2022

Deep Learning Detection in the Visible and Radio Spectrums

Greg Clancy Murray
West Virginia University, gcm0011@mix.wvu.edu

DEEP LEARNING DETECTION IN THE VISIBLE AND RADIO SPECTRUMS

by

Greg Clancy Murray

Thesis submitted to the
Benjamin M. Statler College of Engineering and Mineral Resources
at West Virginia University

in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Science

Thirimachos Bourlai, Ph.D., Committee Chairperson
Natalia Schmid, Ph.D.
Yuxin Liu, Ph.D.
Frances Van Scoy, Ph.D.

Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia
2022

Keywords: CNN, Detection, Satellite, RFI

ABSTRACT

Deep Learning Detection in the Visible and Radio Spectrums

by Greg Clancy Murray

Deep learning models with convolutional neural networks are being used to solve some of the most difficult problems in computing today. Complicating factors to the use and development of deep learning models include lack of availability of large volumes of data, lack of problem specific samples, and the lack variations in the specific samples available. The costs to collect this data and to compute the models for the task of detection remains a inhibitory condition for all but the most well funded organizations. This thesis seeks to approach deep learning from a cost reduction and hybrid perspective — incorporating techniques of transfer learning, training augmentation, synthetic data generation, morphological computations, as well as statistical and thresholding model fusion — in the task of detection in two domains: visible spectrum detection of target spacecraft, and radio spectrum detection of radio frequency interference in 2D astronomical time-frequency data. The effects of training augmentation on object detection performance is studied in the visible spectrum, as well as the effect of image degradation on detection performance. Supplementing training on degraded images significantly improves the detection results, and in scenarios with low factors of degradation, the baseline results are exceeded. Morphological operations on degraded data shows promise in reducing computational requirements in some detection tasks. The proposed Mask R-CNN model is able to detect and localize properly on spacecraft images degraded by high levels of pixel loss. Deep learning models such as U-Net have been leveraged for the task of radio frequency interference labeling (flagging). Model variations on U-Net architecture design such as layer size and composition are continuing to be explored, however, the examination of deep learning models combined with statistical tests and thresholding techniques for radio frequency interference mitigation is in its infancy. For the radio spectrum domain, the use of the U-Net model combined with various statistical tests and the SumThreshold technique in an output fusion model is tested against a baseline of SumThreshold alone, for the detection of radio frequency interference. This thesis also contributes an improved dataset for spacecraft detection, and a simple technique for the generation of synthetic channelized voltage data for simulating radio astronomy spectra recordings in a 2D time-frequency plot.

I dedicate this thesis to my loved ones — my children Clancy Isaac, Nora, Elias, Gwendolyn, and my trusted partner and truest friend Jamie — and to the God who establishes my thoughts, and gives wisdom (Prov. 16:3, 2:6)...

Acknowledgments

With my deepest appreciation I would like to mention my committee chair and advisor Dr. Thirimachos Bourlai for his patient support and wise guidance during these few years on the path to completing this thesis and this Master's degree. I have been the beneficiary of his consistent pattern of lifting up his students to reach new heights, and his belief in my potential has helped when I lacked such belief myself. As a mentor and supervisor, he has developed my personal as well as my professional character. I am sincerely grateful.

I would like to mention my committee members for their patience and guidance in this endeavor. I am grateful to Dr. Natalia Schmid for her honesty and kindness in sharing her wealth of knowledge. Without her, this thesis would be lacking in many areas. I am likewise grateful for Dr. Yuxin Liu and Dr. Frances Van Scoy for their suggestions and guidance.

I would like to thank Dr. Anthony Pyzdrowski, who gave me the encouragement to pursue my dreams, and to make no excuses; it is possible after all (let me practice my surprised face.)

I am thankful to my colleagues Suha, Ananya, Dylan, Jake, Matt, and Kelsey for their cooperation and friendship in our shared journey.

Contents

List of Figures

List of Tables

Acronyms

NASA	National Aeronautics and Space Administration
ESA	European Space Administration
AGWN	Additive Gaussian White Noise
IoU	Intersection over Union
COCO	Common Objects in Context (Standard Dataset)
AP	Average Precision
mAP	mean Average Precision
CNN	Convolutional Neural Network
VOC	Visual Object Challenge
SPEED	Spacecraft PosE Estimation Dataset

Chapter 1

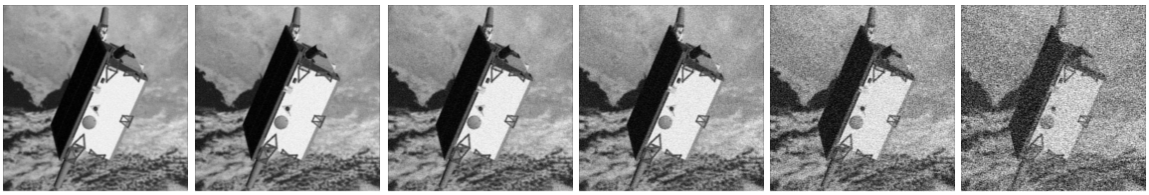
Introduction

1.1 Motivation

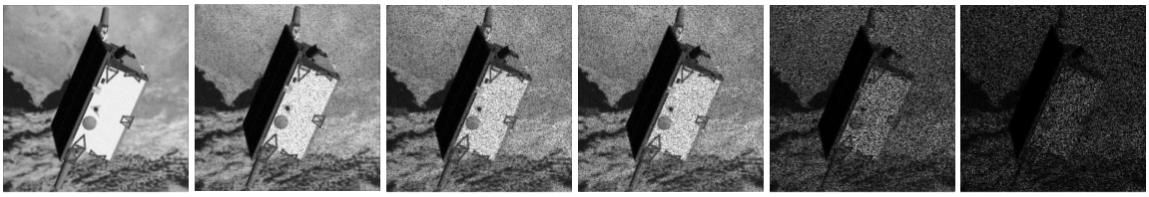
The use of deep learning models in space applications is a growing field; due to the extensive nature of space research, travel, Earth observations, astronomy (both visible and radiometric) and the plethora of data that come from such activities, state-of-the-art machine learning approaches are invaluable to the solving of space related problems.

1.1.1 Image Degradation in Spacecraft Detection

One such problem is the detection and localization of target spacecraft from an observing spacecraft. Many tasks can be contemplated for this situation: inspection, craft repair, surveillance, and military action, to name just a few. This is complicated by a few different factors, however. There are many different kinds of spacecraft, both known and unknown. The ability to detect what is a spacecraft is partially reliant on the proper classification of the image scene, and for the automatic detection and localization, a large database of images may be needed to determine if localized objects are indeed the target spacecraft. Dataset contributions from Dung, et al [1] are one



(a)



(b)

Figure 1.1: a. Increasing levels of noise, as might be produced in the imaging device affected by radiation. b. Levels of pixel loss increasing, as might be experienced in damaged charge coupled device imaging hardware.

step toward a more robust detection framework for satellites in the wild. The SPEED dataset, introduced by Kisantal, Sharma and Mate, et al [2] (which is used in this thesis) is another important contribution to this task. Within the sphere of detection of known spacecraft, however, there remain a few considerations. The underlying assumption of this detection and localization (and in some cases pose estimation) task is that the spacecraft in question is a non-cooperative one. Before pose estimation can be done, or even localization, the craft must be detected. This detection may be complicated by natural or unnatural means. Two such complications include pixel loss due to CCD damage and radiation induced noise, emulated in Fig. 1.1 with the image of a target satellite spacecraft.

1.1.2 RFI Detection in Astronomical Data

The second thesis topic, another similarly space oriented detection task, is focused on the much different data of channelized voltages representing the recording of astronomical signals. Frequently, these signals are disturbed with terrestrial interference

from cell phones, two-way radios, microwave ovens, etc. This complicates the discovery of important astronomical insights, such as pulsars. Because of this pervasive threat to the radio spectrum being surveyed by the world's radio telescopes, it is important to detect RFI (radio frequency interference) and remove it from the data before further processing.

1.2 Problem Statement

The use of deep learning models in the space oriented applications of spacecraft detection and RFI detection and mitigation is difficult for many reasons. To develop a convolutional neural network spacecraft detection model that can continue to operate in the face of intentional or unintentional damage to imaging sensors, and to develop a deep learning model that can detect and label various RFI in time-frequency plots of channelized voltages, certain challenges must be overcome. The challenges include:

1. A lack of sufficient data to robustly train and test deep learning models (which often require tens of thousands of images or samples at a minimum).

2. The scant availability of data which simulates or has captured the difficult problem of degraded or otherwise corrupted recording of events.

3. The extensive economic and computational costs of developing models that can be robust enough to overcome all expected obstacles in this problem space of detection and localization, as well as the cost to record or collect sufficient data to develop a model that would be useful in practice.

To overcome these challenges, the cost of collecting such a large amount of data must be mitigated, the conditions for training against degraded or corrupted samples must not increase the cost of data collection, and cost of implementing these models

must provide a sufficient benefit to justify the cost in terms of training and the computational costs of hardware, software, maintenance, and ease of use.

1.3 Contributions of Thesis

The motivation of the first thesis topic is the improvement of deep learning knowledge in the task of detection and localization of spacecraft with the use of Mask R-CNN, and to improve the understanding of the effects of image degradation on this detection task, and how the combination of basic mathematical image operations can assist in the deep learning sphere. The use of Mask R-CNN in spacecraft and satellite detection as in the author's previous work, Murray, Bourlai, and Spolaor [3] is included in this thesis, with additional work related to image degradation mitigation.

The second topic of this thesis is the development of a deep learning model which can successfully outperform a state-of-the-art algorithm by the addition of a fusion of convolutional neural networks and statistical inputs. By using parallel processing hardware and convolutional neural networks trained on labeled data, models can consistently detect interference in a variety of time-frequency plot data, with improvements resulting from fusion with statistical outputs.

Firstly, This thesis proposes a baseline comparison of Mask R-CNN performance under conditions of image degradation, and explores degradation mitigation from both an image processing and training augmentation perspective. This work shows how to overcome the challenge of developing a model for spacecraft detection and localization which is not hampered by the dearth of data, utilizing image augmentation to improve the amount of data, and exploring the use of mathematical operations of dilation and erosion to overcome image degradation under certain conditions. The motivation for this study is the use of such off-the-shelf object detection architectures under the harsh conditions that might be expected in the environment of outer space; this

would further improve the development of this architecture, and the use of the SPEED spacecraft image dataset provided by [2] and encourage more exploration of spacecraft detection and segmentation in the future. By applying this model to a dataset with simulated radiation induced noise and pixel loss, this work shows that the model can be made robust enough to carry out the detection of a given satellite. The use of mathematical operations is also shown to have useful properties in certain scenarios, which may improve the detection of satellite spacecrafts along with properly trained models, with limited additional computational overhead.

Secondly, This thesis explores the use of the U-Net model combined with various statistical tests and the SumThreshold technique in an output fusion model for the detection of radio frequency interference, tested against a baseline of SumThreshold alone. Deep learning models such as U-Net have been leveraged for the task of radio frequency interference labeling (flagging). Model variations on U-Net architecture design such as layer size and composition are continuing to be explored, however, the examination of deep learning models combined with statistical tests and thresholding techniques for radio frequency interference mitigation is in its infancy. By fusing statistical inputs, the SumThreshold method, and two different deep learning models, the obstacle of developing a single extremely capable deep model for RFI detection is circumvented, allowing quicker (and more thus more economic) training of deep models, with results exceeding those of individual approaches.

Thirdly, in the development of these two deep learning models, this thesis provides improved datasets and synthetic data generation tools for exploring these detection problems in the future. This work provides an improved dataset for satellite detection, with an extensive hand-labeled subset of the SPEED dataset, covering various background conditions, relative satellite sizes, lighting conditions, and satellite poses. This work also provides a pre-generated set of synthetic channelized voltage time-series data for the detection of RFI, as well as software for generating such images.

1.4 Organization of Thesis

This thesis is organized in the following chapters and sections:

- Chapter 2 — An overview of the primary deep learning topics that are relied on throughout this thesis as it relates to training, bias and variance, knowledge transfer, and other key concepts.

- Chapter 3 — An overview of work related to this thesis and some descriptions and exposition of the approaches used in similar deep learning object detection problems, the background work on target spacecraft detection, as well as the foundational work in radio frequency interference detection and mitigation as it relates to deep learning.

- Chapter 4 — The development of the Mask R-CNN model used for the detection and localization of target spacecraft, including the development and use of the SPEED spacecraft dataset, the training regiment with augmented and unaugmented dataset comparisons, model backbone evaluation, mathematical image processing operations, and results of training experiments.

- Chapter 5 — The development of a deep learning U-Net fusion model for the purposes of detecting radio frequency interference in raw channelized voltages presented as 2D time-frequency plots (similar to spectrograms), the use and evaluation of statistics combined with U-Net model, and the training preprocessing techniques used in the fusion model, along with experiment results.

- Chapter 6 — Summary of thesis contributions and direction of future work, discussion of improvements to this work and concluding remarks.

- An appendix containing examples, code, and other supplemental material.

Chapter 2

Deep Learning Fundamentals

2.1 Training a Deep Learning Model

The major bottleneck of deep learning models is in the collection and labeling of data. Deep learning models trained from scratch (that is, without the use of knowledge transfer) spend a great deal of the training time learning basic functions to filter the input images. If the data is not representative of the problem space, or if the data simply lacks variation, the resulting model will not be useful. This is due to the bias variance trade-off. Transfer learning alleviates this problem by providing model weights which have been previously learned on a great deal of data, but this does not remove the requirement of a large, representative corpus of data to develop such models. Training augmentation can increase the training samples, but cannot decrease the time needed to train a large deep learning model. Using the two together can often solve this major bottleneck.

2.1.1 Bias and Variance

The bias-variance tradeoff is the eternal battle of deep learning, and much like the "force" of Star Wars fame, the universe of deep learning must function within the bal-

ance of these two conditions, and neither extreme bias nor extreme variance produces models which are useful. A model which is biased has not yet learned features representative of the data, and is overly simplistic. During training, the model has usually been exposed to a small subset of data from the true population of samples and has not yet learned (weighted the network toward) features which excel at determining the class we are interested in. The model may learn a feature which seems to predict the target class, however, this illusion rests on a foundation of sand; the features are shortcuts applicable only to the small subset of the true population, not distinguishing features of the whole. Figure 2.1(a) shows a concrete example of this concept in the classification of flowers. This flower classification model has only been trained on daffodils and dandelions so far, and unfortunately, the model has linked flowers inextricably with yellow. The tradeoff between bias and variance can be visualized as a curve, as seen in Figure 2.2(a).

The result of a highly biased model is underfitting, which is when a model fails to discover the underlying features that represent the true population. Contrasted with the opposite condition, a high variance model results in overfitting, where often a complex model learns many unimportant features (considered as noise) and also does well in training, but has a difficult time generalizing to the true population due to the many conflicting features. The optimal region of this tradeoff will result in relatively low bias and variance, with tension between them. All other things being equal, a less complex model is preferred, as increased complexity often leads to high variance. In Figure 2.3(a), a good fit can be found with a simple quadratic function. The underfitting linear function fails to capture the essence of the true function, while the overfitting complex function fails to model the true function because it is distracted by random sample variations.

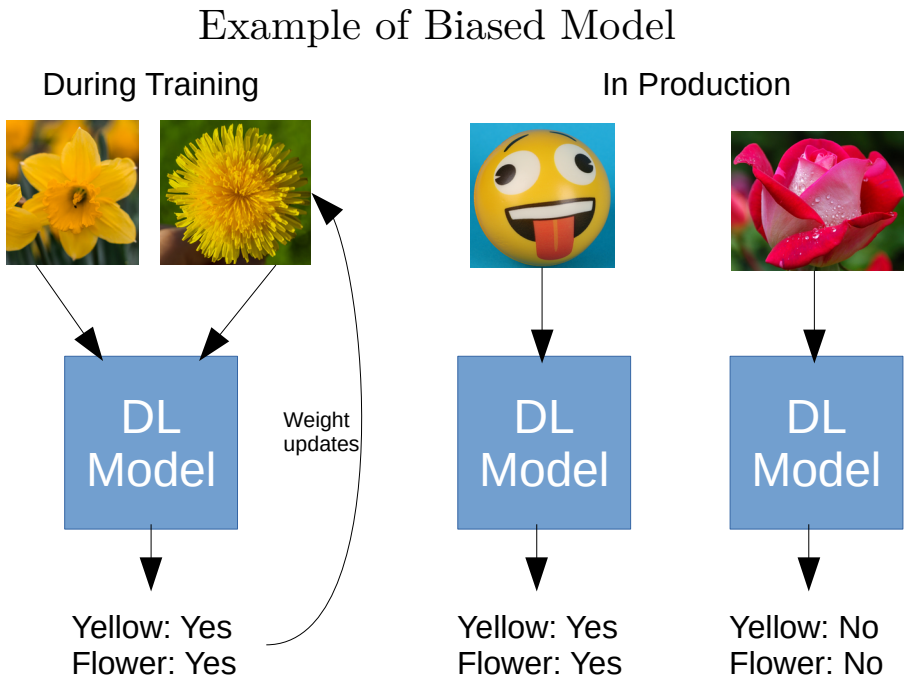


Figure 2.1: This flower classification model has been trained on only daffodil and dandelion images. The model has associated yellow with flowers, which leads to a production model which cannot properly identify a rose as a flower, and incorrectly classifies a yellow ball as a flower. Photos provided by Pexels.com. Used with permission.

2.1.2 Training Augmentation

To produce a model with low bias, the model needs to be exposed to a large number of samples which are representative of the population of the target class. In many cases, there are far too few samples to properly train a deep learning model. As an example, consider the Tufted Titmouse, a bird which infrequently lights upon our family's bird feeder. If I were to train a model to detect when this bird has returned in a live video feed, I would need to use many different images of this bird for the model to be successful. For this example, if I only have one image of this bird, with some simple changes to this image, I can create brand new Titmouses (not to imply that all Tufted Titmouses look alike). Figure 2.4 shows how simple image manipulations

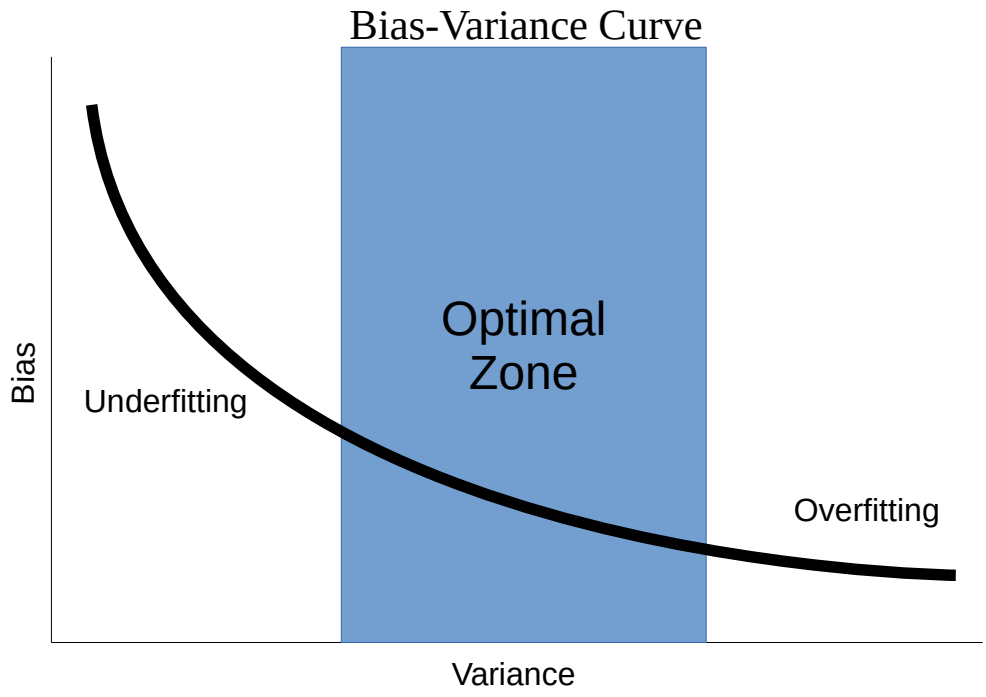


Figure 2.2: An increase in bias will decrease variance and result in underfitting, while an increase in variance will decrease bias and result in overfitting. Balancing these two elements results in, not perfect, but optimal performance.

such as translations and rotations, blurs, and the like can result in novel images which can improve the training corpus for the deep learning model. As a human, we can see that these images are clearly variations on the same bird, but to the deep learning model, each presents a new and useful view of the Tufted Titmouse.

2.1.3 Regularization

When dealing with the problem of a highly variable model, that is, a model which overfits the training data and perhaps does not generalize as well as it could, we must regularize or smooth the model. This can be done through various means, each with the same goal of creating a more generalizing model. We can make the model more sparse (by removing neural connections), or decrease the influence of

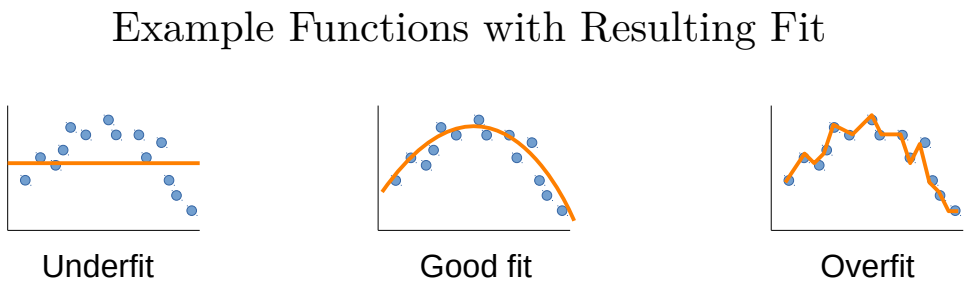


Figure 2.3: The simplistic linear function on the left results in a model which cannot predict future points. The right model, however, is overly concerned with subtle random variations in the samples. A good fit (middle) is found by balancing bias and variance, and the less complex model is preferred for this purpose.

