3-22-2019

Unresolved Object Detection Using Synthetic Data Generation and Artificial Neural Networks

Yong U. Sinn

UNRESOLVED OBJECT DETECTION USING SYNTHETIC DATA GENERATION AND ARTIFICIAL NEURAL NETWORKS

THESIS

Yong U Sinn, Capt, USAF

AFIT-ENG-MS-19-M-055

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-MS-19-M-055

UNRESOLVED OBJECT DETECTION USING SYNTHETIC DATA GENERATION AND ARTIFICIAL NEURAL NETWORKS

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Computer Engineering

Yong U Sinn, B.S.E.E.

Capt, USAF

March 2019

AFIT-ENG-MS-19-M-055

UNRESOLVED OBJECT DETECTION USING SYNTHETIC DATA GENERATION AND ARTIFICIAL NEURAL NETWORKS

THESIS

Yong U Sinn, B.S.E.E.
Capt, USAF

Committee Membership:

Dr. Kenneth M. Hopkinson
Chair

Dr. Bryan J. Steward
Member

Dr. Brett J. Borghetti
Member

AFIT-ENG-MS-19-M-055

Abstract

This research presents and solves constrained real-world problems of using synthetic data to train artificial neural networks (ANNs) to detect unresolved moving objects in wide field of view (WFOV) electro-optical/infrared (EO/IR) satellite motion imagery. Objectives include demonstrating the use of the Air Force Institute of Technology (AFIT) Sensor and Scene Emulation Tool (ASSET) as an effective tool for generating EO/IR motion imagery representative of real WFOV sensors and describing the ANN architectures, training, and testing results obtained. Deep learning using a 3-D convolutional neural network (3D ConvNet), long short term memory (LSTM) network, and U-Net are used to solve the problem of EO/IR unresolved object detection. U-Net is shown to be a promising ANN architecture for performing EO/IR unresolved object detection. In two of the experiments, U-Net achieved 90% and 88% pixel prediction accuracy. In addition, the results show ASSET is capable of generating sufficient information needed to train deep learning models.

Table of Contents

Page

List of Figures

Figure Page

List of Abbreviations

AFIT The Air Force Institute of Technology

AI artificial intelligence

AOI area of interest

API application program interface

ASSET AFIT Sensor and Scene Emulation Tool

AUC area under the curve

BCE binary cross-entropy

ConvNet convolutional neural network

CT computed tomography

CTISR Center for Technical Intelligence Studies and Research

EO/IR electro-optical and infrared

FCN fully convolutional network

FN false negative

FNR false negative rate

FP false positive

FPR false positive rate

GEO geosynchronous earth orbit

LMS least-mean-square

LSTM long short-term memory network

MAE mean absolute error

MATLAB Matrix Laboratory

MRI magnetic resonance image

MSE mean squared error

PR precision-recall

ReLU rectified linear unit

RNN recurrent neural network

ROC receiver operating characteristic

SNR signal-to-noise ratio

SVM support vector machine

TN true negative

TP true positive

TPR true positive rate

WFOV wide field of view

UNRESOLVED OBJECT DETECTION USING SYNTHETIC
DATA GENERATION AND ARTIFICIAL NEURAL NETWORKS

I. Introduction

1.1 Background

Electro-optical and infrared (EO/IR) unresolved object detection using wide field of view (WFOV) sensors is a challenging problem due to the numerous variables and complexities involved. As defined here, EO/IR unresolved objects are not spatially resolved, occupy a small ratio of pixels within an image, and can be indistinguishable from satellite sensor noise. Satellite sensors, orbiting at or below a geosynchronous earth orbit (GEO), also operate over diverse environments with varying background signal characteristics. Unstable atmospheric and weather conditions add to the complexity of the problem.

Many of the existing methods of solving the EO/IR unresolved object detection problem involves using closed form algorithms to process a sequence of images, as opposed to a single image at a time. This allows detection algorithms to utilize both the temporal and spatial changes between images to better distinguish clutter or background pixels from the desired object pixels. Example algorithms include: 3D matched filtering, temporal filtering, and triple temporal filters [1]. Downsides to using closed form algorithms are the amount of computational power and time requirements needed to produce a solution [1]. In addition, these closed form algorithms are often generic and are not optimized for any specific satellite environment.

1.2 Problem Statement

Using artificial neural networks (ANNs) to optimally solve the EO/IR unresolved object detection problem is an intractable task due to the vast volume of labeled truth data required to train an ANN for the wide variability in object, background, atmospheric, and sensor conditions. That said, there have been recent advancements in EO/IR data generation which may allow generation of sufficient quantities of realistic, labeled data to effectively perform training. This is an unexplored area which combines knowledge gained from the remote sensing and medical artificial intelligence (AI) fields. The Air Force Institute of Technology (AFIT) Sensor and Scene Emulation Tool (ASSET) is the key enabler of this research and is the source of all EO/IR data generation [2].

1.3 Research Goals

This thesis is the initial exploration of using synthetically generated EO/IR datasets in training deep learning models to perform unresolved object detection. It documents the EO/IR data generation settings using ASSET, details the ANN data pre-processing steps, explores several ANN architectures created specifically for unresolved object detection, and evaluates the performance impacts of training with different EO/IR datasets.

1.4 Research Questions

The work in this thesis was designed with the intent of answering these three overarching questions:

1. Can ANNs be used to perform EO/IR unresolved object detection?

2. Which ANN architecture could be used?

3. How does the object signal-to-noise ratio (SNR), overall background to object pixel ratio, and the number of scenarios within the dataset impact ANN performance?

1.5 Hypothesis

This research hypothesizes that ANNs can be used to perform EO/IR unresolved object detection with the appropriate architecture. It also hypothesizes a larger EO/IR dataset with more scenarios will have a greater positive impact to the performance of the ANN.

