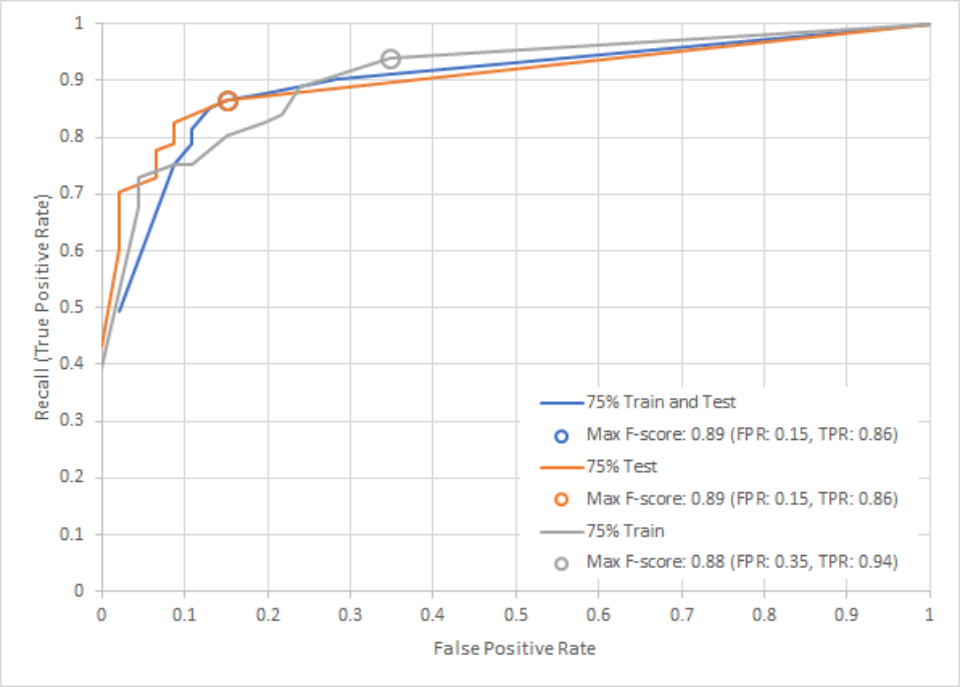


Addition of noise to the test set only results in a higher number of undetected images when compared to models trained and tested with no noise present.

Figure 21. Performance of the SSD300 object detection architecture retrained on our dataset using transfer learning when noise is added to 75% of either the training set, test set, or both.

Figure 22 displays the ROC curve for the ship detection task when 75% of test images, training images, or images in both data sets are blurred using a gaussian blur. We find that in all three scenarios the presence of blur results in a higher number of false positive ship detections when compared to our original model with no blurry images. Unlike with the ship classification task, the presence of blur in the test set only does not

result in a large amount of undetected ships. In the ship classification task, we see a drop in recall from 1.0 to 0.90 when blur is added to the test set only. In this task, we find recall only drops from 0.88 to 0.86 with the same addition of blur to the test set.



Addition of blurred images results in a higher false positive rate in all three models with the worst performer being the model with blurred images added only to the training set.

Figure 22. Performance of the SSD300 object detection architecture retrained on our dataset using transfer learning when 75% of either the training set, test set, or both is blurred.