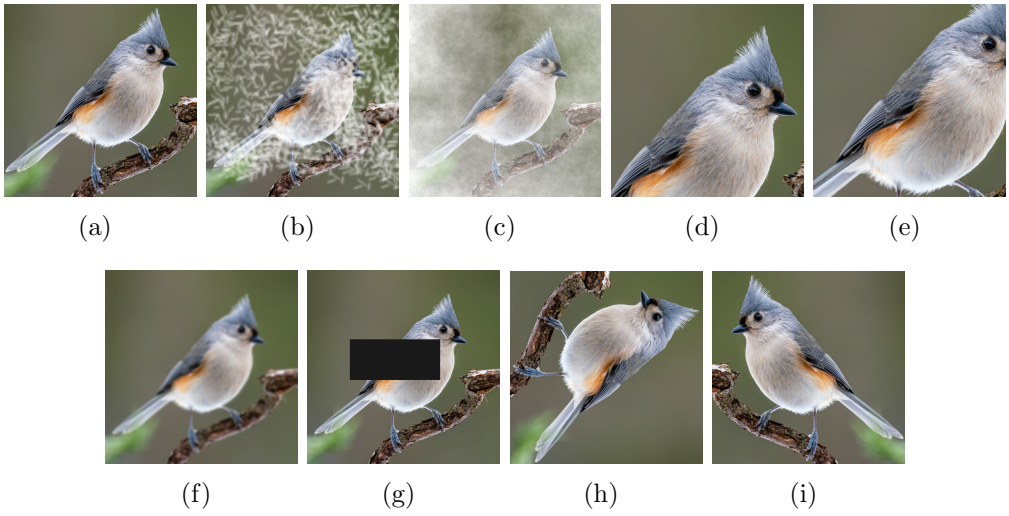


Figure 2.4: Augmenting the original image (a) of the Tufted Titmouse results in eight new images. Simulated weather (b), (c), cropping and clipping (d),(e), blurring (f), occlusion (g), rotation and flipping (h),(i). Photo provided by Pexels.com. Used with permission.

features which are not as important. We can temporarily cut off random neural pathways during training to prevent the model from relying on learned "crutches" or shortcuts. Ultimately regularization seeks to create a simpler model which has learned the essence of the problem. Paradoxically, a model which is too good on the training data is unlikely to be generalizable, and a regularized version of the same

model, while performing worse on the training data, will often perform better on real world data. To prevent a deep learning model from becoming an "expert" during training, a simple technique of regularization is to stop the training early. Anyone who has been told to not "overthink it" when being taught a new task has implicitly had this form of regularization applied to their training. The more training time and the more samples the model is exposed to, the better generalizing the model becomes as well, considering that the data is representative of the true population. Theoretically, if all possible samples were used to train a model, then any test sample would have been trained on, and there would be no need to regularize the model, as the training performance would be equivalent to the test performance. This is possible only for the simplest of problems, and so a model confined by time and space will require valuable "rules of thumb" to be useful in real world scenarios. We can ensure our deep learning model remains in the optimal zone of the bias-variance curve by using one (or many) forms of regularization.

2.1.4 Transfer Learning

Training deep learning models currently requires a large repository of labeled data, on the order of tens of thousands to millions of samples. The manual labor involved in labeling the data alone costs many man-hours, and the collection of this data also requires a great deal of time and money. Once this data is collected, the computationally intensive process of training a deep model on the data is started, and this process may not be complete for weeks or months — if the computational resources are of a sufficient caliber in terms of speed and memory. The final result of these endeavors is a model which has been trained on a wide variety of samples and is theoretically capable of performing well across the problem space. Although this model may have been trained to detect cars or boats, stars or starships, in fact a great deal of the computational resources for this model were invested in learning how to

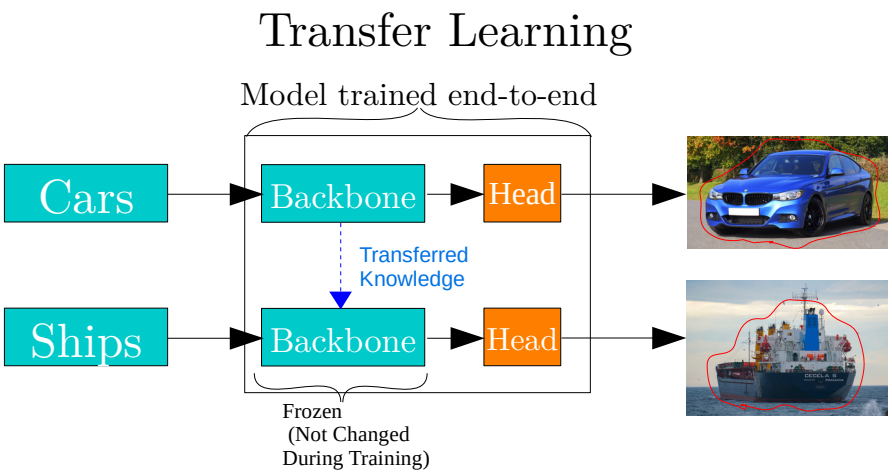


Figure 2.5: The transfer of the backbone of the model, the learned weights, to the second model (bottom) allows the ship detection model to train only the head network and avoid long training times. Photos provided by Pexels.com. Used with permission.

detect generic shapes and edges. Whether the model was trained on ships or on cars, with a reasonable amount of resources, this model could be adapted to detect any number of other objects. For this reason, a model that was trained on a large variety of common objects, such as the Common Objects in Context dataset (COCO), can have its weights transferred to another model to be used in a different context.

In Fig. 2.5(a) the model that was trained on cars from end to end learned how to distinguish a great many number of shapes, edges, colors and the like. The backbone of this model is composed of weighted values for the neural connections of the model which were learned through many iterations of training on car images. For a Convolutional Neural Network, the weights are connected to feature mappings from each layer of convolutional operations, determining the kernel for this operation used over the input image. Earlier layers detect broader features such as size, general shape, color, or brightness. Layers further into the model detect finer features, such as doors, windows, mufflers, or side-view mirrors. By removing the very end of this

model which is focused on very car specific features, and training on top of the layers which detect the broad features, we can reuse the learning from the car model for our ship model.

2.2 Summary

Training a deep learning model from scratch is an important part of the ecosystem in the modern machine learning paradigm, but it is not necessary to always do so. Using the weights from previously trained models by transferring them to new models and adjusting them to fit the current problem can reduce training in terms of time and CPU cycles, ultimately reducing costs. By also increasing the usable corpus of training and testing data using data augmentation techniques, we can improve the model's ability to generalize and avoid biasing our models.

Chapter 3

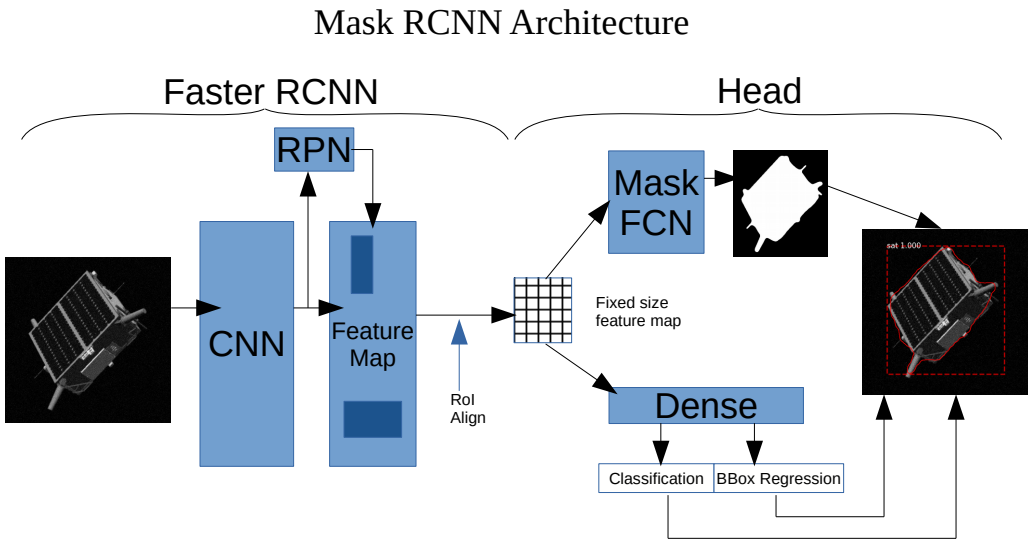
Background and Related Work

3.1 Visible Band Target Spacecraft Detection

Various detection tasks have been approached using the Mask R-CNN architecture in the years since He et al. [4]. Built upon earlier work including Faster R-CNN [5], Mask R-CNN has been used in the detection and segmentation of gas plumes [6], road damage [7], hands [8], cellular nuclei [9], ships [10], and many others [11],[12],[13], [14].

The Mask R-CNN network is composed of a Faster R-CNN network with two head networks, as shown in Fig. 3.1 the latter is composed of a fully connected network for bounding box regression and instance classification, and a convolutional network for instance segmentation. An important deviation from the Faster R-CNN network is the use of an alignment focused region of interest during the pooling operation, referred to as RoI Align. This operation improves mask segmentation over the Faster R-CNN RoI max pooling operation by keeping fractional offsets and not performing quantization. This is illustrated in [[4], Fig. 3].

Mask R-CNN usage for terrestrial-oriented tasks is well established. The performance of CNN architectures with image degradation factors has been explored by [15]. Particular attention was paid to Gaussian noise and minimal required noise for



(a)

Figure 3.1: The Mask RCNN architecture is based on the Faster RCNN backbone. The RoIAlign replaces the Faster RCNN RoIPooling, which improves the accuracy of the localization. The head unit of the model is the primary focus of training, with transfer learning used to fill the backbone network.

images to be incorrectly classified. Also [15] has explored how training with augmented images improved the performance of the models. The conclusion was that while augmentation improved classification tasks, noise is still able to confuse the model, even at lower, imperceptible levels. The effects of image degradation on CNN classification are examined by [16] for a wide range of degradation factors, as well as many approaches to degradation removal and the associated improvements and regressions. Section 4 follows the general approach of [16], in examining increasingly degraded images. While training augmentation is one method explored, the applications of mathematical morphology operations (closing and opening) [17], [18], and the use of Gaussian smoothing [19] to the performance of object detection are additional approaches investigated in this thesis. As explained by [20], the use of classical algorithms can still provide valuable contributions to computer vision; the use of dilation operations on the depth completion task shows that image processing alone

can provide computationally inexpensive ways to improve results. Bench-marking for classification tasks introduced by [21] are valuable contributions for general evaluation of neural network robustness in cases of common image degradation, although what can be considered common degradation or corruption is often different depending on the context of the detection or classification, such as with exploration in the harsh conditions of space.

Morphological Operations: Opening and Closing

The morphological operations Opening and Closing are compound operations of erosion and dilation, and are the duals of each other. Given a kernel K of a given size and shape, (also referred to as a structuring element), the erosion function E, and the dilation function D, the compound operation Opening definition is as follows:

$$(f \circ K) = (D \circ E) \circ K, \tag{3.1}$$

and the compound operation Closing definition,

$$(f \circ K) = (E \circ D) \circ K. \tag{3.2}$$

These operations are so named for the effect they have on gaps and edges of a binary image. The Opening operation will tend to create gaps where thin edges of pixels connect between larger groups of pixels, and the Closing operation will tend to fill in the holes where there small gaps. The choosing of the kernel for this operation is an important factor in the effectiveness of the removal of gaps or stray pixels.

3.1.1 Use of The SPEED Dataset

Pose estimation, the task of determining an object's orientation and distance from a point of reference (e.g. a camera), has been restricted in deep learning due to

the limited availability of quality data. For research on spacecraft pose estimation to advance, more deep learning datasets for spacecraft pose estimation would need to be developed. To that end, the detection, tracking, and pose estimation of non-cooperative spacecraft was the focus of the July 2019 Kelvins challenge [2]. The challenge asked competitors to improve upon some baseline results in the pose estimation task. The dataset used was compiled using synthetic and real images derived by a 3D model of the Tango spacecraft and a camera model accurately representing the camera aboard the Mango spacecraft. Because of this challenge, other researchers have used this dataset, as seen in [22], and [23]. Additionally, work on pose estimation using segmentation driven point detection is discussed by [24].

3.2 Earth Based Spacecraft Observation and Detection

3.3 Radio Frequency Interference Detection

The approaches to RFI detection and mitigation have changed since the recent feasibility of applying neural network and deep learning models to this problem. The contributions of classical statistics, however, cannot be overlooked; their simplicity of implementation and speed have enabled detection of astronomical signals from fast spinning radio pulsars, to the compositions of galaxies, while avoiding serious contamination of the signals. Descriptions of traditional methods for RFI removal and comparisons of statistical methods of weak RFI removal has been compared by [25] giving a good understanding of the discriminatory power of each method, and the limitations. The most simplistic of these methods is time domain and frequency domain blanking. The advantage of these methods is in limiting the computational overhead by focusing on amplitude thresholds, that is, power levels that exceed a pre-

defined variance. This is often executed online during data collection during the time integration of the signal. This leads to the disadvantage of this method; when the RFI power is integrated over time, or when the frequency resolution is quantized into larger bins, short bursts of signals and narrow signals can be missed. Spectrogram methods are also available, which use image processing techniques (edge detection, histogram information, etc.) to detect RFI in images which represent the frequency window over a time interval (the images are derived from Fourier transform of the signal data.) This thesis relies on interpretation of the data as channelized voltages presented in a 2D time-frequency plot, a form similar to a spectrogram, for the purposes of processing using deep learning models originally designed for image inputs as in the original U-Net model [26]. The statistical focus of [25] is on deriving a combination of statistics to deal with RFI in radiometric signal reception. The key concept and assumption of the received data is that "RFI- free radiometric signal should be a zero-mean random Gaussian variable." [25]. They conclude that the kurtosis statistic, followed by the Anderson-Darling test perform the best in regard to normality tests on radiometric data. [25] also reports that the Shapiro-Wilk test has degraded performance for larger sample sizes, and various skew measures are not recommended due to poor performance. As the kurtosis statistic has difficulty with certain signals (for example, pulsed signals of duty cycle 50%) combining these two tests are recommended. Evaluation of kurtosis in sinusoidal RFI detection was examined by [27] confirming this observation by describing the conditions in which the resultant metric would fail to detect RFI, using the first four statistical moments; [27] shows that in the absence of RFI (a correct result) and when the RFI signal duty cycle is 50%, the kurtosis statistic is 3, indicating the possibility of a false negative.

Spectrogram methods include smoothing operations (low-pass filtering) combined with thresholding, and other filtering techniques such as Wiener filtering as shown by [28]. Attention is also on the transformation of the signal into spectrogram form,

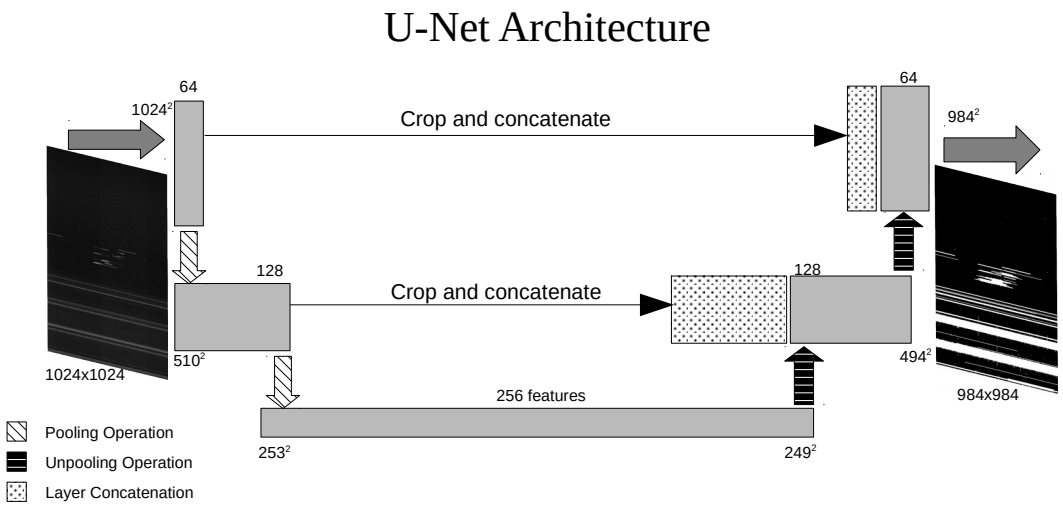


Figure 3.2: The U-Net model is a fully convolutional network which uses two phases of convolutions, and downward pooling phase and an upward unpooling phase. The layers in the first phase are concatenated with corresponding layers in the second phase, often called "skip" connections, providing broad features from the input image as well as detailed features from the normal pipeline.

selecting proper windows for application of filters, and selecting the correct threshold value which [28] determines manually through variations of RFI power. [28] states that this threshold selection is important to the smoothing algorithm; indeed throughout [28] proper selection of windows, thresholds, and estimations of thermal noise play a critical role in using these methods. SumThreshold was introduced as a competitive post-correlation RFI mitigation method by [29] which compares to other spectrogram techniques to advance the art of automated RFI flagging. This algorithm performed better than other manual thresholding techniques in some cases, but most significantly improved workload costs of manual labeling. This algorithm is used in this thesis both for comparison of performance and included in the fusion models. In the deep learning field, U-Net has been utilized for problems as diverse as medical imaging, introduced by [26], as well as RFI detection as in [30], from which this thesis begins the exploration of RFI detection. A diagram of the basic U-Net model is shown in Figure 3.2. Advances in the use of U-Net in RFI detection include

[31] with the AC-UNet model, which incorporates atrous (or dilated) convolutions as [32] into the model, which increases view of the input without increasing the output of the operation, which increases the receptive field. For this thesis, however, this more advanced model is not explored.

SumThreshold

The SumThreshold algorithm was introduced by Offringa, et al [29]. This method uses windows across time and frequency domains to calculate average power levels compared to a calculated threshold. The inputs into the algorithm are independent frequency channels and time bins; for each given channel or bin, a subsequence M (which is a power of two) is evaluated with a moving window of size M, and if the average power calculated for that window exceeds the threshold function, X^{M}, then all samples in this window are flagged. Additionally, the samples which are flagged for a smaller window are not included in the calculation for subsequent windows. In order to limit the computational requirements for this algorithm, windows were limited to powers of two, up to a maximum of 1024 samples per window, which decreases the time complexity from $O(NlogN)$ to $O(N)$, requiring up to only 11 iterations of the algorithm. Offringa [33] gives the algorithm in Table 3.1 for a single iteration of SumThreshold.

Table 3.1: SumThreshold Algorithm (single iteration)

- Slide a window over the data, with size equal to the sub-sequence size M to be tested in this iteration.
- Maintain the sum and the number of unflagged samples in the window. In particular, when moving the window one sample to the right:
  - If the sample to the right was not flagged in previous iterations, add it to the sum and increase the counter.
  - If the sample to the left was not flagged in previous iterations, subtracted it from the sum and decrease the counter.
- For each window position, the average can be calculated by dividing the sum with the counter. If this average exceeds the threshold X, flag all samples in the window.

Chapter 4

Overcoming Image Degradation in Satellite Detection

4.1 Methodology Overview

This section, outlines the differences in implementation between [4] and the approach taken in this thesis based on the work by [34]. Also explained is the method for applying image degradation for evaluation of all experiments using degraded images, the evaluation metrics used in interpreting the results of the experiments, and the methods for training the models used in these experiments.

4.1.1 Model Implementation

The off-the-shelf implementation of Mask R-CNN is adapted from [34]; the model differs from [4] in various ways. Images are resized into square dimensions (from 1920x1200 pixels to 1920x1920 pixels with the SPEED dataset) with zero-padding, as opposed to rescaling the shorter edge of the image to 800 pixels as in [4]. Bounding boxes are generated from the mask data rather than predetermined; the smallest bounding box that fits the masks are used for the bounding boxes. [34] states this

method is accurate to the COCO dataset bounding boxes. The learning rate for the implementation is decreased to 0.001 from 0.02, which [34] states caused an exploding gradient problem. The model code has been changed to allow the application of image processing during training, for the purpose of applying online degradation. The region proposal network anchor scales have been left to the default values, anchors including $[32, 64, 128, 256, 512]$, with ratio values including $[0.5, 1, 2]$, which for the larger images used in this paper, may be detrimental to optimal localization. (see section 4.2.3).

4.1.2 Evaluation Metrics

Commonly used metrics in object detection include precision and recall, average precision (across various configurations), mean average precision (with a variety of interpretations), and intersection over union (IoU). The post-2010 Pascal VOC AP metric is used, by using the maximum precision for each unique recall value to interpolate the values, averaged over 10 IoU thresholds as in the COCO metrics. Also the performance of the model at three different scales is explored, much like the COCO scales metric. However, this thesis differs in the definition of the three scales: images are considered large for objects with greater than about 300K pixels, medium for objects about 100K-200K pixels, and small for images below about 50K pixels. For IoU, a bitmap comparison of the predicted masks and the ground truth mask is used instead of bounding box IoU. Fig. 4.1 gives an example of the detection output, showing the boundary of a detected satellite, and the corresponding ground truth. Considering that this thesis topic explores the performance of this architecture on a single dataset with a single object class, any comparison that would require a more standardized metric, notwithstanding [35], would ultimately be a dissimilar comparison, therefore, these metrics are adequate to evaluate architecture performance under various degradation and training specifications for this specific dataset.

4.1.3 Dataset Description

The data is derived from the SPEED dataset used for the ESA Kelvins Pose Estimation Challenge [2]. The synthetic images, to simulate a captured image, have emulated shot noise and depth of field added by [2]. The full dataset contains 12,000 training images with approximately half of the images having black backgrounds simulating the starfield of space, with the other half having Earth backgrounds, either partially filled or full. Of these black background images, 600 images were selected randomly (evenly for all three scales) for training the models, and 300 were selected for use as testing images for the degradation experiments. On these images, thresholding is used with an empirically determined level to segment the satellite from the black background, creating bitmap masks for each image. The imperfect mask images were then further edited to remove any occlusions or stray pixels. For the complex images, 300 of each the partial and full background images were selected randomly. For each of these images a bitmap mask was created. These masks are 8-bit monochrome images converted to binary masks when the mask image is loaded into memory. The spacecraft images are 8-bit monochrome and are 1920 x 1200 pixels, made square during training with padding to 1920 pixels. Although the spacecraft images in this dataset are grayscale images, the images were converted to RGB values to simplify input into the network.

4.1.4 Synthesis of Degraded Images

For image degradation, the Imgaug library [36] was used for online augmentation during training, and OpenCV [37] was used for image processing experiments. For all the noise factors added to experiments, white Gaussian noise is used to generate the images in accord with [38], with the standard deviation being scaled according to a σ factor, which is multiplied with the maximal value of the value range for the 8-bit monochrome image. For the pixel dropout experiments, a set probability of pixels

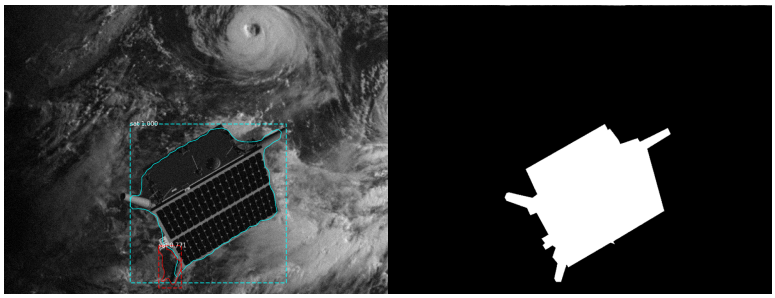


Figure 4.1: A detection example for Mask R-CNN on a complex background image with added noise, $\sigma = 0.03$. Top: The Mask R-CNN detection output gives a category confidence for one correct and one erroneous detection. The bottom right antenna of the satellite has been detected incorrectly as a separate satellite. Bottom: The ground truth of this image is labeled with white pixels as a mask for the detector to learn. ©2021 IEEE.

were dropped from the image for all channels (setting the pixel to black to simulate the loss of that picture element) depending on the experiment. In some experiments, the probability that a pixel will be dropped is drawn from a range of probabilities per image (see 4.2.2.)

4.1.5 Training Approach

The training regimen consists of loading pre-trained weights for the initial setup, and training the head networks for 35 epochs while freezing the backbone layers. Step size was set to 500, the number of training images, with 100 images held out for validation. When training on augmented images, degradation was applied 50% of the time to an image to supplement the training of the model. When image processing was used, it was only applied to images which were degraded. Both the pre-trained weight comparisons and the augmentation experiments used six fold cross-validation, with results being averaged across folds as well as IoU. Degradation experiments were similarly trained without cross-validation, and tested on a set of 100 images x 3 size scales. Due to memory constraints, the batch size for training was set to a single image. Training was done on an NVIDIA Xp Titan GPU. Training took

approximately 16 hours for six fold validation, or about 3 hours for a single model. All training was balanced and stratified across the three size scales (see 4.1.2).

4.2 Effects of Image Degradation to Mask R-CNN Object Detection

In this section, the data and the experimental setup is described. Also the results of the ResNet model comparison, augmentation comparison experiments, and the image degradation experiments are presented. For all augmentation and degradation experiments, RE-CO101 was used as the baseline model.

4.2.1 Pretrained Weight Comparison

In order to compare backbone architectures for later experiments, four combinations of pre-trained weights and ResNets were compared: between ResNet101 and ResNet50, and between COCO pre-trained weights and ImageNet weights. For this experiment, the model combinations were averaged over six folds across IOU $\in [0.5 : 0.05 : 0.95]$. This is reported as mAP@[0.5:0.95] for this single class. In the results presented in Fig. 4.2, ResNet101 using ImageNet weights from Keras [39] show poor performance, while the three other combinations are comparable. For further experimentation, the RE-CO101 model was explored to take advantage of the deeper architecture and the pre-trained COCO weights.

4.2.2 Augmentation Experiments

For the augmentation experiments, the images were disturbed with two kinds of degradation: Gaussian noise, and pixel dropout. For the Gaussian noise augmentation experiments, noise was applied in five settings, with the standard deviation scaling

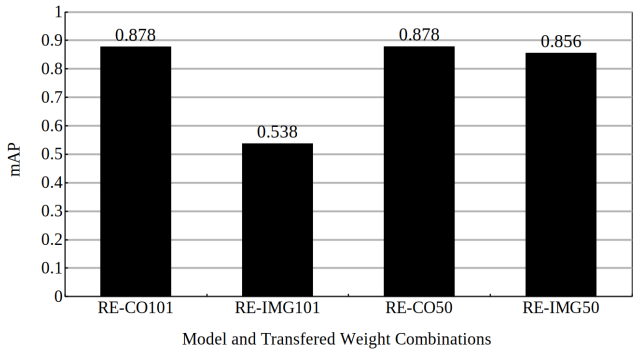
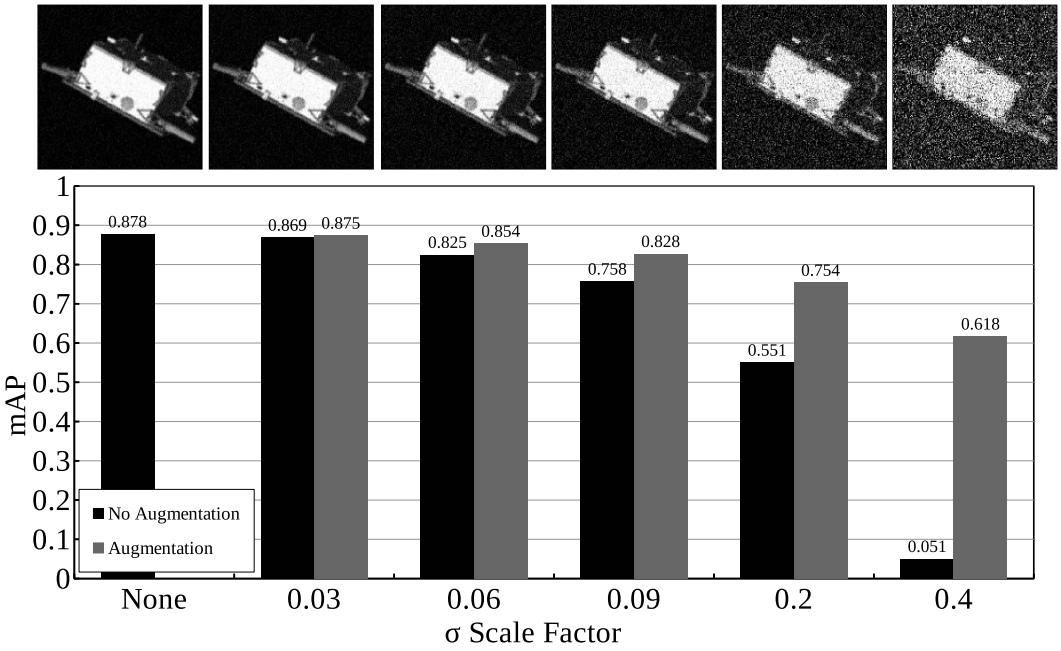


Figure 4.2: ResNet model head networks were trained on the black background SPEED data with previous layers being frozen, for 35 epochs. RE-CO models used weights trained on the COCO dataset, while RE-IMG models used ImageNet trained weights. ResNet101 and ResNet50 models were used as the initial models. Results showed no significant mAP difference between RE-CO101 and RE-CO50 models.

factor $\sigma \in \{0.03, 0.06, 0.09, 0.2, 0.4\}$ where the scaling is multiplied by 255. For the pixel dropout augmentation experiments, again five levels of degradation were applied, with the pixels-dropped percent parameter $\in \{(5, 10), (15, 20), (25, 30), 60, 80\}$; the first three levels of pixel dropout are ranges of probabilities, drawn from for each image for that experiment, so that in the first experiment, an image might have a pixel loss of 5%, while the next image might have 8%, for example. These models were evaluated using stratified six-fold cross validation using the 600 training images, averaged across 10 IoU (mAP@[0.5:0.95]). The comparisons between three lower levels of degradation and two higher levels of degradation given in Figs. 4.3 and 4.4 show that augmenting model training with degraded images provides a more profound improvement as image degradation increases.

For the Gaussian noise experiment, the critical point for reducing performance occurs as early as $\sigma = 0.06$. At this point, even with augmentation, the mAP has dropped by 0.02. For the pixel dropout, mAP remains steady into pixel losses of about 15-20%, with augmented models even improving results through pixel losses of 60%. The augmented experiment for pixel loss maintained a mAP difference from

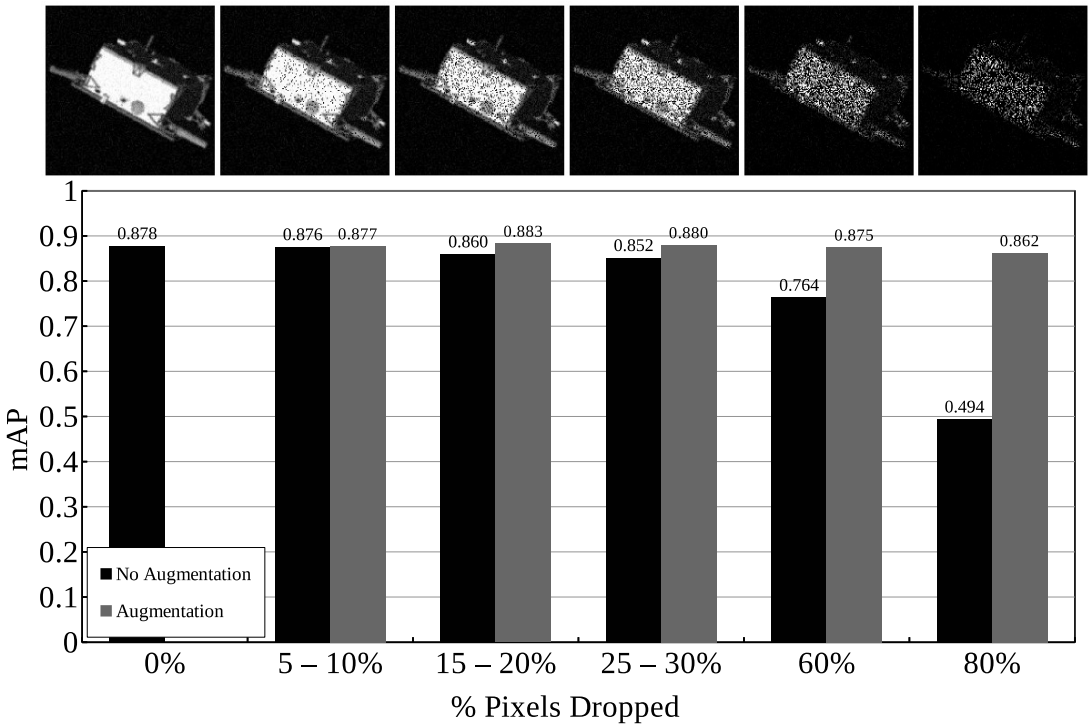


(a)

Figure 4.3: Comparisons of Gaussian noise degradation effects training with and without augmentation. The increase in degradation above 20% shows that training on augmented images significantly improves mAP compared to lower levels of degradation. Although lower levels of Gaussian noise are almost imperceptible in the images at the scaling factor of 0.09, the model performance is lowered by mAP of 0.12, and 0.05 for non-augmented and augmented training, respectively. ©2021 IEEE.

baseline of about 0.01 even after a pixel loss of 80%.

In the Gaussian noise experiment, augmentation improves results as compared to the baseline; furthermore, the difference in mAP improvement increases by about 0.01-0.02 for each σ factor, or a gap growth rate of 13%. For the pixel dropout experiment, the augmentation improves results as well, with the difference in mAP improvement increasing as more pixels are dropped, by a gap growth rate of about 7%.



(a)

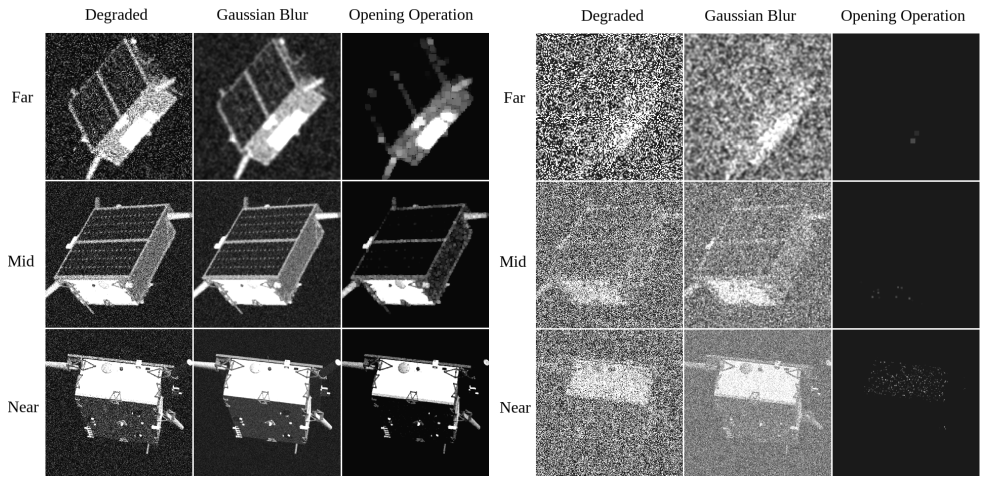
Figure 4.4: Comparisons of dropout degradation effects training with and without augmentation. The trend of augmentation improving results more significantly for higher levels of degradation continues with the pixel dropout experiment. With pixel losses of 80%, the augmented model only has a drop of 0.016 mAP. ©2021 IEEE.

4.2.3 Degradation Experiments

In order to explore the ability of the model to overcome image degradation in the object detection task, thirteen experiments for each size stratification were performed, including various image processing and training augmentation settings. Due to the size of the images and the use of the default anchor scales (the largest of which is 512 pixels), it is possible that the large scale images do not have the optimal localization and therefore lower mAP. However, as results are evaluated within scaling categories, the comparisons made amongst the thirteen experimental regimens and through each level of degradation should not be affected.

Gaussian Noise

For the Gaussian noise experiments, six levels of degradation were selected for performance comparison. These levels are multiples of the standard deviation of pixel value, which range from 0 to 255, and result in Gaussian white noise which affects the pixels of the image by standard deviation scale factor (the σ factor on the left of the tables) times the maximum value range. The six scaling factors range from zero (or no added noise) to 0.5 (or Gaussian noise with a deviation of 127) in increments of 0.1, examples of which can be seen in Fig. 4.5 for scaling at 0.10 and 0.50. The various settings for the experiments include whether or not training used augmented images, what level of degradation were used for those training images, whether no image processing (NP) was used after training, whether Gaussian filtering (GP) or opening operation (OP) were used after training during detection, and whether Gaussian filtering or opening operation were performed prior to training to further augment the images and during detection (GA and OA, respectively.) For the Gaussian filtering, the kernel size used was square 5x5, and the kernel for the opening operation was also square 5x5. In Tables 4.1, 4.2, and 4.3, the experiments with no image processing only perform well when there is no added noise, which is expected. In Table 4.1, Gaussian filter pre-processing and a similar experiment without image processing performed better than the baseline for testing without added noise. In all cases, opening operation performed worse than the other experiments. For most other experiments, the best results were for models trained with images augmented with noise at $\sigma = 0.30$ or $\sigma = 0.40$, which were then pre-processed with a Gaussian filter prior to training. Training on noisy images with $\sigma = 0.40$ seemed to be more broadly applicable for training improvement compared to $\sigma = 0.30$.



(a) (b)

Figure 4.5: Left panel (4.5(a)): shows images at three scales with the Gaussian noise deviation scaling at 0.10. Right panel (4.5(b)): shows the same images with the noise deviation scaling at 0.50. As image size increases, both Gaussian filtering and opening operation result in clearer images for the square 5x5 kernel. Significant artifacts of the morphological operation occur for medium and small images, while the large image remains relatively undistorted. For 4.5(a), the lower level of noise makes the opening operation a viable pre-processing technique, however, for 4.5(b), the opening operation makes an already difficult to distinguish image almost completely imperceptible.

Table 4.1: Gaussian Noise Degradation (Large) - mAP

	No Augmentation			Augment: $\sigma = 0.30$			Augment: $\sigma = 0.40$			Augment: $\sigma = 0.30$		Augment: $\sigma = 0.40$	
	NP	GP	OP	NP	GP	OP	NP	GP	OP	Aug: Gs.	Aug: Op.	Aug: Gs.	Aug: Op.
0σ	0.870	0.866	0.803	**0.871**	**0.871**	0.851	0.834	0.813	0.776	0.870	0.860	0.867	0.859
0.1σ	0.567	0.626	0.449	0.707	0.731	0.479	0.617	0.637	0.380	0.763	0.683	**0.791**	0.721
0.2σ	0.326	0.493	0.244	0.568	0.611	0.273	0.463	0.511	0.181	**0.692**	0.555	0.686	0.583
0.3σ	0.124	0.355	0.109	0.548	0.473	0.133	0.365	0.403	0.055	**0.619**	0.399	0.599	0.451
0.4σ	0.022	0.211	0.058	0.055	0.324	0.062	0.361	0.286	0.018	0.535	0.301	**0.537**	0.341
0.5σ	0.001	0.089	0.022	0.000	0.232	0.024	0.125	0.167	0.006	0.408	0.233	**0.437**	0.264

Table 4.2: Gaussian Noise Degradation (Medium) - mAP

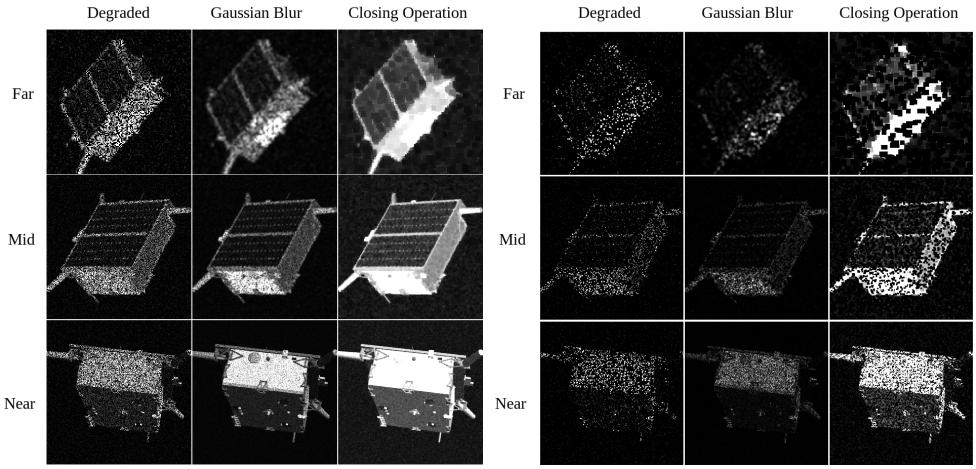
	No Augmentation			Augment: $\sigma = 0.30$			Augment: $\sigma = 0.40$			Augment: $\sigma = 0.30$		Augment: $\sigma = 0.40$	
	NP	GP	OP	NP	GP	OP	NP	GP	OP	Aug: Gs.	Aug: Op.	Aug: Gs.	Aug: Op.
0σ	**0.877**	0.874	0.816	0.857	0.861	0.834	0.844	0.833	0.806	0.872	0.836	0.874	0.841
0.1σ	0.717	0.752	0.568	0.801	0.789	0.602	0.734	0.734	0.555	0.833	0.793	**0.835**	0.788
0.2σ	0.589	0.675	0.382	0.732	0.708	0.426	0.683	0.666	0.324	0.788	0.690	**0.804**	0.719
0.3σ	0.317	0.532	0.183	0.688	0.635	0.249	0.597	0.594	0.144	**0.748**	0.623	0.740	0.626
0.4σ	0.080	0.415	0.063	0.241	0.547	0.115	0.532	0.490	0.048	0.664	0.493	**0.674**	0.540
0.5σ	0.010	0.279	0.008	0.003	0.415	0.027	0.177	0.435	0.008	0.539	0.374	**0.560**	0.453

Table 4.3: Gaussian Noise Degradation (Small) - mAP

	No Augmentation			Augment: $\sigma = 0.30$			Augment: $\sigma = 0.40$			Augment: $\sigma = 0.30$		Augment: $\sigma = 0.40$	
	NP	GP	OP	NP	GP	OP	NP	GP	OP	Aug: Gs.	Aug: Op.	Aug: Gs.	Aug: Op.
0σ	0.883	0.892	0.844	0.888	0.894	0.865	0.875	0.890	0.859	**0.899**	0.866	0.896	0.877
0.1σ	0.820	0.856	0.604	0.869	0.875	0.664	0.859	0.862	0.554	**0.879**	0.833	0.867	0.842
0.2σ	0.715	0.799	0.251	0.851	0.833	0.357	0.815	0.834	0.237	0.817	0.762	**0.853**	0.729
0.3σ	0.391	0.649	0.049	0.779	0.797	0.095	0.777	0.754	0.030	0.783	0.579	**0.803**	0.603
0.4σ	0.030	0.459	0.002	0.231	0.709	0.009	0.719	0.670	0.002	0.644	0.377	**0.721**	0.397
0.5σ	0.000	0.277	0.000	0.004	0.548	0.000	0.419	**0.580**	0.000	0.395	0.098	0.568	0.191

Pixel Dropout

For the pixel dropout experiments, six levels of degradation were utilized for evaluation, with pixel losses of 0% as the initial experiment without pixel loss, and pixel losses of 50%-90% in 10% increments for the remaining experiments. Fig. 4.6 shows examples of pixel losses of 50% and 90% comparing the three scales show the effect of pre-processing on the images. Gaussian filter and closing operation both use square 5x5 kernels for all scales. For the tables of results, CP refers to closing operation, while other image processing labels retain the meaning from previous tables. For large images, Table 4.4 shows that the best mAP for the approaches to detection with pixel loss from 0% to 70% occur in the case supplementing training with images with pixel losses of 80%, and no pre-processing of the images. Training with pixel losses of 80% that then have the closing operation applied edged out training without the closing operation for pixel losses of 80%-90% by 0.001 and 0.023 mAP, respectively. For medium sized images, no training approach has a clear superiority, however, augmented models continue to edge out the baseline performance by 0.001-0.002 mAP. For the smaller images, the dominant training approach for the assembled experiments is supplementing training with degraded images without pre-processing, with training on degraded images processed with a Gaussian filter also a strong approach (Table 4.6.)