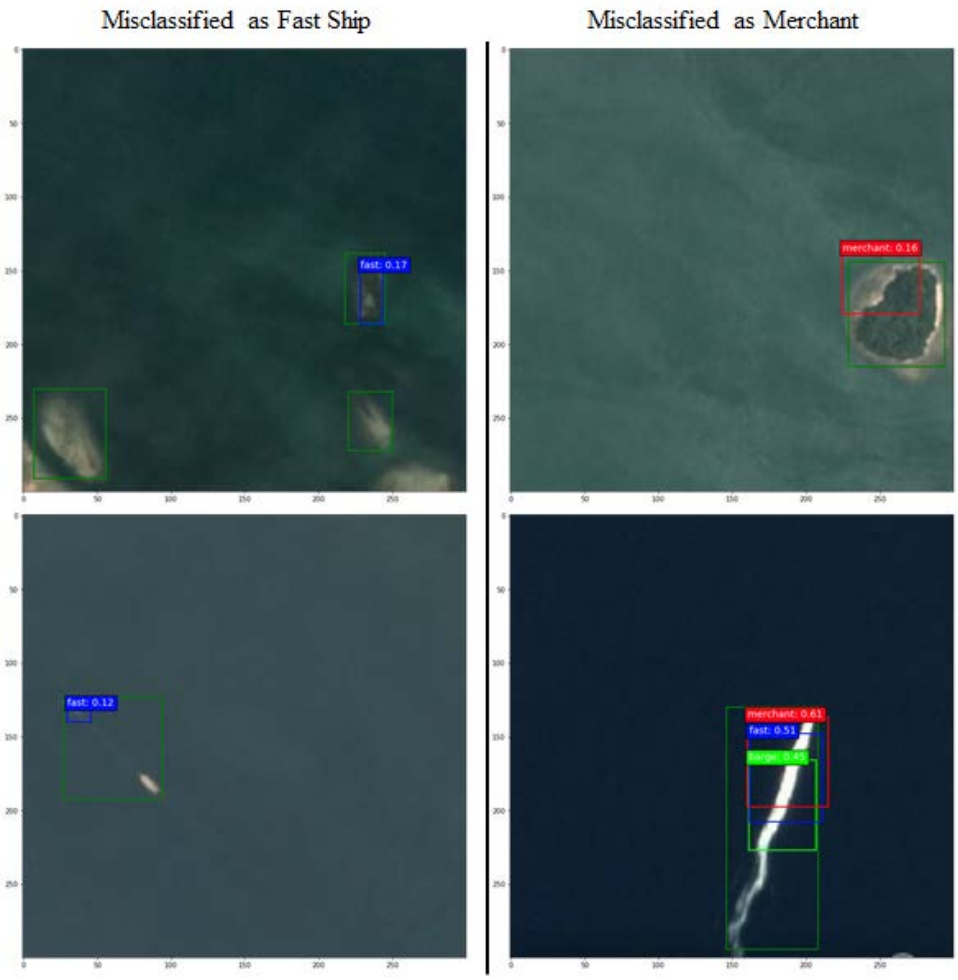
3. Experiment 7: Model Performance when Trained to Detect Sub-categories of Ship Images

We find that our model is able to correctly detect specific ship types with an average recall of 0.82 and average precision of 0.86 (see Table 8 for the full confusion matrix). The model performs the worst on the barge subcategory, possibly because barges appear in imagery as two small ships traveling in a straight line. Those examples where the tug and barge are close together are detected at a higher rate than those with a greater distance between the two components of the barge. In this second case, the barges are either undetected or a bounding box is drawn around the tug only and it is labeled as a fast ship.

Two fast ships are also misclassified as merchants. Both cases are fast ships with a very long wake where the model classifies different parts of the wake as different ship types. In both cases, merchant is the highest scoring category; however, a fast ship bounding box is also drawn with a lower probability score. No objects are misclassified as a barge. Additionally, of the eight false positives detected only one has a probability score over 0.5. Subsections of larger islands are misclassified as merchants, while several small islands and clouds are misclassified as fast ships. Examples of the misclassified images discussed above are shown in Figure 23.

Table 8. Confusion matrix representing SSD300 model performance on the ship subcategory detection task.

		Predicted Class				
		Barge	**Fast Ship**	**Merchant**	**Not Detected**	**Recall (TPR)**
Actual Class	**Barge**	15	3		2	0.75
	Fast ship		19	2	2	0.83
	Merchant			34	4	0.89
	Non-Ship		3	5	38	0.83
	Precision	1.00	0.76	0.83	0.83	



Dark green bounding boxes were hand drawn around both ship and non-ship classes during construction of the dataset. Red, blue, and bright green bounding boxes indicate the classifier presents the presence of a merchant, a fast ship, and a barge respectively.

Figure 23. Examples of images that are misclassified as fast ships and merchants.