(a) (b)

Figure 4.6: For pixel losses of 50% (4.6(a)) and losses of 90% (4.6(b)), the degraded image with Gaussian filtering and closing operation is shown, with particular interest to be found in comparing the differences in the closing operation at different scales and pixel losses. As with the noise degraded images, as image size increases, operations result in clearer images for the square 5x5 kernel, however, artifacts of the closing operation are clearly evident for smaller images. At both extremes of pixel loss, the closing operation does qualitatively improve the visibility of the general shape of the spacecraft, however, for detection, the quantitative results are more nuanced.

Table 4.4: Pixel Dropout Degradation (Large) - mAP

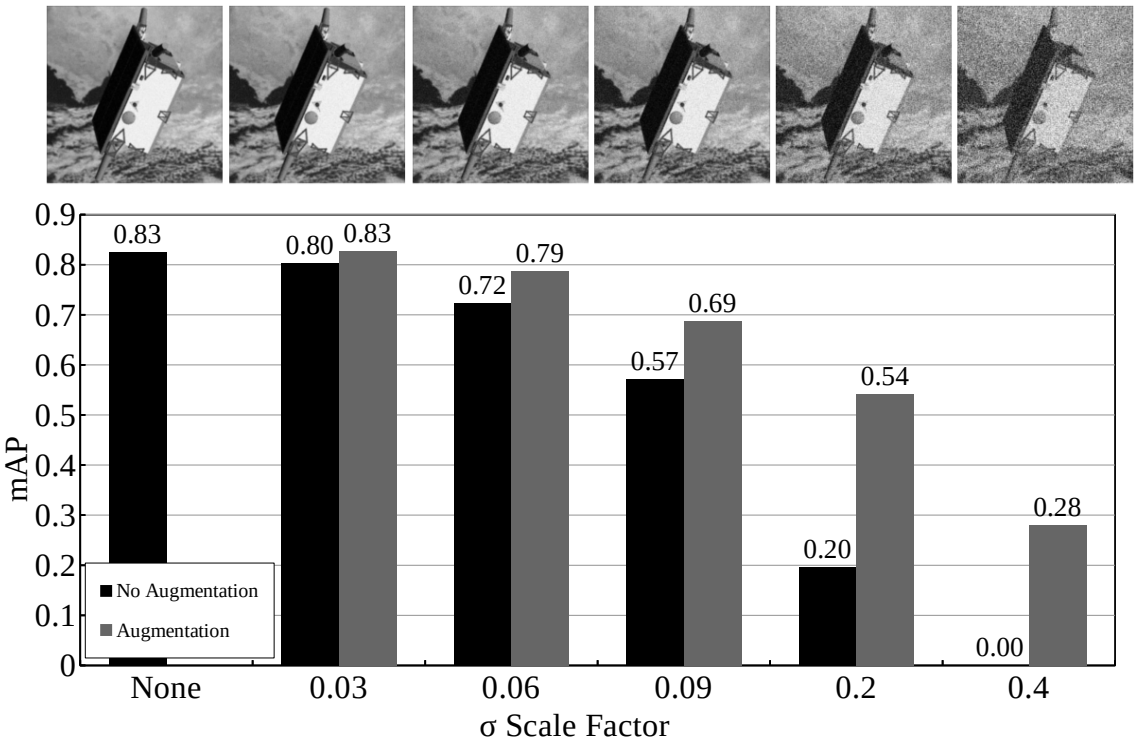
	No Augmentation			Aug: *loss* = 70%			Aug: *loss* = 80%			Aug: *loss* = 70%		Aug: *loss* = 80%	
	NP	GP	CP	NP	GP	CP	NP	GP	CP	Aug: Gs.	Aug: Cl.	Aug: Gs.	Aug: Cl.
0%	0.870	0.866	0.746	0.872	0.860	0.839	**0.882**	0.861	0.826	0.879	0.872	0.881	0.867
50%	0.680	0.686	0.709	0.884	0.862	0.835	**0.893**	0.878	0.830	0.867	0.865	0.869	0.866
60%	0.599	0.607	0.682	0.873	0.860	0.825	**0.887**	0.878	0.824	0.851	0.856	0.869	0.864
70%	0.475	0.542	0.662	0.859	0.819	0.819	**0.887**	0.876	0.833	0.811	0.869	0.854	0.864
80%	0.323	0.450	0.571	0.795	0.768	0.844	0.870	0.829	0.844	0.723	0.862	0.789	**0.871**
90%	0.045	0.247	0.337	0.591	0.616	0.728	0.769	0.768	0.750	0.555	0.725	0.614	**0.792**

Table 4.5: Pixel Dropout Degradation (Medium) - mAP

	No Augmentation			Aug: *loss* = 70%			Aug: *loss* = 80%			Aug: *loss* = 70%		Aug: *loss* = 80%	
	NP	GP	CP	NP	GP	CP	NP	GP	CP	Aug: Gs.	Aug: Cl.	Aug: Gs.	Aug: Cl.
0%	0.877	0.874	0.818	0.874	0.870	0.827	0.866	0.867	0.840	**0.878**	0.846	0.875	0.845
50%	0.752	0.785	0.813	0.859	0.848	0.838	**0.878**	0.856	0.830	0.859	0.851	0.868	0.855
60%	0.718	0.738	0.784	0.842	0.831	0.827	0.863	0.842	0.831	**0.865**	0.855	0.856	0.861
70%	0.642	0.670	0.736	0.832	0.812	0.831	0.831	0.842	0.811	0.841	0.847	**0.861**	0.849
80%	0.539	0.563	0.693	0.796	0.790	0.808	**0.835**	0.824	0.830	0.782	**0.835**	0.816	**0.835**
90%	0.296	0.409	0.469	0.704	0.725	0.741	**0.799**	0.780	0.779	0.724	0.756	0.750	0.791

Table 4.6: Pixel Dropout Degradation (Small) - mAP

	No Augmentation			Aug: *loss* = 70%			Aug: *loss* = 80%			Aug: *loss* = 70%		Aug: *loss* = 80%	
	NP	GP	CP	NP	GP	CP	NP	GP	CP	Aug: Gs.	Aug: Cl.	Aug: Gs.	Aug: Cl.
0%	0.883	0.892	0.862	0.891	**0.908**	0.871	0.889	0.899	0.864	0.888	0.883	0.907	0.886
50%	0.771	0.843	0.808	**0.913**	0.890	0.864	0.907	0.885	0.862	0.882	0.882	0.905	0.874
60%	0.733	0.802	0.812	0.899	0.891	0.869	**0.910**	0.875	0.858	0.888	0.873	0.899	0.880
70%	0.648	0.765	0.788	0.889	0.868	0.843	**0.895**	0.867	0.850	0.859	0.866	0.878	0.873
80%	0.489	0.661	0.737	0.865	0.855	0.821	**0.867**	0.849	0.810	0.834	0.852	0.865	0.863
90%	0.141	0.454	0.412	0.757	0.757	0.705	0.803	0.784	0.752	0.736	0.710	**0.817**	0.758



(a)

Figure 4.7: Comparisons of Gaussian noise degradation effects training with and without augmentation, in the complex background case. Slight levels of degradation can be compensated for through training augmentation as in the black background experiments. Models performed unsatisfactorily at levels of degradation $\sigma = 0.06$ and above.

Complex Background Experiments

For the complex background experiments, pixel dropout hampered performance more than Gaussian noise in some lower degradation scenarios, but Gaussian noise proved to be the more pernicious threat. For the Gaussian noise experiments (Fig. 4.7), again a σ scale factor of 0.06 shows an early performance deterioration as in the black background experiments, with a decrease in mAP of 0.04 with the augmented model. A deeper look at the Gaussian model shows that performance across all IoU thresholds with extreme noise degradation ($\sigma = 0.4$) is completely ineffectual, but

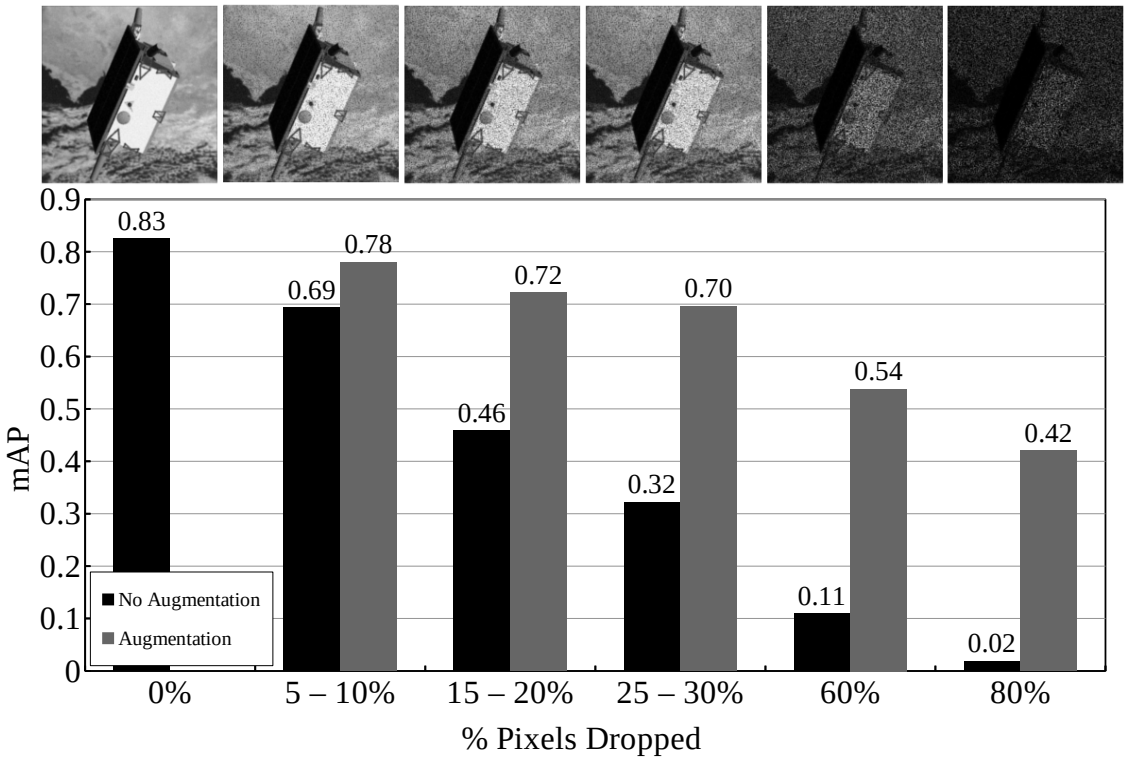
Table 4.7: Detection Performance AP per IoU with Noise Scale Factor $\sigma = 0.4$. ©2021 IEEE.

IoU	AP w/o Augmentation	AP w/ Augmentation
0.5	0.0	0.65
0.55	0.0	0.58
0.6	0.0	0.52
0.65	0.0	0.43
0.7	0.0	0.33
0.75	0.0	0.20
0.8	0.0	0.09
0.85	0.0	0.01
0.9	0.0	0.0
0.95	0.0	0.0

that augmenting training with degraded images improves performance considerably, as shown in Table 4.7. Pixel losses of 5 - 10% were enough to decrease the pixel loss model performance, even with augmentation (Fig. 4.8). This unaugmented model showed no robustness whatsoever to pixel loss, compared to the black background experiments (Fig. 4.4). At higher levels of pixel loss, this model performed better than the Gaussian noise model, but only marginally. Looking at the average precision results across all IoU thresholds for the model tested with extreme pixel loss (80%), shown in Table 4.8, it can be concluded that with relaxed standards for IoU, pixel loss is a recoverable scenario for space satellite detection even in some extreme cases. Although the noise affected model performed better than the pixel loss model at lower degradation factors, all complex background experiment models performed worse than the black background models.

4.2.4 Effects of AGWN on Detection

Evaluating the results of both the black background and complex background experiments, our expectations of decreasing performance in the face of greater image degradation was confirmed, but a more detailed explanation is necessary, particularly



(a)

Figure 4.8: Comparisons of dropout degradation effects training with and without augmentation in the complex background case. Augmentation improved results more significantly for higher levels of degradation, however, the model failed to achieve the same high level of performance as with the black background experiments. With 80% of the pixels dropped from the image, the augmented model decreased in performance by 0.44 mAP as compared to the black background model. ©2021 IEEE.

as it relates to the Additive Gaussian white noise, AGWN. The augmentation library [36] was used to degrade training images with additive Gaussian white noise. The AGWN degradation was performed by scaling the monochrome pixel range of the image (255 possible values of gray) to be between 0 and 255 by a fractional σ factor, $0 \leq \sigma \leq 1$, and setting this as the deviation of the (zero mean) normal distribution that the noise was generated from. Then each pixel in the image had noise added to it, drawn from the described distribution. The effect is, as the σ factor increases, the overall noise energy of the image increases as more added noise is drawn from higher gray values. To understand the effect of the increased deviation range (in-

Table 4.8: Detection Performance AP per IoU with Pixel Drop Percentage 80%. ©2021 IEEE.

IoU	**AP w/o Augmentation**	**AP w/ Augmentation**
0.5	0.05	0.77
0.55	0.05	0.76
0.6	0.03	0.67
0.65	0.02	0.63
0.7	0.02	0.55
0.75	0.0	0.44
0.8	0.0	0.27
0.85	0.0	0.11
0.9	0.0	0.0
0.95	0.0	0.0

creased value range of added noise) on detection and localization, we must recognize the noise inherent in the image, as well as the added noise, and how these conceptually contribute to difficulty in these tasks. In all images, noise is present due to the circumstances of the image creation, for example, emulated shot noise. The complex background images have confounding extraneous pixels, namely the Earth image, which we can call background noise. As convolutional networks are designed to learn edge detection filtering, detectable edges are paramount in successful detection and localization. As can be expected, the Earth background can disturb or conceal the edges of the spacecraft. But as AGWN increases in energy across the image (as the σ factor increases and higher value pixels are added randomly to the image), contrast of the image decreases, making edges harder to detect. This seems to have a multiplicative effect when combined with background noise, but augmenting training with degraded samples tampers this effect, even as noise increases. Contrasting the complex background noise experiments with the black background noise experiments, the effect of AGWN on detection and localization in the augmented models has a nearly linear effect. In Fig. 4.3, the augmented model (AGWN affected images included in training) seems to show a drop of about 0.4 mAP for σ factor difference of 0.03, 0.8 mAP across an increased σ factor of about 0.10, and a drop of about 0.12 mAP across

an increased σ factor of 0.2. In Fig 4.7, a σ factor difference of 0.03 results in a mAP decrease of 0.1, an increase of σ by about 0.1 results in a mAP decrease of 0.15, and an increase of σ by 0.2 results in a mAP decrease of 0.26. Between the two augmented experiments, increases in noise scaled by σ result in nearly linear decreases in mAP, up to our limited view of $\sigma = 0.4$.

Chapter 5

Deep Learning and Statistical Fusion Model for RFI detection

5.1 Methodology

5.1.1 Datasets: Training, Validation, and Testing

The detection of RFI through the combined use of time-frequency plot analysis and image oriented deep learning algorithm required a robust and test-friendly dataset for evaluating performance improvements. To that end, a simple method for generating synthetic data was developed, based on a qualitative examination of real world filterbank data. This synthetic data was used in training the models, validating their performance, and for the final model testing. By creating a synthetic set of data, the ground truth labeling is controlled for better accuracy in performance evaluation. The simulated channelized voltage plots generated were based on 8-bit filterbank data – each pixel may have one of 255 different values – and the data was partitioned into plots of 1024 channels by 1024 time samples. These synthetic images are created by filling the selected channel and time samples with Gaussian noise, which is then randomly perturbed by anywhere from one to thirty channelwise (narrowband) fake

RFI signals. These signals can have the following features: total channel or limited time samples, periodic or random, bleeding to nearby channels or single channel only. Also, a small number of broadband signals (up to three) were possible, which may or may not cross all channels. Using this method, 5,000 training images were generated, as well as 200 validation images, and 600 testing images. Two models were trained on real data, and in those cases the data was labeled using SumThreshold.

Real data

For two of the U-Net models, real data was used for training. Figure 5.1 gives a comparison of the synthetic and real data, which prominently displays channelwise RFI. Data from the Greenbank Telescope as recorded by the Spigot system [40], was used for two sources of real data. The first dataset, composed of 1024 8-bit channels (covering approximately 50 MHz), was split into groups of 1024 time samples, each time sample representing 81.92 μs, and converted into 50 time-frequency plot images. These images were inspected to ensure RFI was present for useful training. The second dataset was similar, except images were 512 x 512 channels and time samples, respectively. These two datasets were used to train the models *FB1024* and *FB512* referenced in table 5.2, 5.3, 5.4, and 5.5.

The differences between our data and that of [30] should be mentioned; the Bleien Observatory data is time-ordered data (TOD) over the span of a 24 hour period, while the Greenbank Telescope data (GBT) is TOD over the span of 84 milliseconds (per image). Due to this difference in sample data length, The characterization of the data from [30] is over multiple hours (with conditions changing from hour to hour, but similar RFI profiles from day to day), specifically for the detection of hydrogen in the 21cm band. This is contrasted with the GBT data, which is focused on the detection of pulsars, at such short time lengths unlikely to contain observable astronomical signals (with the exception of extremely short pulsar signals), and total recorded

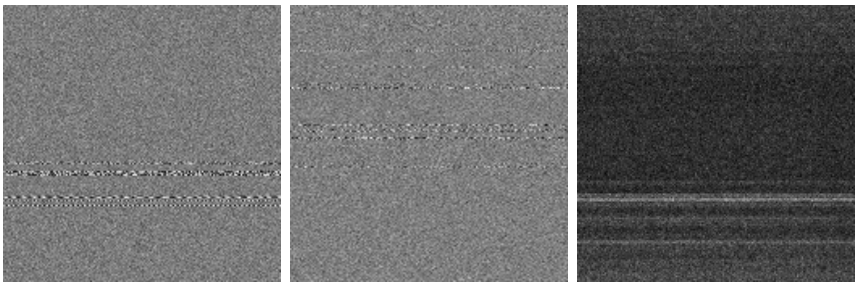
data representing at most a few minutes. Furthermore, the GBT data can essentially be represented as channelized voltages of a Gaussian noise signal perturbed with transient bursts of RFI, whereas the Bleien Observatory data is drift-scan spectrogram data which is affected by atmosphere and instrument noise, along with elevation dependent signal effects. These are emulated in the synthetic data, as well. The U-Net models, however, make no distinction between the two datatypes in flagging RFI (as the U-Net architecture is only concerned with separating foreground pixels from background pixels, regardless of time span representation, time-series frequency representation, channel frequency, or any other astronomical variable.)

5.1.2 Preprocessing

In developing our models, the question of how to best process the time-frequency plots for input into the U-Net models was explored. The intuition being that increasing the contrast between the background and RFI class would lead to better labeling, our efforts focused on choosing the best method for increasing this contrast before input. Early experiments without preprocessing beyond normalization showed that the model spent much of the early training cycles learning how to increase contrast. By exploring this preprocessing space, the model can focus on performance improvements beyond the image processing level. Comparisons of performance using image rescaling, histogram equalization, sigmoid contrast adjustment, and combinations of sigmoid contrast adjustment and histogram equalization were examined.

5.1.3 U-Net Model Design and Training

The initial U-Net model is based on the default architecture provided in [30], with three layers and 64 root features; the cost function is cross-entropy (using a pixel-wise softmax function), with L2 regularization (0.001). The performance of the model was



(a) (b) (c)

Figure 5.1: All images are 172x172 channels and time samples respectively. (a) Synthetically generated data. (b) Real filterbank data. (c) A different set of real filterbank data.

judged based on the Dice coefficient score, where the distance metric, d defined as:

$$d = 1 - \frac{2|P \cap M|}{|P + M|}, \tag{5.1}$$

where P are the predicted pixels and M is the labeled RFI mask. Although using the Dice coefficient as a cost function was possible, early investigation found model learning was very difficult and ultimately too erratic to use directly.

Thresholding

In order to maintain a consistent performance metric, a proper threshold for RFI labeling needed to be applied to the U-Net model output. Since the optimal threshold level was highly dependent on the qualities of each image, Otsu's method [41] was used to automatically select the threshold level. Although this method was used for U-Net performance evaluation, it was not used during the fusion experiments in order to take advantage of the pixelwise softmax values as inputs to the fusion model.

Hyperparameter Search

Our experiments were composed of only the background class and the RFI class, which allowed for some experimentation with various weights and ratios of weights

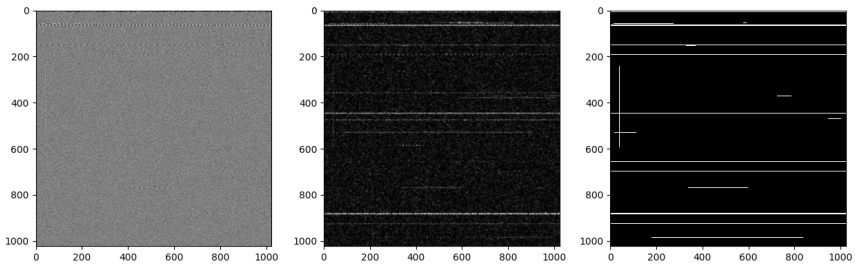
between the classes for improving detection performance. Examined were the default equal weighted models and also giving greater weight to the RFI class by two methods — decreasing the background class by half to 0.5, and doubling the RFI class weight to two. The U-Net models were trained for 20 epochs with 100 iterations per epoch. It was determined empirically that after 20 epochs, model improvements were minor and so further tuning of this hyperparameter was unnecessary. Included in the hyperparameter search was selecting the momentum optimizing rate. Briefly examined was the Adam optimizer but it was found to be difficult to train with, and so the early success with momentum guided the rest of the experiments. The learning rates and decay rates were left at default (0.2 and 0.95 respectively), while the momentum rate was altered for the experiments. Training was done on an NVIDIA Titan Xp GPU, with 12 GB of video memory. For fusion model testing, two GPU with shared memory were used to account for two U-Net models being used in parallel. In the appendix can be found examples of training predictions of model *Exp11*: the initial model before training 6.10, and after 20 epochs of training, 6.11.

5.1.4 Fusion Model Testing Protocol

Models were developed based on combining two separate U-Nets and fusing the results in various combinations with statistical tests and the SumThreshold technique for RFI flagging. To determine the best method for fusing these inputs, six U-Net models were selected to incorporate into the fusion model: two were selected based on best score, two were selected based on differing preprocessing techniques, and the two models trained on real data were selected. These were combined pairwise into fifteen U-Net inputs for the fusion experiments. During test time, statistics and SumThreshold were also run and the predictions combined with the U-Net models in four combinations defined as in table 5.1. The pixel classification threshold is 0.5 in all cases of the fusion models. For the *AVG* and *STATS* models, a comparison has been made between using

Table 5.1: Fusion model testing protocol

Model	Fusion method
AVG	$(U1 + U2 + ST + K + (SW \vee AD))/5$
ADD	$(U1 + U2) > 1 \implies 1; otherwise(U1 + U2)$
NOSTATS	$(U1 + U2 + ST)/3$
STATS	$(U1 + U2 + K + (SW \vee AD))/5$



(a) (b) (c)

Figure 5.2: (a) synthetic data before preprocessing. (b) synthetic data after preprocessing by rescaling pixel values from the 90^{th} to the 99.5^{th} percentile. (c) synthetic data ground truth RFI label mask.

the Shapiro-Wilk test and the Anderson-Darling test, but only one of these tests is used in a given fusion model, combined with the kurtosis test. Also for the *STATS* model, each input is weighted due to the division by 5. This was done since the statistic tests are hard labels while the U-Net models are soft labels; by limiting each input to 0.2 maximum voting power, statistics alone (which flag entire channels) cannot label pixels as RFI, preventing the data from being over flagged.

5.2 Experiments

5.2.1 Hyperparameter Search

Our initial experiments were focused on improving the training of the U-Net model and tuning for the best hyperparameter selections for the data. The application of RFI mitigation in [30] proved that the U-Net model was well suited to this problem as applied to the Bleien Observatory 21cm band data as well as the synthetic HIDE

& SEEK data (a useful suite of astronomical applications provided by [42]), and so applying this model to the Greenbank Telescope Observatory data would be a natural approach.

The experiment results are measured against the SumThreshold results, which is shown in tables 5.2, 5.3, 5.4, and 5.5 as the baseline. The hyperparameters which were explored include the momentum rate, class weight settings, and the preprocessing strategy.

The initial hyperparameter explored was the method of input preprocessing. Using total variation minimization (Chambolle) would not converge, and wavelet denoising resulted in a unacceptable loss of information, which ultimately did not converge, either. This initial approach of denoising was abandoned in exchange for a focus on contrast enhancement. This proved to be much more successful, and it seems intuitive in retrospect, to allow the model to determine what is noise. The candidate models for the fusion experiments were all trained using some contrast enhancing preprocessing technique, with the exception of *FB512* which varied greatly in brightness in both background and RFI. The labeling of this data with SumThreshold led to a great deal of poor labeling of lower level broadband RFI, making it difficult for the model to converge. An effective normalization method was not found, and solo model performance was very poor, although in the fusion experiments this did not seem to affect the final score significantly, showing that the fusion model is robust to a single model failure (analogous to recessive gene expression in biology).

The default setting for the momentum rate of 0.2 did not seem to produce the best training scenario; training was slower, and produced only modest scores. Increasing the momentum beyond 0.99 often lead to the model not converging, and by decreasing the momentum to between 0.5 and 0.7 tended to converge with better results. The exception was with *FB1024* and *FB512*, which did not converge at higher momentum rate.

Weighting the RFI class produced the best results when combined with the higher momentum rate. Although a limited number of experiments were done on this hyperparameter, the results seem to suggest that weighting the RFI twice that of the background class gives the proper level of emphasis on flagging for RFI and is more useful to the model than halving the background class weight.

5.2.2 U-Net Model Experiments

In table 5.2 the results of the candidate U-Net model experiments give two models that surpassed the SumThreshold baseline Dice score. Two models did not converge and resulted in a score of zero; *Exp7* combined the highest momentum with the class weight halving strategy, and *Exp15* performed histogram equalization followed by sigmoid contrast adjustment. The worst performing model that did have some positive result was the *FB512* model, which was hampered by poor ground truth flagging, difficulty with preprocessing, and poor carryover to the validation dataset.

5.2.3 Fusion Experiments

Four different fusion approaches were examined in the fusion experiments. Of these, two are non-statistical methods, *ADD* and *NOSTATS* computed as described in table 5.1. The *ADD* method is a simple addition of the two U-Net models, clipped to unity. The *NOSTATS* method combines the two U-Net model outputs along with the SumThreshold output, and averages them. Overall, the *NOSTATS* model is the poorest performing of the fusion models. The results of the non-statistical fusion experiments are listed in table 5.5, with model *11_12* performing the best in the *ADD* model, and *09_11* performing the best in the *NOSTATS* model.

Improved fusion results came from combining statistical information, along with U-Net and SumThreshold. The best results were had by combining all 5 outputs and taking the average result, listed in tables 5.1 and 5.3 as *AVG*. To examine the effect

Table 5.2: U-Net model results (validation)

Model	Mom. Rate	Weights	Preprocess	Dice Score
Exp4	0.20	{1,1}	rescale	0.6111
Exp5	0.20	{0.5,1}	rescale	0.5911
Exp6	0.99	{1,1}	rescale	0.5822
Exp7	0.99	{0.5,1}	rescale	0
Exp8	0.50	{1,1}	rescale	0.5992
Exp9	0.50	{1,2}	rescale	**0.6997**
Exp10	0.70	{1,1}	rescale	0.5860
Exp11	0.70	{1,2}	rescale	**0.7471**
Exp12	0.70	{1,2}	eq hist.	0.5876
Exp13	0.70	{1,2}	sigmoid cor.	0.3691
Exp14	0.70	{1,2}	sig./hist.	0.6262
Exp15	0.70	{1,2}	hist./sig.	0
FB512	0.20	{1,2}	norm	0.0206
FB1024	0.20	{1,2}	rescale	0.4941
ST (baseline)	-	-	-	0.6949

of combining the Anderson-Darling test with the kurtosis test (over Shapiro-Wilk combined with kurtosis), two sets of fusion models were tested with these setups, as described in table 5.1. The results of these experiments show that these two statistical combinations give similar results, with a slight edge given to the Anderson-Darling test combined with kurtosis. Listed in table 5.4 are the results of the *STATS* models; this set of experiments was the second best approach with a Dice score of 0.7782 for the Anderson-Darling model, and 0.7716 for the Shapiro-Wilk model, as evaluated by the validation dataset.

5.2.4 Final Experimental Results

The U-Net model pairs were combined with the SumThreshold, Anderson-Darling test, and kurtosis test (*AVG* model) and run on the 600 image test set. Figure 5.3 shows a boxplot of the top three fusion models and the baseline. All three fusion models outperformed the baseline; of all the experiments, only one performed worse than the baseline, the *FB512_FB1024* model, with a median score of 0.6611

Table 5.3: Average Fusion Model Comparison (validation)

U-Net Pair	**AVG (AD+K)**	**AVG (SW+K)**
14_FB1024	0.7954	0.7903
12_14	0.7968	0.7884
14_FB512	0.7895	0.7867
12_FB1024	0.7599	0.7660
12_FB512	0.7478	0.7556
11_FB1024	0.8072	0.8026
11_14	**0.8283**	0.8155
11_12	0.8223	**0.8179**
11_FB512	0.8074	0.8053
09_FB1024	0.7780	0.7796
09_14	0.8141	0.8075
09_12	0.8002	0.8028
09_11	0.8133	0.8090
09_FB512	0.7730	0.7784
FB512_FB1024	0.6442	0.6557
ST (baseline)	0.6949	

Table 5.4: Statistical Fusion Model Comparison (validation)

U-Net Pair	**STATS (AD+K)**	**STATS (SW+K)**
14_FB1024	0.7158	0.7123
12_14	0.7191	0.7100
14_FB512	0.7005	0.6958
12_FB1024	0.6345	0.6364
12_FB512	0.5951	0.5962
11_FB1024	0.7466	0.7457
11_14	**0.7782**	**0.7716**
11_12	0.7690	0.7689
11_FB512	0.7465	0.7476
09_FB1024	0.6956	0.6962
09_14	0.7602	0.7532
09_12	0.7383	0.7370
09_11	0.7699	0.7674
09_FB512	0.6904	0.6897
FB512_FB1024	0.0044	0.0044
ST (baseline)	0.6949	

Table 5.5: Non-statistical Fusion Results (Validation)

U-Net Pair	ADD	NOSTATS
14_FB1024	0.6759	0.6040
12_14	0.6396	0.6000
14_FB512	0.6600	0.5850
12_FB1024	0.6537	0.5381
12_FB512	0.6141	0.5019
11_FB1024	0.7364	0.6298
11_14	0.7062	0.6503
11_12	**0.7569**	0.6403
11_FB512	0.7456	0.6275
09_FB1024	0.7026	0.5947
09_14	0.7024	0.6325
09_12	0.7408	0.6150
09_11	0.7274	**0.7347**
09_FB512	0.7016	0.5862
FB512_FB1024	0.0045	0.0043
ST (baseline)	0.6949	

and mean of 0.6332 (baseline 0.7041 and 0.6969, respectively). Our proposed and best performing model from experiment U-Net pair *11_14* fused with SumThreshold, Anderson-Darling test, and kurtosis test, surpasses the baseline with a mean Dice score of 0.8103 and median Dice score of 0.8395. Examples of the RFI detection using this model can be found in the Appendix, along *ADD*, *NOSTATS*, and *STATS* versions of this fusion model.

5.2.5 Discussion

Although the fusion models across nearly all *AVG* experiments outperformed the baseline SumThreshold method, the top three models in the final results have traits in common that seem to promote the best RFI flagging. Weighting the RFI class higher than the background class is found in each U-Net model, which tends to increase rates of RFI flagging overall. Each model uses a contrast enhancing preprocessing technique, which eases the model training and relieves the model of having to

learn such functions to perform the more important task of RFI flagging. But along with this, it is interesting that the top performing model pairs combine two separate methods for contrast enhancement. It is possible that the alternative preprocessing methods help the U-Net model "see" a different perspective, thereby contributing more useful information to the final fusion output. The strength of the *AVG* fusion model is in the array of different views voting and contributing to the final outcome, which improves the RFI flagging beyond SumThreshold alone.

Normality Test Comparison

Although [25] suggested the use of Anderson-Darling combined with kurtosis to flag RFI, the Shapiro-Wilk test in this case seemed to work combined with kurtosis nearly as well as Anderson-Darling. Considerations of sample sizes and signal source given in [25] had less impact in our experiments. The sample size for most experiments were 2^{10}, much lower than the sample size limit for Shapiro-Wilk of at most 2000 (the threshold at which the Shapiro-Wilk test needs to be performed in separate sample blocks.) The signals generated for our synthetic data were designed with random elements of periodization, which would seem to make it unlikely that the blind spot of Kurtosis described in [25] for various duty cycles for different signals would impact overall detection results. It is also worth considering that the sample size overall is low, compared to the large sample sizes in [25] of 2^{16}, however, performance of the tests relative to each other remained consistent for all sample sizes.

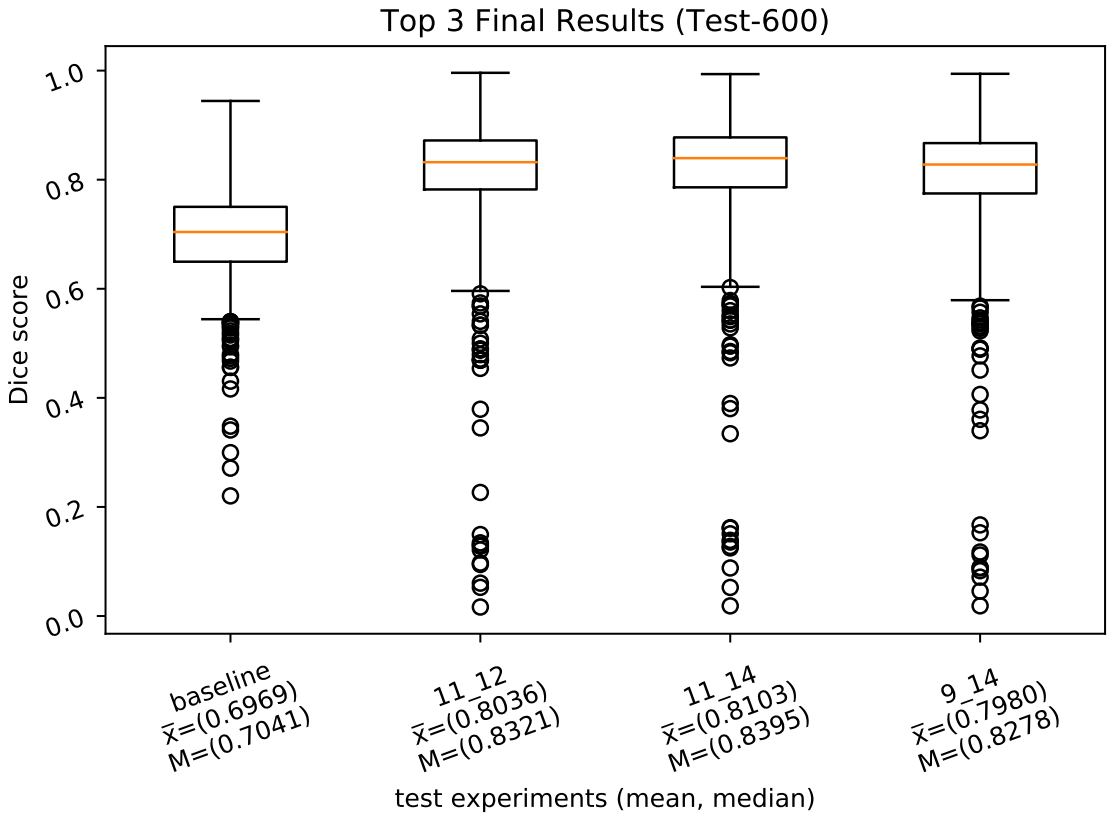


Figure 5.3: The top three fusion models and the SumThreshold baseline, evaluated on the test set of 600 images. The mean and median scores of the test set are listed below.

Chapter 6

Summary, Conclusion, and Further Work

6.1 Satellite Detection

The first thesis topic explored the evaluation of an off-the-shelf implementation of the Mask R-CNN architecture under Gaussian noise and pixel loss degradation using the SPEED dataset. Also explored was the use of alternative ResNet backbone architectures, and the use of training augmentation to improve the robustness of the model to noise and pixel loss degradation factors. While there are some aspects of image processing that can be combined with training augmentation to improve results, these few simple techniques explored are useful only in very niché circumstances such as the use of the closing operation on images with pixel loss and an appropriately selected kernel size. In all cases, supplementing training on degraded images improved results significantly, in some cases improving better than the baseline results. Also, the model was able to detect and localize properly on images degraded by pixel loss; the model continued to perform close to baseline in conditions even up to 80% pixel loss for the black background experiments. Gaussian noise proved to be a significant

problem for even lower levels of degradation, although training augmentation does seem to improve performance slightly averaged across all IoU thresholds. Models trained on complex images performed worse than those trained on black background images, with models trained on images degraded by pixel loss lacking robustness at higher IoU thresholds, even with training augmentation. An off-the-shelf Mask R-CNN implementation would have difficulty continuing the mission in the face of possible radiation noise, but with model augmentation and a relaxed IoU threshold, it may be able to recover and sustain the mission in the face of significant pixel loss.

6.2 RFI Detection

The use of the U-Net model was examined for the application of RFI detection and flagging in combination with statistical tests and thresholding techniques as a fusion model. Through hyperparameter search and variations on output fusion methods, it has been determined that averaging the outputs of the Kurtosis and Anderson-Darling statistical tests with the SumThreshold thresholding technique and the time-frequency plot flagging from a pair of U-Net models with alternate image preprocessing methods produces a fusion model which surpasses the use of SumThreshold alone. This fusion model would be most applicable to the flagging of offline data, but with increased computational resources and hardware designed for this method, real time flagging using this model is feasible. Future research into the hardware implementation of combined methods of deep learning and statistical tests for the flagging of RFI are certainly warranted.

6.3 Further Work

In the satellite detection chapters, Mask R-CNN was the focus of the experiments, with variations on the backbone architectures; this can be expanded to other modern

model architectures also oriented toward instance segmentation. Improvements to the experiments in image degradation can also be explored, including increased dissection of the pixel loss and Gaussian white noise factors for a closer look at the limits of the models, as well as learned variations in the kernel sizes for the application of traditional morphological operations. For the RFI detection chapters, the next stage to this work is the application of recent models which use a modified U-Net, some examples including the AC-UNet [31] and the inclusion of squeeze-excitation layers [43] into the model for an attention based approach to RFI detection.

Bibliography

[1] Dung, H. A., Chen, B., and Chin, T.-J., "A spacecraft dataset for detection, segmentation and parts recognition," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2012–2019.

[2] Kisantal, M., Sharma, S., Park, T. H., Izzo, D., Märtens, M., and D'Amico, S., "Satellite pose estimation challenge: Dataset, competition design, and results," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 56, No. 5, 2020, pp. 4083–4098.

[3] Murray, G., Bourlai, T., and Spolaor, M., "Mask R-CNN: Detection Performance on SPEED Spacecraft With Image Degradation," *2021 IEEE International Conference on Big Data (Big Data)*, IEEE, 2021, pp. 4183–4190.

[4] He, K., Gkioxari, G., Dollar, P., and Girshick, R., "Mask R-CNN," *2017 IEEE International Conference on Computer Vision (ICCV)*, IEEE, oct 2017.

[5] Ren, S., He, K., Girshick, R., and Sun, J., "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *Advances in Neural Information Processing Systems 28*, edited by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Curran Associates, Inc., 2015, pp. 91–99.

[6] Peng, X., QIN, H., HU, Z., CAI, B., LIANG, J., and OU, H., "Gas plume detection in infrared image using mask R-CNN with attention mechanism,"

AOPC 2019: AI in Optics and Photonics, edited by J. Tanida, Y. Jiang, D. Liu, J. Greivenkamp, H. Gong, and J. Lu, SPIE, dec 2019.

[7] Singh, J. and Shekhar, S., "Road Damage Detection And Classification In Smartphone Captured Images Using Mask R-CNN," 2018.

[8] Nguyen, D., Le, T., Tran, T., Vu, H., Le, T., and Doan, H., "Hand segmentation under different viewpoints by combination of Mask R-CNN with tracking," *2018 5th Asian Conference on Defense Technology (ACDT)*, Oct 2018, pp. 14–20.

[9] Johnson, J. W., "Automatic Nucleus Segmentation with Mask-RCNN," *Advances in Intelligent Systems and Computing*, Springer International Publishing, apr 2019, pp. 399–407.

[10] Nie, S., Jiang, Z., Zhang, H., Cai, B., and Yao, Y., "Inshore Ship Detection Based on Mask R-CNN," *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, IEEE, jul 2018.

[11] Gamage, H. V. L. C., Wijesinghe, W. O. K. I. S., and Perera, I., "Instance-Based Segmentation for Boundary Detection of Neuropathic Ulcers Through Mask-RCNN," *Artificial Neural Networks and Machine Learning – ICANN 2019: Workshop and Special Sessions*, Springer International Publishing, 2019, pp. 511–522.

[12] Couteaux, V., Si-Mohamed, S., Nempont, O., Lefevre, T., Popoff, A., Pizaine, G., Villain, N., Bloch, I., Cotten, A., and Boussel, L., "Automatic knee meniscus tear detection and orientation classification with Mask-RCNN," *Diagnostic and Interventional Imaging*, Vol. 100, No. 4, apr 2019, pp. 235–242.

[13] Jaiswal, A. K., Tiwari, P., Kumar, S., Gupta, D., Khanna, A., and Rodrigues, J. J., "Identifying pneumonia in chest X-rays: A deep learning approach," *Measurement*, Vol. 145, oct 2019, pp. 511–518.

[14] Paste, A. S. and Chickerur, S., "Analysis of Instance Segmentation using Mask-RCNN," *2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, Vol. 1, July 2019, pp. 191–196.

[15] Aghdam, H. H., Heravi, E. J., and Puig, D., "Analyzing the Stability of Convolutional Neural Networks against Image Degradation," *Proceedings of the 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, SCITEPRESS - Science and and Technology Publications, 2016.

[16] Pei, Y., Huang, Y., Zou, Q., Zhang, X., and Wang, S., "Effects of Image Degradation and Degradation Removal to CNN-based Image Classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019, pp. 1–1.

[17] Tang, J. and Li, L., "An Image Filtering Method Based on Morphological Gradient Operator," *Revista de la Facultad de Ingeniería U.C.V.*, Vol. 32, No. 10, 2017, pp. 71–77.