C. COMPARISON OF TRAINING TIME BETWEEN CPU AND GPU

Table 9 provides a comparison of training times for each model type trained on both the CPU and GPU described in Chapter IV. Although, the GPU provides faster training, the CPU is still able to complete training on a set of 2000 images in under four minutes for object classification tasks. CPU use for the task of object detection appears to be less viable, requiring a training time of over 36 hours. This is significant because it demonstrates that transfer learning may be an effective machine learning technique for

some tasks even in an operational environment where computational resources may be limited.

Table 9. Comparison of training times in seconds for our object classification and object detection models.

	GPU	**CPU**
Object Classification Model	44.208	213.804
Object Detection Model	662.95	36+ hrs

VI. CONCLUSION AND FUTURE WORK

We now discuss the answers to the research questions addressed by this thesis as demonstrated by the results presented in Chapter V. We begin by discussing the benefits and risks of using transfer learning to re-train CNNs to classify objects of interest to the DoD in satellite imagery. We then discuss the benefits and risks of applying similar techniques to the related task of object detection on a dataset obtained from the same sensor. Lastly, we discuss future work that could further advance research in this application of machine learning.

A. BENEFITS AND RISKS OF USING RETRAINED CNNS FOR OBJECT CLASSIFICATION IN SATELLITE IMAGERY

Our results demonstrate that a CNN classifier initially trained on a large dataset of photographic images can be successfully retrained to classify objects in satellite imagery. At the highest performing classification threshold our model performed with an F-score of 0.99, producing no false-negative classifications and only two false-positive classifications. This high level of performance was obtained using a significantly smaller training set than that which the network was initially trained on: our model was trained on only 2000 images, whereas the original VGG-16 network was trained on a set of over 1.2 million images [28], [6]. We also show that our retrained model performs well when trained on a dataset as small as 200 images; producing a maximum F-score of 0.97, only 0.02 lower than the model trained on 2000 images. The ability to train on such a comparatively small dataset and still produce high levels of accuracy is of great benefit to the DoD. Using this method of training CNNs to correctly classify novel object categories of interest to the DoD will greatly reduce the overhead required for development of new models in terms of the training images needed and the time required to manually label new training datasets. Additionally, we show that the process of transfer learning is not very resource intensive. We find that retraining the VGG-16 network on 2000 images takes approximately three minutes longer when training is conducted using a CPU compared with a GPU. This demonstrates that although training with a GPU is still faster, it is not required to produce new CNN models in a timely manner if the technique of transfer learning is applied.

There is still some risk associated with using this type of CNN for object classification, however. Our results show that the retrained VGG-16 model is sensitive to the presence of images containing noise and blurred images in the test set. Noise is added to simulate degraded satellite imagery, while blur is used to simulate images of a lower resolution. In both cases, we see a reduction in model accuracy when these effects are added to the test dataset only. In the case of noise, we see a drop in maximum F-score from 0.99 to 0.95 when noise is added to 75% of the test set; and in the case of blur we see a drop in maximum F-score from 0.99 to 0.94. We find that this reduction in performance can be mitigated by including examples of noise and blur in the training set; producing overall accuracies of 0.99 and 0.98 for noise and blur, respectively.

These results demonstrate the importance of ensuring that the training is representative of the images you later hope to classify. It is often tempting to include only perfect examples of images when building a dataset; however, this is not useful if there are defects present in the real-world images the model will later encounter. We show that the negative effects of testing our model on images containing both noise and blur can be mitigated through the inclusion of similarly degraded images in our training set. Additionally, the effect produced by the addition of blurred images demonstrates that similar images may be misclassified if they are obtained from different sensors that have different image resolutions. This presents a challenge for the DoD as it may rule out the use of publicly available overhead datasets for training, further highlighting the importance of being able to produce an accurate model from a limited number of training images specific to the sensor currently of interest.

We also demonstrate the potential for a retrained CNN to correctly classify sub-categories of objects when trained on as few as 200 examples of each image category. We find that our model is able to correctly classify the ship sub-categories tested with an average recall of 0.83 and an average precision of 0.85. Although, this is low compared with our results for a single classification task, it is worth noting that while multiple ships were mis-categorized, they were all mis-categorized as other types of ships. We find that our model has the most difficulty recognizing small, fast moving ships, but performs well when presented with larger merchant ships. This result indicates that using a retrained CNN

for sub-category classification may be more useful for certain tasks than others. In the case of our dataset, sub-category classification could be useful to pull out merchants from the dataset but would be less useful in identifying the presence of speedboats in a particular region of the oceans.