[18] Raid, A., Khedr, W., El-Dosuky, M., and Aoud, M., "Image restoration based on morphological operations," *International Journal of Computer Science, Engineering and Information Technology (IJCSEIT)*, Vol. 4, No. 3, 2014, pp. 9–21.

[19] Agustina, I., Nasir, F., and Setiawan, A., "The Implementation of Image Smoothing to Reduce Noise using Gaussian Filter," *International Journal of Computer Applications*, Vol. 177, No. 5, nov 2017, pp. 15–19.

[20] Ku, J., Harakeh, A., and Waslander, S. L., "In Defense of Classical Image Processing: Fast Depth Completion on the CPU," *2018 15th Conference on Computer and Robot Vision (CRV)*, IEEE, may 2018.

[21] *Benchmarking Neural Network Robustness to Common Corruptions and Perturbations*, 2019.

[22] Chen, B., Cao, J., Parra, A., and Chin, T.-J., "Satellite Pose Estimation with Deep Landmark Regression and Nonlinear Pose Refinement," *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, IEEE, oct 2019.

[23] Hussain, S., Bhanu, C., Ravichandran, A., and Kondur, S., "Satellite Pose Estimation using Convolutional Neural Networks," .

[24] Gerard, K., "Segmentation-driven Satellite Pose Estimation," .

[25] Forte, G. F., Tarongí Bauza, J. M., dePau, V., Vall·llossera, M., and Camps, A., "Experimental Study on the Performance of RFI Detection Algorithms in Microwave Radiometry: Toward an Optimum Combined Test," *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 51, No. 10, Oct. 2013, pp. 4936–4944.

[26] Ronneberger, O., Fischer, P., and Brox, T., "U-Net: Convolutional Networks for Biomedical Image Segmentation - 1505.04597," .

[27] De Roo, R., Misra, S., and Ruf, C., "Sensitivity of the Kurtosis Statistic as a Detector of Pulsed Sinusoidal RFI," *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 45, No. 7, 2007.

[28] Jose Miguel Tarongi and Adriano Camps, "Radio Frequency Interference Detection and Mitigation Algorithms Based on Spectrogram Analysis," *Algorithms*, Vol. 4, No. 4, 1999, pp. 239–261.

[29] Offringa, A. R., de Bruyn, A. G., Biehl, M., Zaroubi, S., Bernardi, G., and Pandey, V. N., "Post-correlation radio frequency interference classification methods," *Monthly Notices of the Royal Astronomical Society*, Vol. 405, No. 1, June 2010, pp. 155–167.

[30] Akeret, J., Chang, C., Lucchi, A., and Refregier, A., "Radio frequency interference mitigation using deep convolutional neural networks," *Astronomy and Computing*, Vol. 18, Jan. 2017, pp. 35–39.

[31] Yan, R.-Q., Dai, C., Liu, W., Li, J.-X., Chen, S.-Y., Yu, X.-C., Zuo, S.-F., and Chen, X.-L., "Radio frequency interference detection based on the AC-UNet model," *Research in Astronomy and Astrophysics*, Vol. 21, No. 5, June 2021, pp. 119.

[32] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L., "Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs," *ICLR*, 2016.

[33] Offringa, A. et al., "The SumThreshold method: technical details," *Tech. Rep.*, 2012.

[34] Abdulla, W., "Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow," 2017.

[35] Stanford, S. A., "Apples and Oranges – A Comparison," Internet, 1995.

[36] Jung, A. B., Wada, K., Crall, J., Tanaka, S., Graving, J., Reinders, C., Yadav, S., Banerjee, J., Vecsei, G., Kraft, A., Rui, Z., Borovec, J., Vallentin, C., Zhydenko, S., Pfeiffer, K., Cook, B., Fernández, I., De Rainville, F.-M., Weng, C.-H., Ayala-Acevedo, A., Meudec, R., Laporte, M., et al., "imgaug," https://github.com/aleju/imgaug, 2020, Online; accessed 01-Feb-2020.

[37] Bradski, G., "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[38] Lefkimmiatis, S., "Universal Denoising Networks : A Novel CNN Architecture for Image Denoising," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, jun 2018.

[39] Chollet, F. et al., "Keras," https://keras.io, 2015.

[40] Kaplan, D., Escoffier, R., Lacasse, R., O'Neil, K., Ford, J., Ransom, S., Anderson, S., Cordes, J., Lazio, T., and Kulkarni, S., "The Green Bank Telescope Pulsar Spigot," *Publications of the Astronomical Society of the Pacific*, Vol. 117, No. 832, 2005, pp. 643–653.

[41] Otsu, N., "A Threshold Selection Method from Gray-Level Histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 9, No. 1, 1979, pp. 62–66.

[42] Akeret, J., Seehars, S., Chang, C., Monstein, C., Amara, A., and Refregier, A., "HIDE & SEEK: End-to-end packages to simulate and process radio survey data," *Astronomy and Computing*, Vol. 18, Jan. 2017, pp. 8–17.

[43] Hu, J., Shen, L., and Sun, G., "Squeeze-and-excitation networks," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

Appendix

```
## rfi generator script copyright 2020 Greg Murray

import matplotlib.pyplot as plt
import numpy
import sys
from skimage import data, io, filters
import random
import math

class transmitter:
    def __init__(self, cosine=True, pi_start=0, pi_inc=0.25,
   strength=[1.0,6.0], periodic=False, kind='narrow'):
        self.cosine = cosine
        self.pi_start = pi_start
        self.pi_inc = pi_inc
        self.strength = strength
        self.periodic = periodic
        self.kind = kind

    def emit(self, values, nchan, ntimes, mask=None, schan=None,
   stime=None, full=False, bleed=False, prevpival=0):
        start_channel = 0
        start_time = 0
        #start at some random channel
```

```
23          if schan:
24              start_channel = schan
25              if start_channel >= nchan:
26                  start_channel = 0
27              if start_channel < 0:
28                  start_channel = nchan-1
29          else:
30              start_channel = random.randint(0, nchan-1)
31
32          #start somewhere random in the channel
33          if stime:
34              start_time = stime
35              if start_time > ntimes:
36                  start_time = 0
37              if start_time < 0:
38                  start_time = ntimes-1
39          else:
40              start_time = random.randint(0, ntimes-1)
41
42          #if using cosine, track the current pi value
43          if prevpival:
44              pival = prevpival
45          else:
46              pival = self.pi_start
47
48          #from start, perform a transmission
49          strength = random.uniform(self.strength[0],self.strength[1])
50
51          if self.kind == 'narrow':
52              if full:
53                  rng = ntimes
54                  start_time = 0
55              else:
```

```
56                rng = ntimes - start_time
57                #cut the emission short
58                rng = rng - random.randint(0, rng)
59
60            #do an emission
61            for timebin in range(rng):
62                values[start_channel, start_time + timebin] = values
[start_channel, start_time + timebin] + (math.cos(pival) if self.
cosine else 1) * strength
63                if mask is not None:
64                    mask[start_channel, start_time + timebin] = mask
[start_channel, start_time + timebin] + 1
65                if self.periodic:
66                    pival = pival + self.pi_inc
67                else:
68                    pival = pival + self.pi_inc * random.uniform
(1.0,4.0)
69
70            #if bleed, do another emission, with possible emission
again, etc, right next to channel
71            if bleed:
72                values, mask = self.emit(values, nchan, ntimes,mask=
mask, schan=start_channel+1, full=full,bleed=random.choice([True,
False]))
73
74        if self.kind == "broad":
75            if full:
76                rng = nchan
77                start_channel = 0
78            else:
79                rng = nchan - start_channel
80                #cut it short
81                rng = rng - random.randint(0, rng)
```

```
82
83              #do an emission
84              for channel in range(rng):
85                  values[int(start_channel + channel), int(start_time)
    ] = values[int(start_channel + channel), int(start_time)] + (math
    .cos(pival) if self.cosine else 1) * strength
86                  if mask is not None:
87                      mask[int(start_channel + channel), int(
    start_time)] = mask[int(start_channel + channel), int(start_time)
    ] + 1
88                  if self.periodic and not full:
89                      pival = pival + self.pi_inc
90                      #chance to turn off periodic
91                      self.periodic = random.choice([True,False])
92                      values, mask = self.emit(values, nchan, ntimes,
    mask=mask, stime=start_time+pival, full=False, bleed=bleed,
    prevpival=pival)
93
94          return values, mask
95
96  def main():
97      import argparse
98      import os
99      import skimage
100     from skimage import io
101
102     # Parse command line arguments
103     parser = argparse.ArgumentParser(
104         description='create synthetic RFI images with masks')
105     parser.add_argument("-d", "--directory", required=True,
106                         metavar="{directory}",
107                         help="where to store the generated images")
108     parser.add_argument('-o','--outfile', required=True,
```

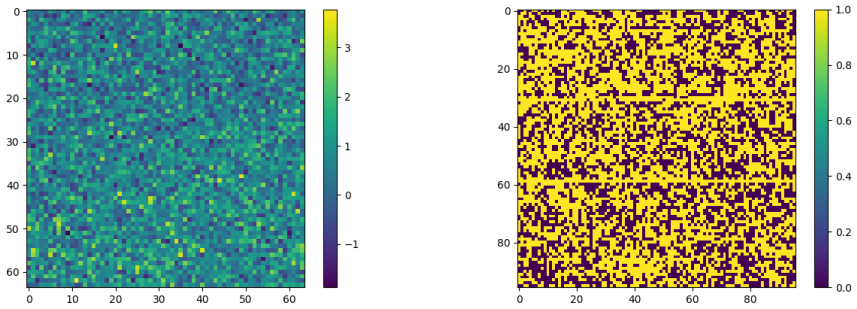
```
109                         metavar="{base filename}",
110                         help="file base name for output; will be
    appended by image index")
111     parser.add_argument('-c', '--channels', required=True, type=int,
112                         metavar="{integer}",
113                         help="the number of channels in this fake
    data.")
114     parser.add_argument('-t', '--timebins', required=True, type=int,
115                         metavar="{integer}",
116                         help="the number of samples in this fake
    data.")
117     parser.add_argument('-n', '--num', required=True, type=int,
118                         metavar="{integer}",
119                         help="the number of images to generate")
120     parser.add_argument('--example', default=False, action="
    store_true",
121                         help="whether we just want to show a quick
    example.")
122     parser.add_argument('--masks', default=False, action="store_true
    ",
123                         help="whether we want to create and store
    masks.")
124     parser.add_argument('-a','--addcount', required=False, default
    =0, type=int, metavar="{integer}",
125                         help="the number to start counting from when
     making files.")
126     args = parser.parse_args()
127
128     #setup
129     nch = args.channels
130     ntimes = args.timebins
131     mask=None
132
```

```
133     if args.example:
134         example(nch=args.channels,ntimes=args.timebins, masks=args.
    masks)
135         exit(0)
136 
137     #generate as many images as requested
138     print("the number of images to generate is {}.\n".format(args.
    num))
139     for i in range(args.num):
140         #create a new image
141         values = numpy.random.normal(0, 1, [nch, ntimes])
142         if args.masks:
143             mask = numpy.zeros((nch,ntimes))
144 
145         #setup random number of transmissions
146         narrow_count = random.randint(1,30)
147         broad_count = random.randint(0,3)
148 
149         #emit randomly that number of times in random places in the
    data
150         for j in range(narrow_count):
151             xmtr = transmitter(pi_start=random.randint(0,10), pi_inc
    =random.uniform(0.2,4.0),periodic=random.choice([True,False]))
152             values, mask = xmtr.emit(values, nch, ntimes, mask=mask,
     full=random.choice([True,False]),bleed=random.choice([True,False
    ]))
153 
154         for j in range(broad_count):
155             xmtr = transmitter(pi_start=random.randint(0,10), pi_inc
    =random.uniform(0.1,0.6),strength=[1.0,6.0], periodic=random.
    choice([False,False,True]),kind="broad")
156             values, mask = xmtr.emit(values, nch, ntimes, mask=mask,
     full=random.choice([False, False, False, True]),bleed=False)
```

```
157
158         #make the mask a boolean mask or range limited to 0 and 1
159         if args.masks:
160             mask = numpy.ma.make_mask(mask, shrink=False)
161
162         ######save the images#####
163         #setup the names
164         image_name = "{}_{}.png".format(args.outfile, i + args.
    addcount)
165         mask_name = "{}_{}_mask.png".format(args.outfile, i + args.
    addcount)
166         image_path = os.path.join(args.directory, image_name)
167         mask_path = os.path.join(args.directory, mask_name)
168
169         #numpy clip
170         #numpy.clip(values,-1,1,out=values)
171
172         #save the images
173         io.imsave(image_path, values, check_contrast=False)
174         if args.masks:
175             io.imsave(mask_path, skimage.img_as_ubyte(mask),
    check_contrast=False)
176
177
178 def example(nch=1024, ntimes=1024, masks=False):
179     #from skimage.exposure import equalize_adapthist, adjust_log,
    rescale_intensity
180     #from skimage.filters import apply_hysteresis_threshold, median
181     from skimage.restoration import denoise_tv_chambolle,
    denoise_wavelet
182     #from skimage import img_as_ubyte, img_as_float
183     #from scipy import ndimage
184     import cv2
```

```
185
186     mask=None
187     # Initialize data with random numbers
188     values = numpy.random.normal(0, 1, [nch, ntimes])
189     if masks:
190         mask = numpy.zeros((nch,ntimes))
191
192     #setup random number of transmissions
193     narrow_count = random.randint(1,30)
194     broad_count = random.randint(0,3)
195
196     #emit randomly that number of times in random places in the data
197     for i in range(narrow_count):
198         xmtr = transmitter(pi_start=random.randint(0,10), pi_inc=
    random.uniform(0.2,4.0),periodic=random.choice([True,False]))
199         values, mask = xmtr.emit(values, nch, ntimes, mask=mask,
    full=random.choice([True,False]),bleed=random.choice([True,False
    ]))
200
201     for i in range(broad_count):
202         xmtr = transmitter(pi_start=random.randint(0,10), pi_inc=
    random.uniform(0.1,0.6), periodic=False,kind="broad")
203         values, mask = xmtr.emit(values, nch, ntimes, mask=mask,
    full=random.choice([False, False, False, True]),bleed=False)
204
205     #get percentile based info
206     values_alt = cv2.normalize(values, None, alpha = 0, beta = 1,
    norm_type = cv2.NORM_MINMAX, dtype = cv2.CV_32F)
207     #values_alt = denoise_tv_chambolle(values_alt, weight=0.1,
    multichannel=False)
208     values_alt = denoise_wavelet(values_alt, multichannel=False,
    rescale_sigma=True)
209     #p_low, p_high = numpy.percentile(values_norm, (90, 99.5))
```

```
210     #values_rescale = rescale_intensity(values_norm, in_range=(p_low
    , p_high))
211     #values_gauss = ndimage.gaussian_filter(values_rescale, sigma=2)
212     #values_rescale = cv2.normalize(values_gauss, None, alpha = 0,
    beta = 1, norm_type = cv2.NORM_MINMAX, dtype = cv2.CV_32F)
213     #values_convert = values_rescale.astype(numpy.float)
214     #values_filter = denoise_bilateral(values_convert)
215     #values_median = median(values_rescale)
216     #values_norm_8u = cv2.normalize(values_median, None, alpha = 0,
    beta = 255, norm_type = cv2.NORM_MINMAX, dtype = cv2.CV_8U)
217     #values_hyster = apply_hysteresis_threshold(values_rescale,
    p_high, p_low)
218
219
220     #make the mask a boolean mask or range limited to 0 and 1
221     if masks:
222         mask = numpy.ma.make_mask(mask, shrink=False)
223
224     #plot the fake spectrogram and the mask
225     fig = plt.figure(figsize=(15,5))
226     fig.add_subplot(1,3,1)
227     plt.imshow(values, cmap='gray')
228     fig.add_subplot(1,3,2)
229     plt.imshow(values_alt, cmap='gray')
230     if masks:
231         fig.add_subplot(1,3,3)
232         plt.imshow(mask, cmap='gray')
233     plt.show()
234
235 if __name__ == "__main__":
236     main()
```



(a) (b)

Figure 6.1: Image 6.1(a) shows a time-frequency plot that uses 32 bit values (floating point) for each pixel. Image 6.1(b) shows the other extreme, where each pixel is represented by either an "on" or "off" binary value.

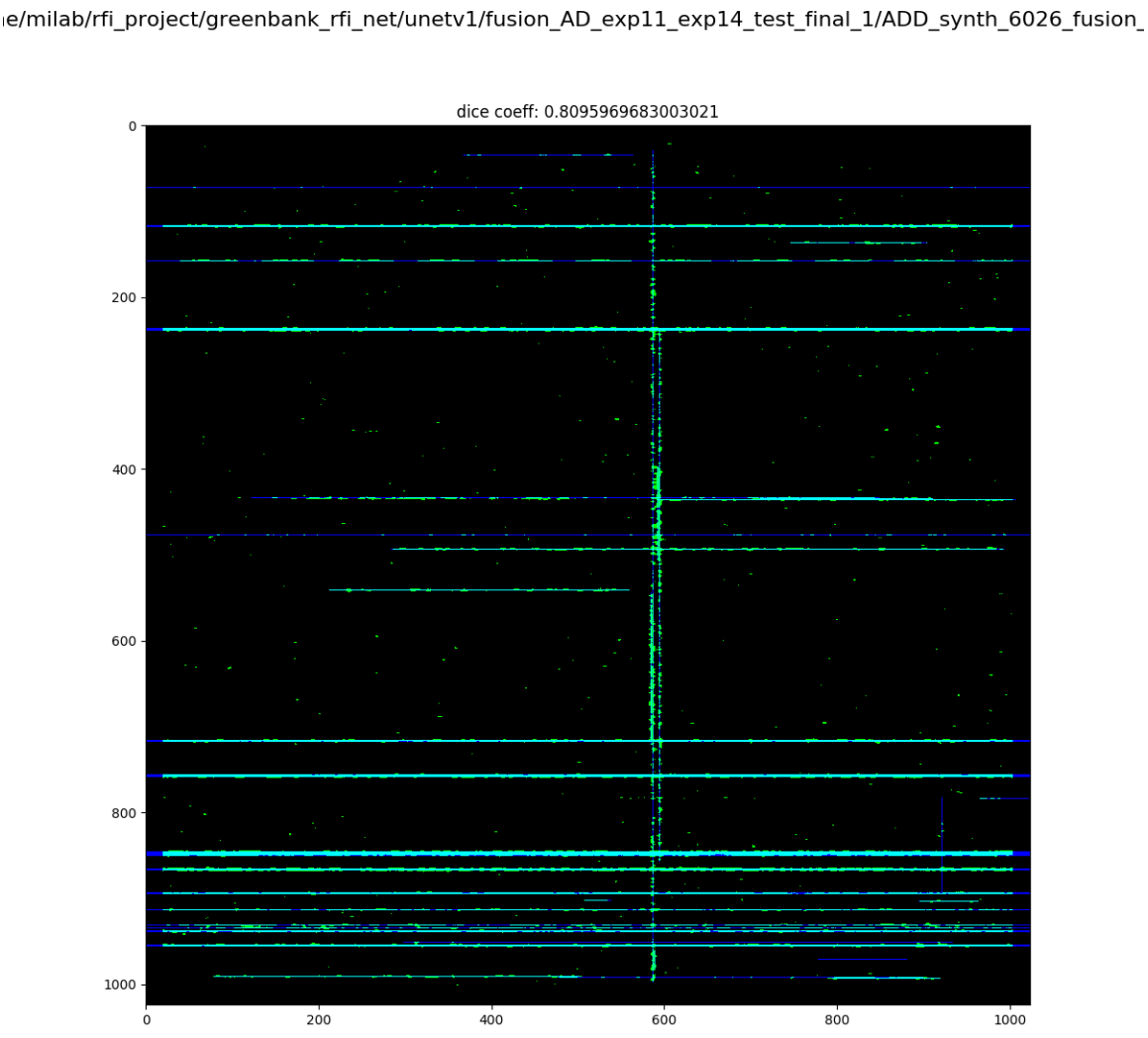


Figure 6.2: Model 11_14 ADD prediction on image 6026. Blue represents the ground truth, green represents incorrect predictions, cyan represents correct predictions. Input pixels (red) removed for clarity.

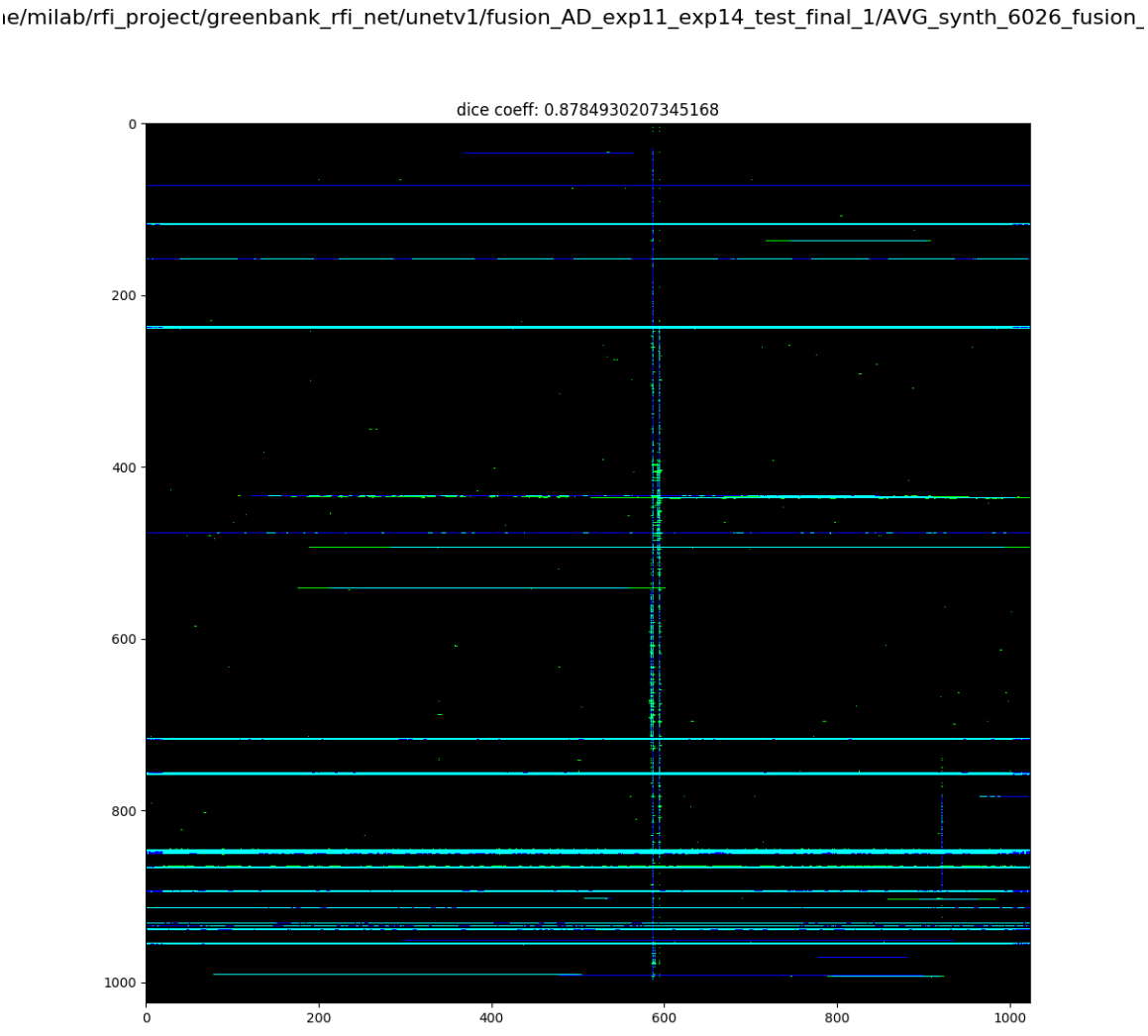


Figure 6.3: Model 11_14 AVG prediction on image 6026. Blue represents the ground truth, green represents incorrect predictions, cyan represents correct predictions. Input pixels (red) removed for clarity.

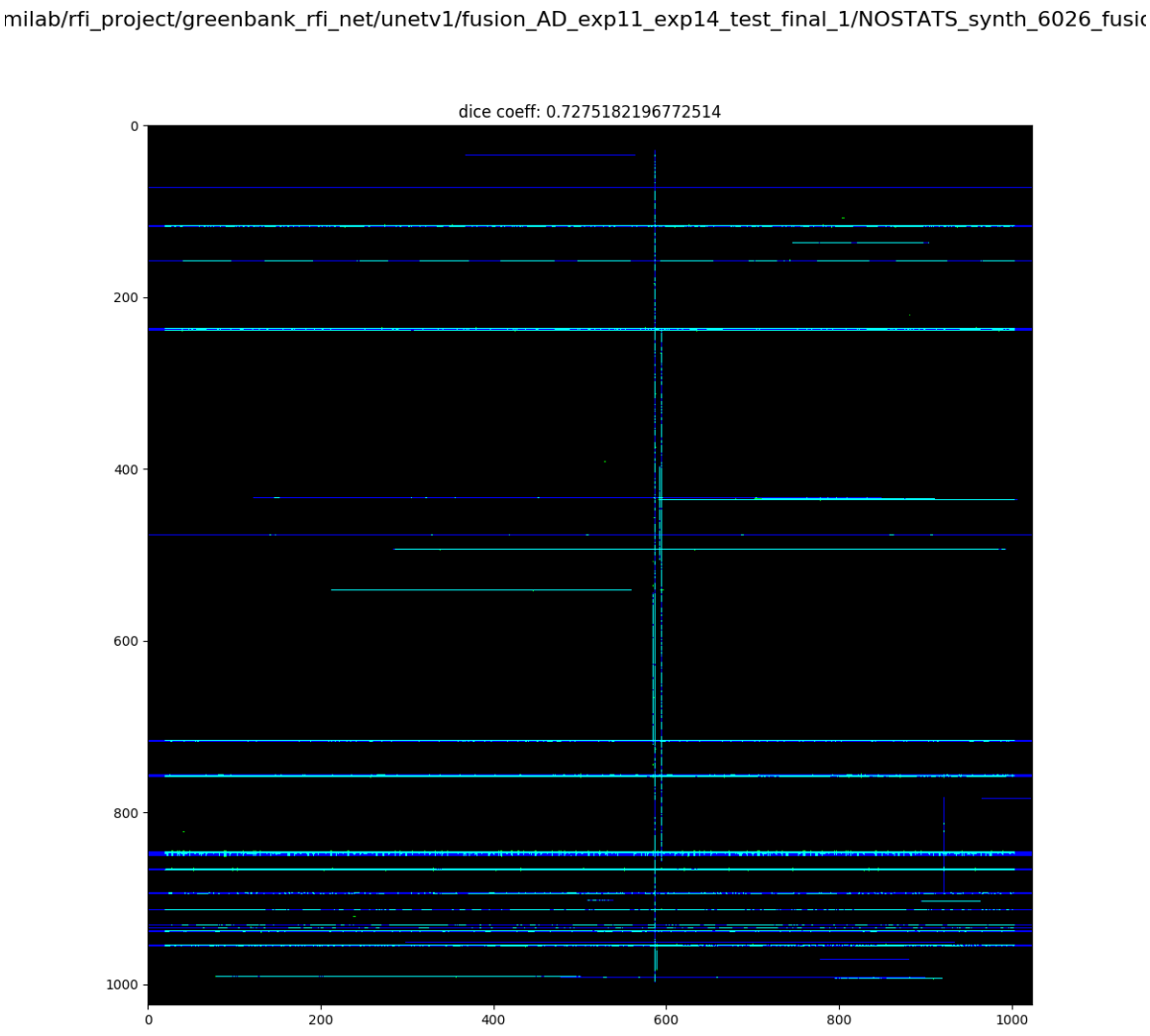


Figure 6.4: Model 11_14 NOSTATS prediction on image 6026. Blue represents the ground truth, green represents incorrect predictions, cyan represents correct predictions. Input pixels (red) removed for clarity.

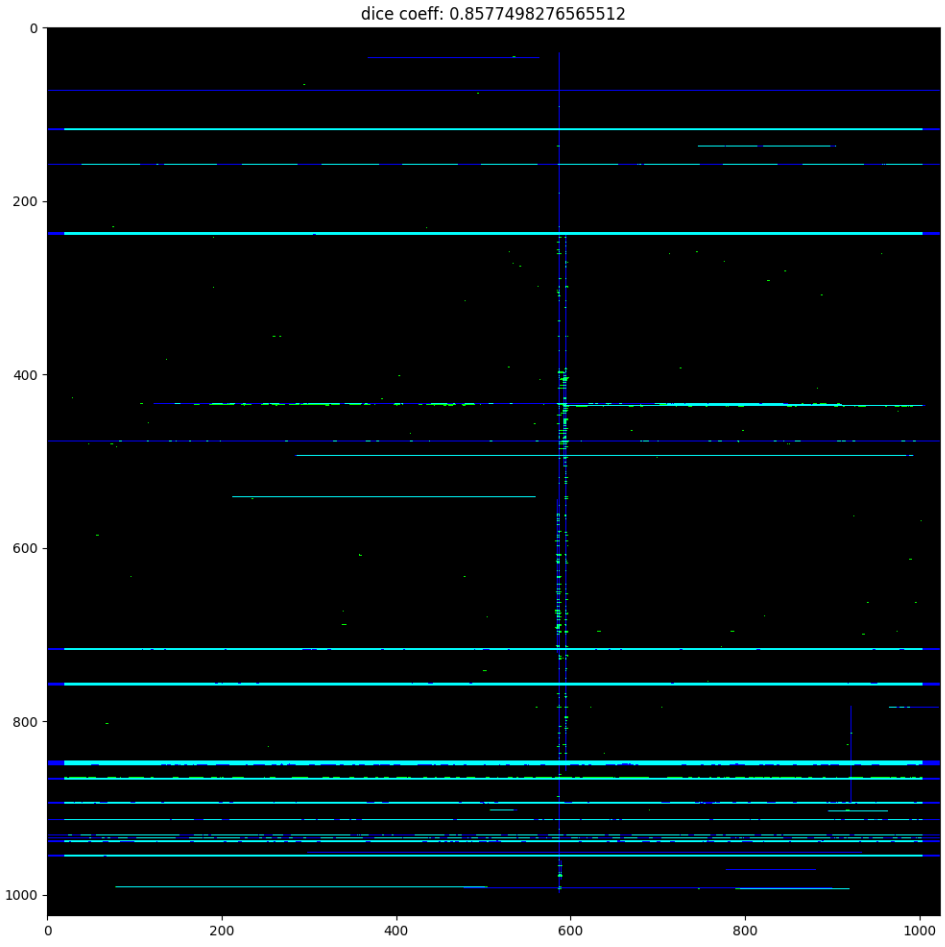


Figure 6.5: Model 11_14 STATS prediction on image 6026. Blue represents the ground truth, green represents incorrect predictions, cyan represents correct predictions. Input pixels (red) removed for clarity.

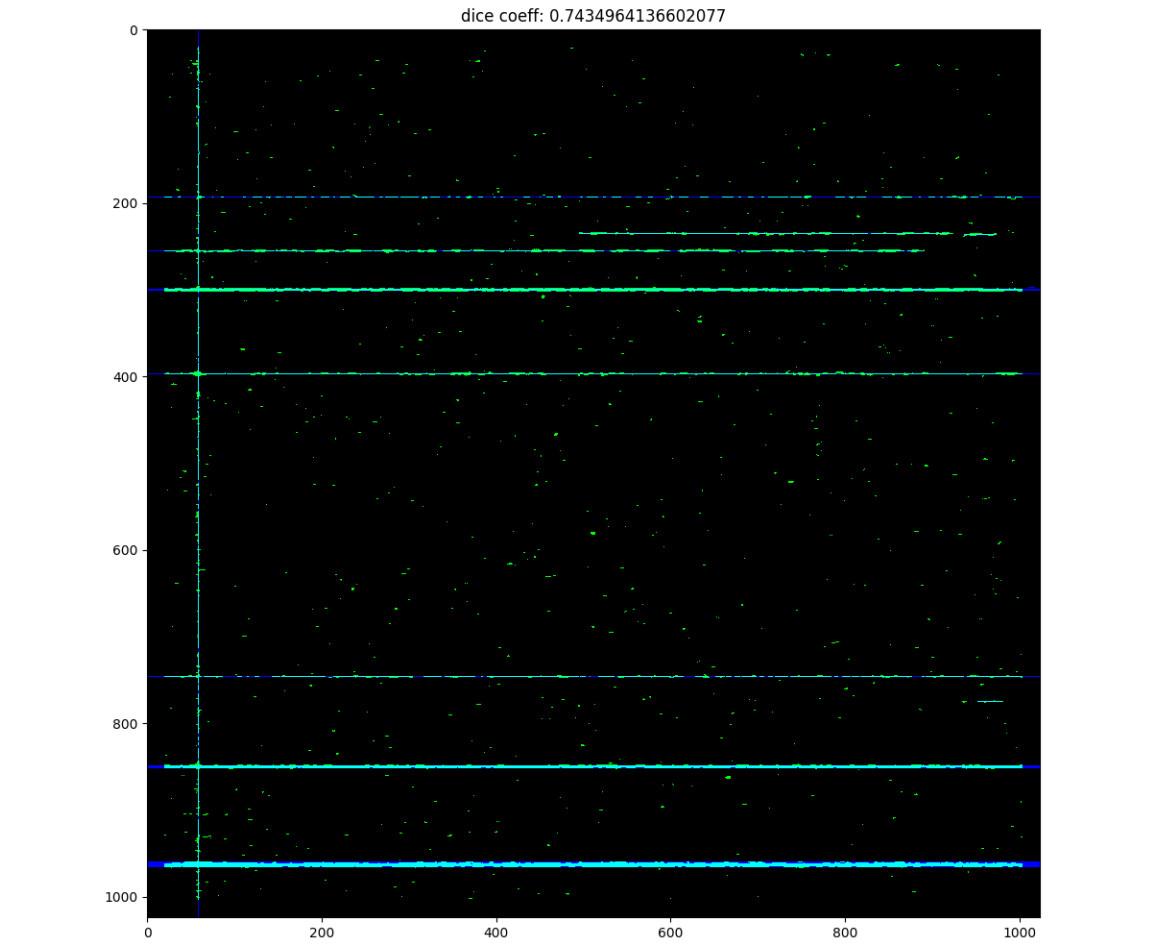


Figure 6.6: Model 11_14 ADD prediction on image 6199. Blue represents the ground truth, green represents incorrect predictions, cyan represents correct predictions. Input pixels (red) removed for clarity.

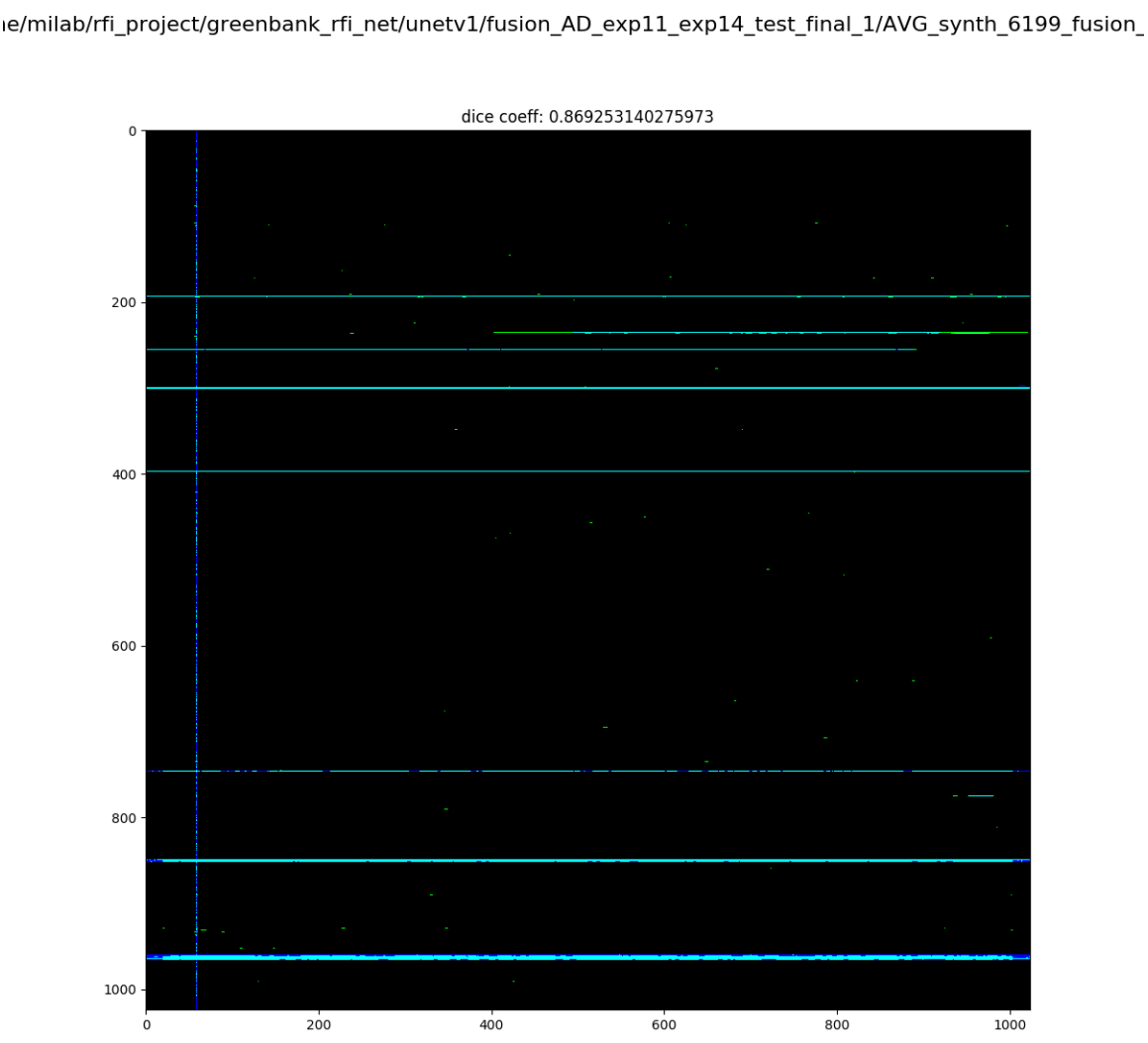


Figure 6.7: Model 11_14 AVG prediction on image 6199. Blue represents the ground truth, green represents incorrect predictions, cyan represents correct predictions. Input pixels (red) removed for clarity.

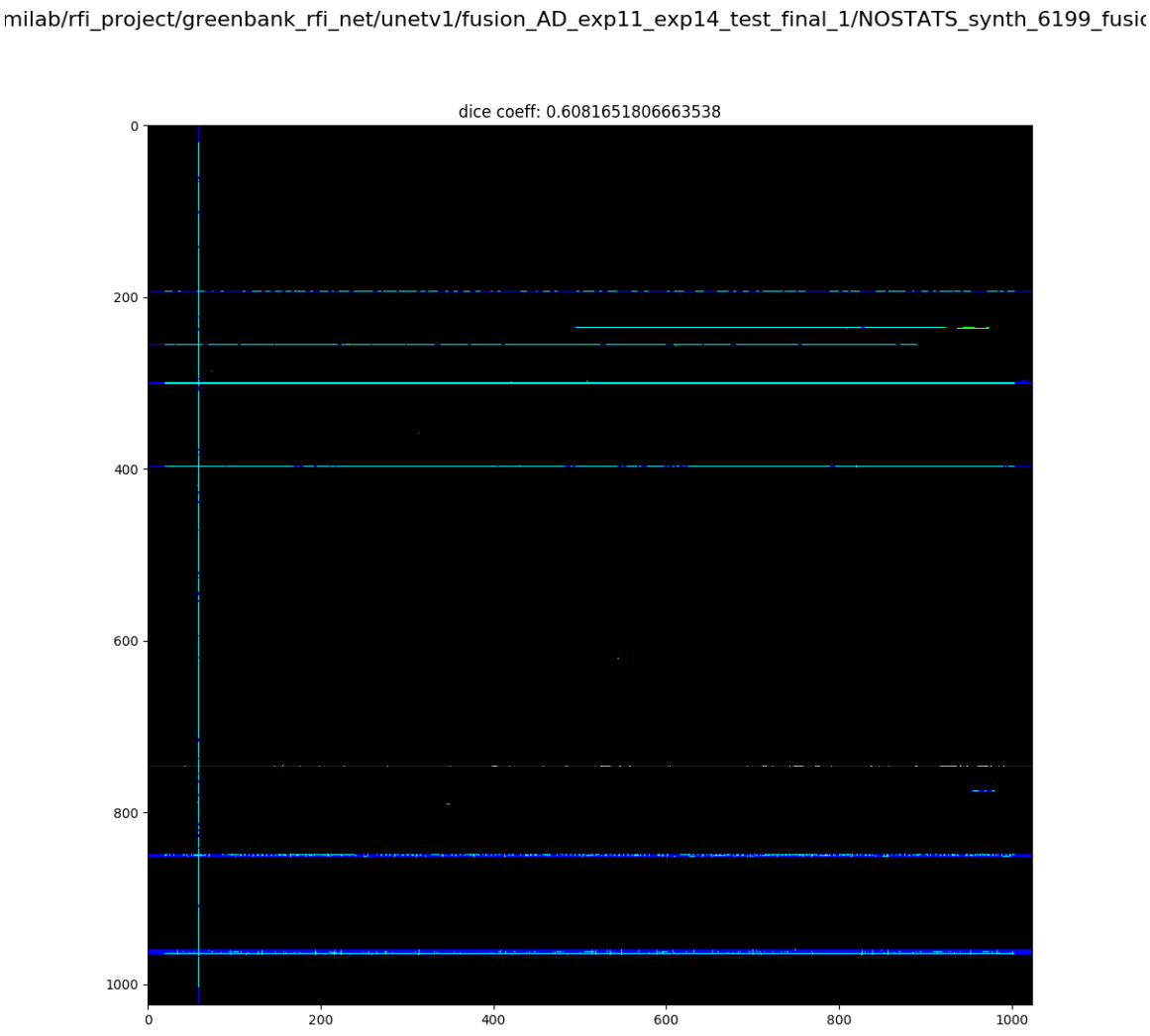


Figure 6.8: Model 11_14 NOSTATS prediction on image 6199. Blue represents the ground truth, green represents incorrect predictions, cyan represents correct predictions. Input pixels (red) removed for clarity.