B. BENEFITS AND RISKS OF USING RETRAINED CNNS FOR OBJECT DETECTION IN SATELLITE IMAGERY

The results of our second set of experiments demonstrate that the technique of transfer learning can also be successfully applied to the task of object detection. Similar to our results for classification of ship sub-categories discussed above, we find that our object detection model has a more difficult time detecting small, fast ships and barges compared to larger merchant vessels in open ocean. To increase model performance in this area we conclude that the detection threshold must be set to a very low value. Although lowering the threshold in this manner increases the number of false positives produced by the model, it produces an overall improvement in model performance. Additionally, the increase in false-positives is not a large concern due to the motivation behind our work: we aim to produce a model that will allow intelligence analysts to quickly find objects in satellite imagery that may be of interest. For this reason, we are more concerned with eliminating as many false-negatives as possible. An analyst using this model, would be able to quickly sort out any false-positives present when viewing the data for further analysis.

Using a retrained CNN for the object detection tasks carries risks similar to those associated with the object classification task in terms of the impact of noise and blur on model performance. We find that the addition of noise to our test set only produces an increase in false-negatives and a reduction of maximum F-score from 0.91 to 0.81. As with our object classification results, the addition of noise to the training set as well as the test set helped to mitigate this issue. The addition of noise to both the training and the test set, also produced a higher number of false positives. However, as discussed above we are concerned less with many false positives than we are with a large number of false-negatives due to the ultimate use case for our model. We find that with blurred images the largest impact is observed if there are blurred images in only the training set. This created a much higher false-positive rate (0.35 compared to 0.15) when presented with our test set. These

results once again indicate that when training a new model, care must be taken to ensure that the training set is fully representative of the data that the model will ultimately be used to detect.

Finally, we show that our object detection model is also able to detect sub-categories of ships; producing an average recall of 0.82 and an average precision of 0.86. Much like our object classification model, the model performs best on the task of detecting merchant ships. This is once again an indication that using a model trained for sub-category classification may be more or less useful depending on your target object. However, the imagery used in our experiments was taken at a relatively low resolution of 3m. There is the potential that sub-category classification models may perform better when trained on higher resolution imagery, as we discuss further in our future work section below.

C. FUTURE WORK

We show that using retrained CNN for object classification and detection has the potential to allow the DoD to quickly train new models for novel datasets without a requirement for large amounts of training data or machines with powerful GPUs. However, there are several areas that require future study that we were not able to address within the scope of this thesis. As mentioned briefly above, all of our data came from the same 3m resolution sensor [21]. Here, we use blur to approximate the effect that including images of lower resolution to the test or training sets will have on model performance. However, we do not directly test any images obtained from a secondary remote sensing platform. It would be useful to compare model performance on models trained on images from several different remote sensing platforms at differing resolutions.

Additionally, this work could be expanded by re-training the VGG-16 object classification and detection architectures for use on other DoD related image sets such as airplanes or ground vehicles. We limited the scope of this thesis to the maritime domain; however, the land domain may provide a unique set of challenges due to the increase in background noise from other objects such as buildings that we did not need to address when focusing primarily on areas of open ocean. Lastly, the techniques used in this thesis should eventually be tested on images from the actual remote sensors used by DoD

intelligence analysts. The work in this thesis provides a solid proof of concept for the use of retrained CNNs in the DoD; however, it will be important to determine the performance of these tools on the actual data and tasks for which we aim to see them used.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet large scale visual recognition challenge," *Int. Journal of Comp. Vis.*, vol. 115, no. 3, pp. 211–252, 2015. [Online] doi: 10.1007/s11263-015-0816-y

[2] T. H. Paul, "Deep learning for media analysis in defense scenarios--an evaluation of an open-source framework for object detection in intelligence related image sets," M.S. thesis, Dept. of Comp. Sci., NPS, Monterey, CA, USA, 2017. [Online] Available: http://hdl.handle.net/10945/55514

[3] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Comp. Vis. and Patt. Rec.*, 2009. [Online]. doi: 10.1109/CVPR.2009.5206848

[4] M. A. Nielsen. (2015). *Neural Networks and Deep Learning*. [Ebrary version]. [Online]. Available: http://neuralnetworksanddeeplearning.com/

[5] A. Karpathy, "Cs231n: Convolutional neural networks for visual recognition," Online Course, 2016. Available: http://cs231n.github.io/

[6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv:1409.1556 [cs.CV], Sep. 2014.