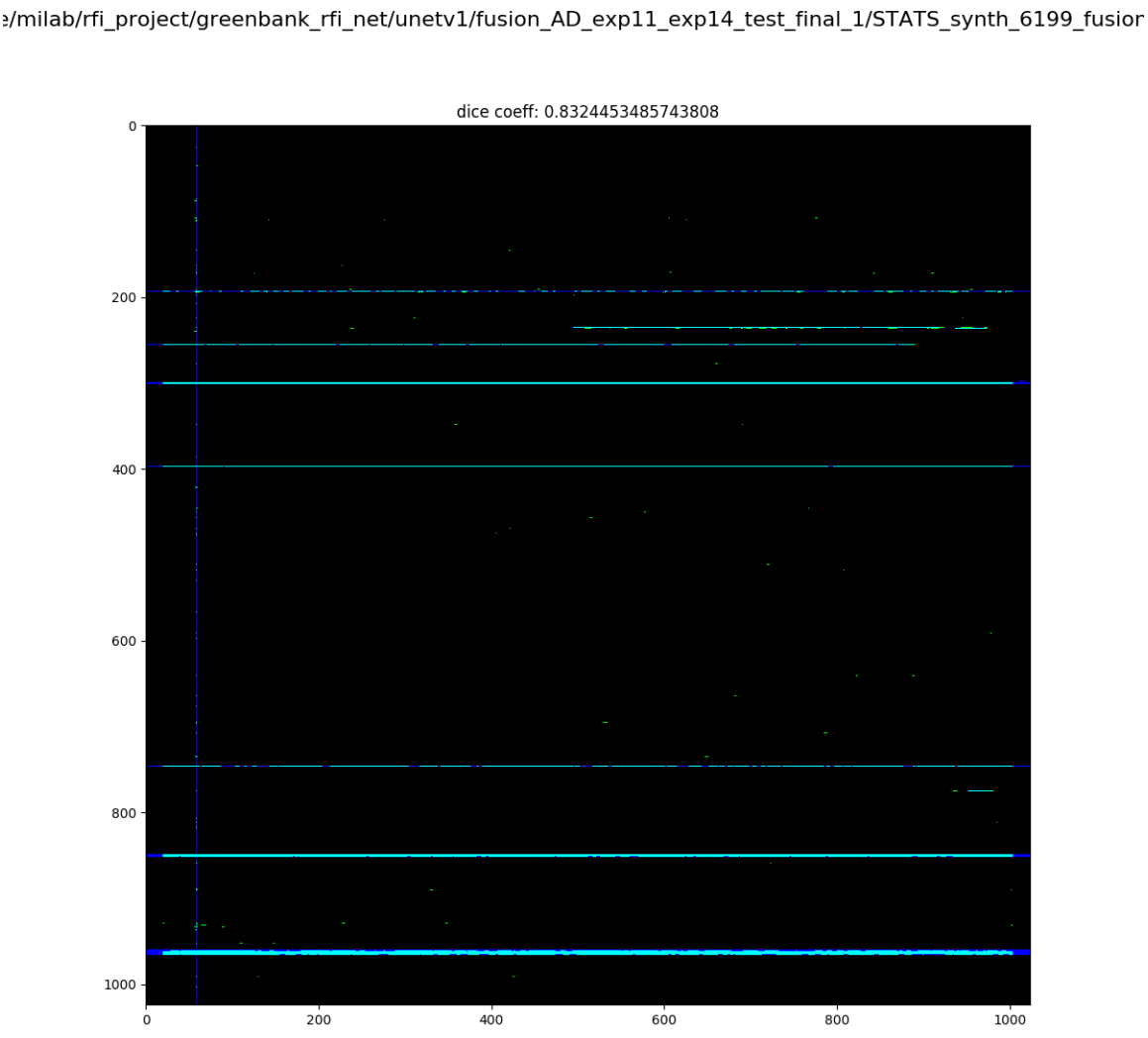


Figure 6.9: Model 11_14 STATS prediction on image 6199. Blue represents the ground truth, green represents incorrect predictions, cyan represents correct predictions. Input pixels (red) removed for clarity.

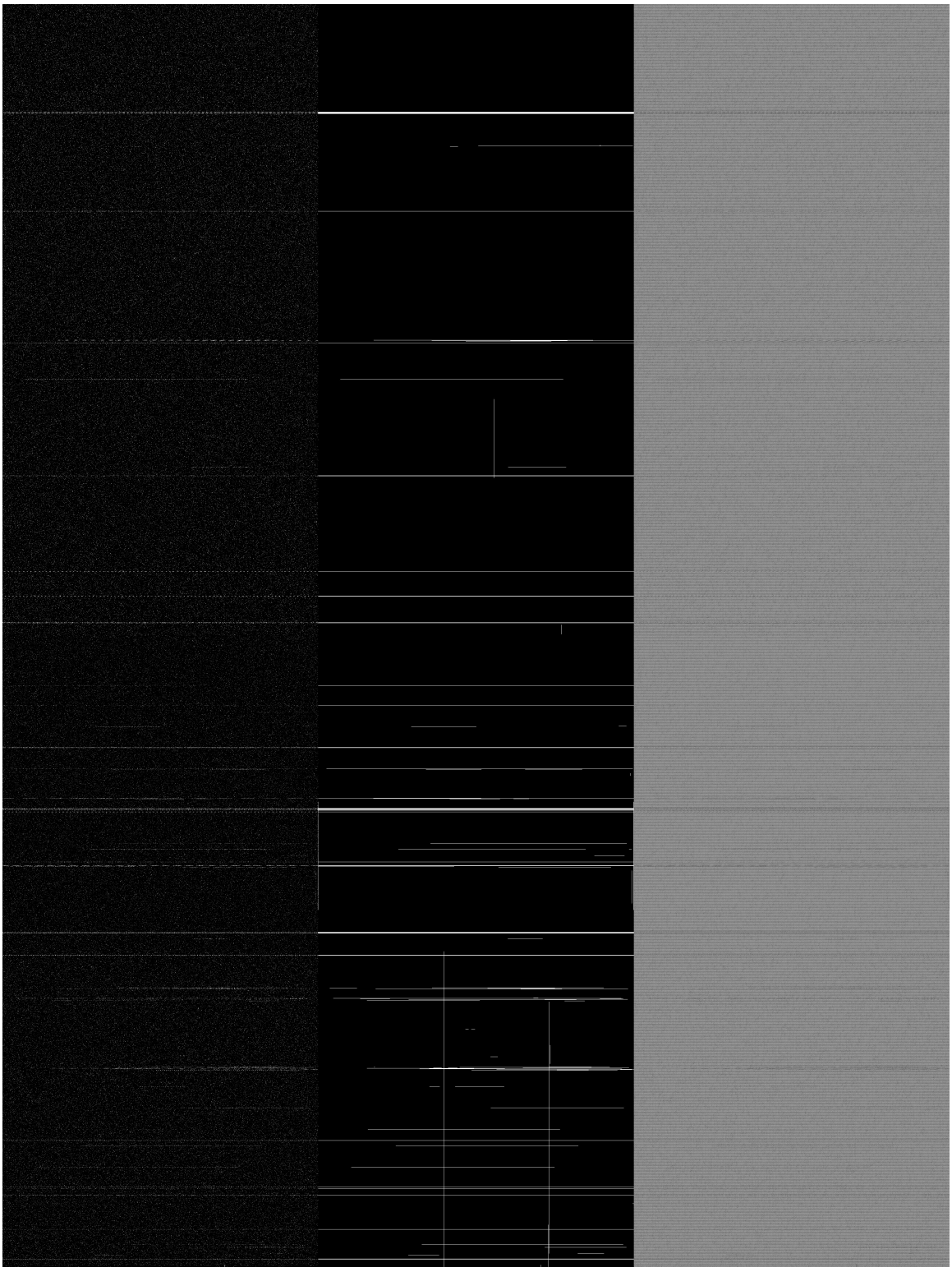


Figure 6.10: Initial series (top to bottom) of Model Exp11 training predictions. Left: input data. Middle: ground truth. Right: prediction.

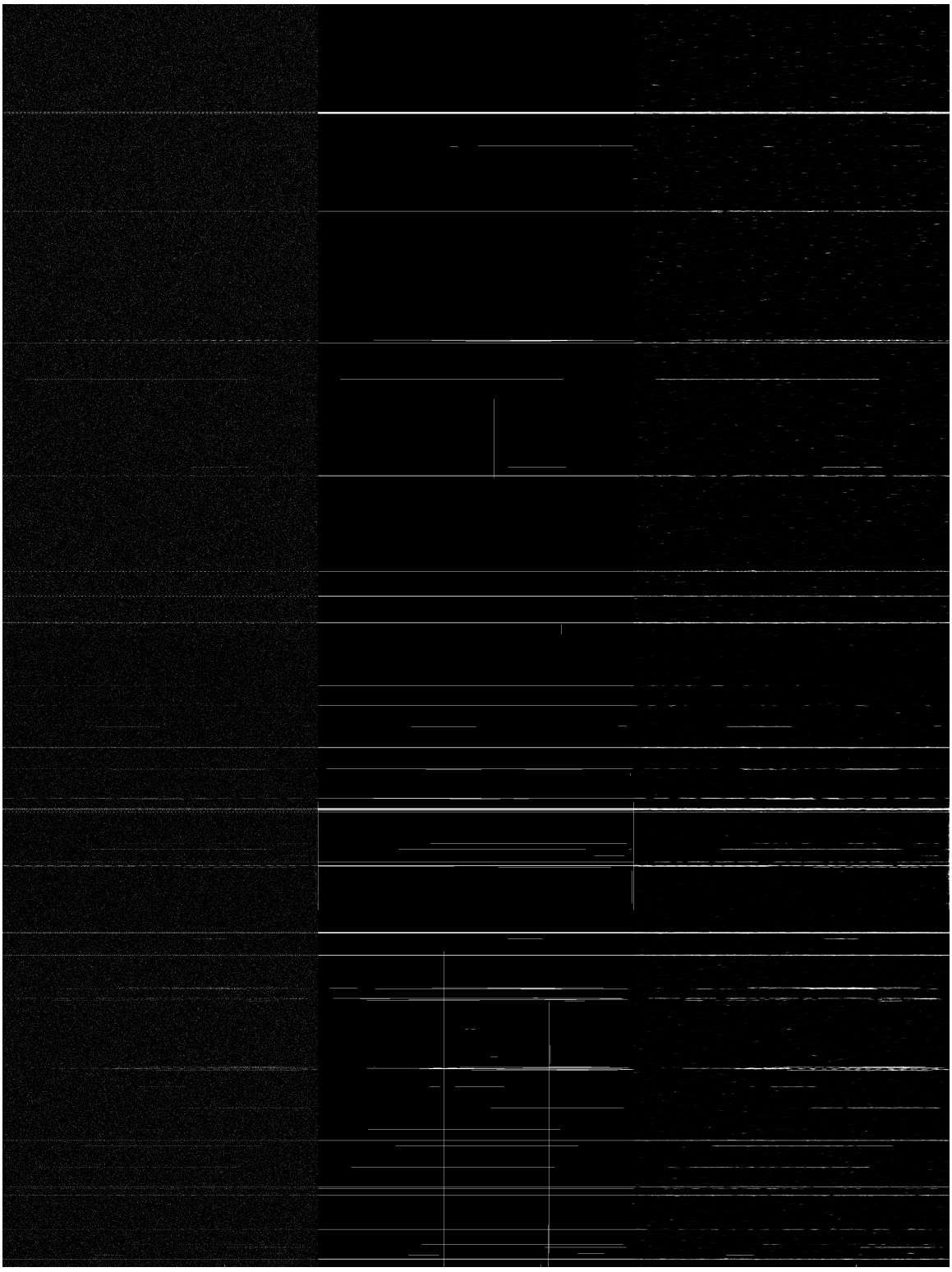


Figure 6.11: Final series (top to bottom) of Model Exp11 training predictions. Left: input data. Middle: ground truth. Right: prediction.

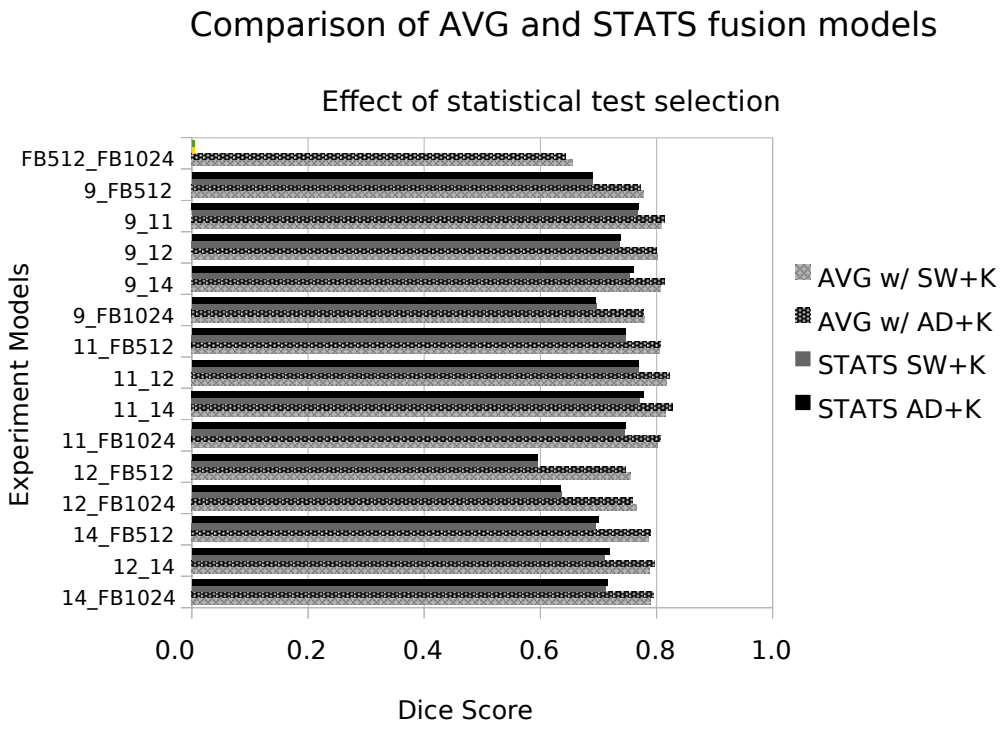


Figure 6.12: A graph displaying the comparison of the effects the Anderson-Darling and Shapiro-Wilk tests have on the Dice coefficient.

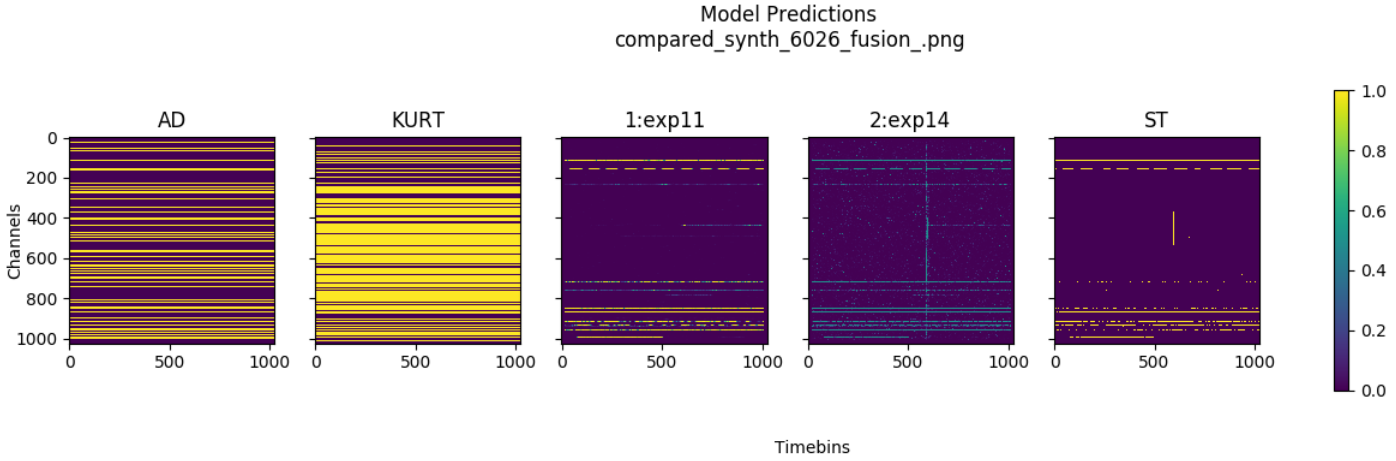


Figure 6.13: For image 6026, each predictive input into the fusion model.

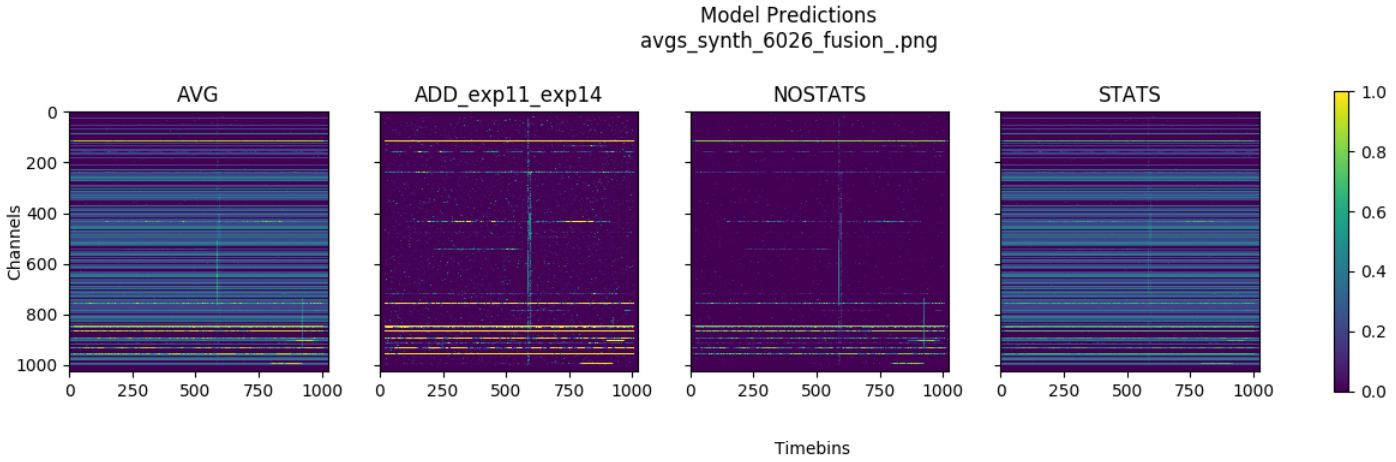


Figure 6.14: For image 6026, each predictive output from fusion model variations.

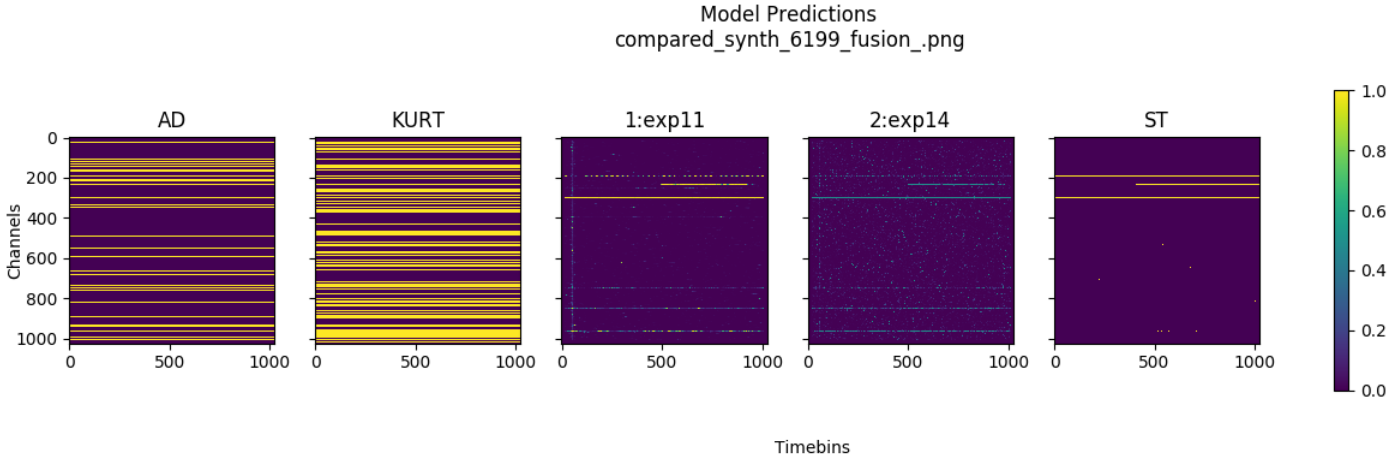


Figure 6.15: For image 6199, each predictive input into the fusion model.

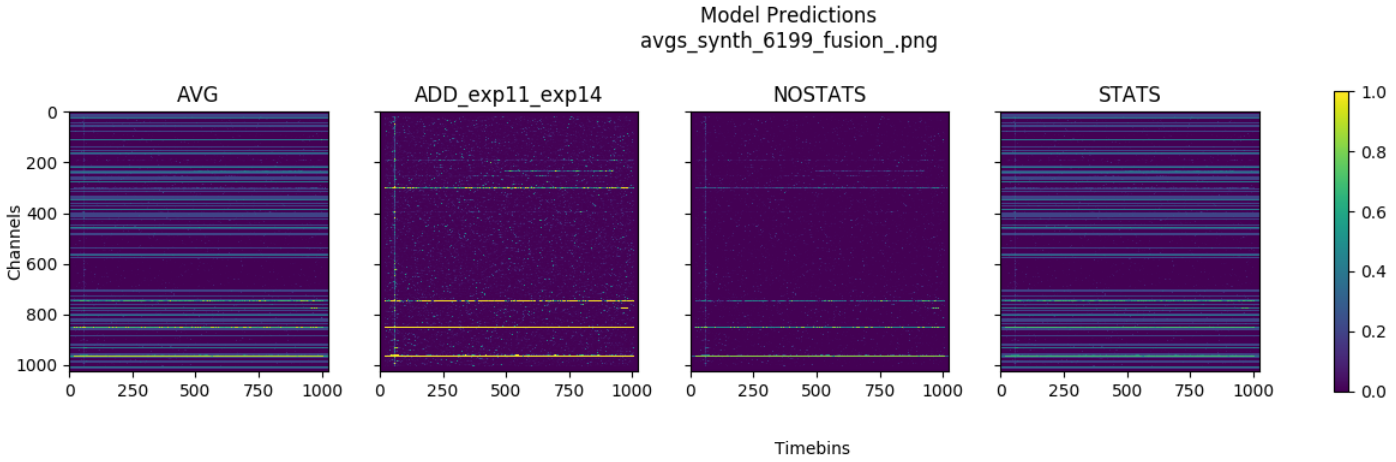


Figure 6.16: For image 6199, each predictive output from fusion model variations.