[7] S. Sharma, "Epoch vs batch size vs iterations," September 22, 2017. [Online]. Available: https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9

[8] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Jour. of Mach. Lear. Res*, vol. 15, pp. 1929-1958, Jun. 2014. [Online]. Available: http://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Comm. of the ACM*, vol. 60, no. 6, pp. 84–90, May 2017. [Online]. doi: 10.1145/3065386

[10] Machine Learning Cheatsheet, "Loss functions," Accessed November 29, 2018. [Online]. Available: https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html

[11] A. Walia, “Types of optimization algorithms use in neural networks and ways to optimize gradient descent,” Jun. 10, 2017 [Online]. Available: https://towardsdatascience.com/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-95ae5d39529f

[12] G. Hinton, N. Srivastava, and K. Swersky, “Neural networks for machine learning,” Accessed November 29, 2018. [Online]. Available: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

[13] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, and A. Berg, “SSD: single shot multibox detector,” arXiv:1512.02325[cs.CV], Dec. 2016.

[14] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” arXiv:1311.2524 [cs.CV], Oct. 2014.

[15] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, “DeCAF: a deep convolutional activation feature for generic visual recognition,” arXiv:1310.1531 [cs.CV], Oct 2013.

[16] TensorFlow, “How to retrain an image classifier for new categories,” November 1, 2018. [Online]. Available: https://www.tensorflow.org/hub/tutorials/image_retraining

[17] Google, “TensorFlow,” Accessed February 10, 2018. [Online]. Available: https://www.tensorflow.org

[18] Keras “Keras: The Python deep learning library,” Accessed May 1, 2018. [Online]. Available: https://keras.io

[19] Stanford Vision Lab, “ImageNet,” 2016. [Online]. Available: http://www.image-net.org

[20] Planet Team, “Planet application program interface: in space for life on earth,” Planet Labs Inc, 2018. [Online]. Available: https://api.planet.com

[21] Planet Labs Inc, “Planet imagery product specifications,” Jan. 2018. [Online]. Available: https://www.planet.com/products/satellite-imagery/files/Planet_Combined_Imagery_Product_Specs_December2017.pdf

[22] T. Fawcett, “An introduction to ROC analysis,” *Patt. Rec.* Letters, vol. 27, no. 8, Jun. 2006. [Online]. doi: 10.1016/j.patrec.2005.10.010

[23] M. Sokolova, N. Japkowicz, and S. Szpakowicz, “Beyond Accuracy, F-score, and ROC: a Family of Discriminant Measures for Performance Evaluation,” *in AI 2006: Adv. in Artif. Intel.: 19th Aus. Joint Conf. on Artif. Intel., Hobart, Australia,* Dec. 2006. [Online] doi: 10.1007/11941439_114

[24] J. Jones, “Parts-based detection of AK-47s for forensic video analysis,” M.S. thesis, Dept. of Comp. Sci., NPS, Monterey, CA, USA, 2010. [Online]. Available: http://hdl.handle.net/10945/5200

[25] D. M. Camp, “Evaluation of object detection algorithms for ship detection in the visible spectrum,” M.S. thesis, Dept. of Elec. and Comp. Eng., NPS, Monterey, CA, USA, 2013. [Online]. Available: http://hdl.handle.net/10945/38891

[26] L. Sharpe, “Feature sets for screenshot detection,” M.S. thesis, Dept. of Comp. Sci., NPS, Monterey, CA, 2013. [Online]. Available: http://hdl.handle.net/10945/34741

[27] Princeton University, “WordNet: a lexical database for English,” 2018. [Online]. Available: https://wordnet.princeton.edu

[28] UNC Vision Lab, “ImageNet Large Scale Visual Recognition Challenge 2017 (ILSVRC2017),” 2017. [Online]. Available: http://image-net.org/challenges/LSVRC/2017/index

[29] C. Szegedy, P. Wei Liu, S. Yangqing Jia, D. Sermanet, D. Reed, V. Anguelov, A. Erhan, A. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions.” *in Comp. Vis. and Patt. Rec*, 2015. [Online]. doi: 10.1109/CVPR.2015.7298594

[30] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” arXiv:1512.03385 [cs.CV], Dec. 2015.

[31] Y. Yao, Z. Jiang, H. Zhang, D. Zhao, and B. Cai, “Ship detection in optical remote sensing images based on deep convolutional neural networks,” *Journal of App. Rem. Sens.*, vol. 11, no. 4, 2017. [Online]. doi: 10.1117/1.JRS.11.042611

[32] K. Rainey, S. Parameswaran, J. Harguess, and J. Stastny, “Vessel classification in overhead satellite imagery using learned dictionaries,” *in Proc. of SPIE*, vol. 8499, Oct. 2012. [Online]. doi: 10.1117/12.928875

[33] K. Rainey, S. Parameswaran, and J. Harguess, “Maritime vessel recognition in degraded satellite imagery,” *in Proc. of SPIE*, vol. 9090, no. 4, Jun. 2014. [Online]. doi: 10.1117/12.2049868

[34] Y. Yang and S. Newsam, “Bag-of-Visual-Words and spatial extensions for land-use classification,” *in ACM SIGSPATIAL Int. Conf. on Adv. in Geo. Info. Sys.*, Nov. 2010. doi: 10.1145/1869790.1869829

[35] I. Demir, K. Koperski, D. Lindenbaum, G. Pang, J. Huang, S. Basu, F. Hughes, D. Tuia, and R. Raskar, “DeepGlobe 2018: a challenge to parse the earth through satellite images,” arXiv:1805.06561 [cs.CV], May 2018.

[36] Defense Innovation Unit Experimental, “DIUx xView 2018 detection challenge,” 2018. [Online]. Available: http://xviewdataset.org

[37] Kaggle Inc, “Airbus ship detection challenge,” 2018. [Online]. Available: https://www.kaggle.com/c/airbus-ship-detection. [Accessed 7 October 2018].

[38] K. Rainey, J. Reeder, and A. Corelli, “Convolution neural networks for ship type recognition,” *in Proc. Of SPIE*, vol. 9844, no. 9, May 2016. [Online]. doi: 10.1117/12.2229366

[39] S. Qi, J. Ma, J. Lin, Y. Li, and J. Tian, “Unsupervised ship detection based on saliency and S-HOG descriptor from optical satellite images,” *IEEE Geosci. Remote Sens. Lett*, vol. 12, no. 7, Mar. 2015. [Online]. doi: 10.1109/LGRS.2015.2408355

[40] Z. Zou and Z. Shi, “Ship detection in spaceborne optical image with SVD networks,” *IEEE Trans. Geosci. Remote Sense*, vol. 54, no. 10, Oct. 2016. [Online] doi: 10.1109/TGRS.2016.2572736

[41] G. Nie, P. Zhang, X. Niu, Y. Dou, and F. Xia, “Ship detection using transfer learned single shot multi box detector,” *in ITM Web of Conf*, vol. 12, Jan. 2017. [Online] doi: 10.1051/itmconf/20171201006

[42] F. Chollet, “Building powerful image classification models using very little data,” The Keras Blog, June 5, 2016. [Online]. Available: https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

[43] T. Amaratunga, “Using bottleneck features for multi-class classification in Keras and Tensorflow,” Codes of Interest, August 8, 2017. [Online]. Available: https://www.codesofinterest.com/2017/08/bottleneck-features-multi-class-classification-keras.html

[44] P. Ferrari, “SSD keras,” GitHub, May 2018. [Online]. Available: https://github.com/pierluigiferrari/ssd_keras

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California