1.6 Thesis Overview

This thesis is described in five chapters. Chapter 2 provides background information on ASSET, 3D convolutional neural network (ConvNet), long short-term memory network (LSTM), and U-Net. It also provides an overview of the related work that supported the development of this work. Chapter 3 provides the methodology of the experiments and describes all ANN architectures, the dataset generation settings, dataset pre-processing steps, and the experimental assumptions. Chapter 4 displays the training results, describes the prediction performance, and provides visual graphs of the ANNs predictions. Chapter 5 presents the results of the experiments and identifies potential future work.

II. Background and Related Research

2.1 Overview

This chapter provides an overview of ASSET, 3D ConvNets, LSTMs, and U-Nets. This section also outlines the related research used in developing this research.

2.2 ASSET

ASSET is a physics-based, image chain model used to generate synthetic imagery data captured by satellite sensors. ASSET is supported by AFIT's Center for Technical Intelligence Studies and Research (CTISR). ASSET is written in the Matrix Laboratory (MATLAB) computer programming language and was originally developed "...to support student research where absolute knowledge of object position and radiometric signature (i.e. truth) is needed, as is the case in detection and tracking efforts, as well as research where large number of datasets are required, such as machine learning" [2]. A detailed description and the initial validation results of ASSET can be found in the paper by Young et al. [2].

ASSET currently contains 182 customizable scenario parameters under five categories: input/output, simulation options, source options, scene parameters, and sensor parameters. By adjusting the parameters within the categories, ASSET is able to generate scenarios with customizable satellite noise characteristics, satellite sensor artifacts, atmospheric conditions, and background scenes.

In addition, ASSET has the capability of inserting a user defined number of moving objects into the generated satellite images and videos. There are currently 13 customizable object parameters. There are currently 13 customizable parameters controlling the signal level and motion of randomly generated moving objects, or the user can provide their own signal and motion profile. The objects created by ASSET

are not spatially resolved, thus they are not visually identifiable. In other words, it is not possible to recognize the object type (e.g. planes, cats, dogs) by its spatial characteristics. Instead, the objects are representative of point-like signals and can be customized to represent objects as observed by real satellite sensors.

2.3 Convolutional Neural Network (CNN or ConvNet)

2D ConvNets are used to solve problems where spatial relationships exist. The convolution and pooling layers reduce the number of parameters in the ANN, which makes 2D ConvNets the go-to network for prediction problems involving images. 3D ConvNets gained popularity when results showed they could outperform 2D ConvNets in solving the problem of human action recognition in videos [3]. Tran et al. has shown the effectiveness of extracting features using 3D ConvNets trained on a large scale video dataset [4]. 3D ConvNets have since expanded in scope to other fields using videos or image sequences as the source of data. Similar to 2D ConvNets, 3D ConvNets have convolution and pooling layers; however, these layers are applied simultaneously in the spatial and temporal dimensions.

2.4 Long Short-Term Memory Network (LSTM)

LSTMs are a type of recurrent neural network (RNN) used to solve sequence prediction problems. Kim et al. has shown LSTMs are effective at predicting future vehicle trajectory coordinates [5]. Sequence prediction problems can be categorized as one-to-one, one-to-many, many-to-one, and many-to-many. This research addresses the many-to-many prediction problem. LSTMs were designed with an improvement which prevents the vanishing and exploding gradient problem found in RNNs. LSTM connections also form a directed graph. The directed graph structure allows LSTMs to maintain memory by using the output as the input in the next sequence member.

2.5 U-Net

U-Net was based on the fully convolutional network (FCN) [6] and was originally used for biomedical image segmentation [7]. The FCN was published in 2014 and contains convolutional, downsampling, and upsampling layers. There are no dense or fully connected layers in a FCN. Similarly, U-Net does not contain dense layers. This vastly reduces the size and training times when compared to traditional ConvNets. An example 3D U-Net architecture can be seen in appendix F. The left side is known as the contracting path, and it reduces the spatial resolution of the image while extracting the feature information. The right side is known as the expansive path, and it increases the spatial resolution while reinserting the feature information. The resulting output is a segmentation map, where every pixel is labeled based on the classification criteria. This research uses binary classification, and every pixel is either a background or an object pixel.

2.6 Related Research

Using ANNs to detect unresolved objects (i.e. point-like signals) in EO/IR motion imagery is a relatively unexplored area when compared to the more general problem of using networks to detect and track spatially resolved objects (e.g. cars, dogs, people). Limited work addressing the problem of detecting unresolved objects in EO/IR motion imagery dates back to late 1980's. In early research, using ANNs to detect objects was more focused on theoretical uses than actual implementation. One of the earliest papers using ANNs to detect unresolved objects in EO/IR imagery was published in 1989 [8]. In the paper, Chenoweth describes the potential use of an ANN to replace a least-mean-square (LMS) filter during image processing. In 1992, Liou and Azimi-Sadjadi developed an ANN which leverages the temporal and spatial relationships present in motion imagery to identify object windows [9]. In

1995, Shirvaikar and Trivedi compare the performance of a neural network filter to a contrast box filter in detecting unresolved objects in high clutter thermal images [10].

Afterwards, the remote sensing AI community shifted focus from using ANNs in favor of deterministic algorithms. Publications using ANNs to detect unresolved objects declined until the resurgence of deep learning, which occurred in the early 2010's. Two recent publications involving the use of ANNs to detect unresolved objects. In 2010, Yun and Zhou compare the performance of support vector machine (SVM) to ANNs in low SNR images [11]. In 2017, Hu et al. use a combination of a bilateral filter with a LSTM to detect and predict unresolved object trajectories [12]. The concept of using a LSTM to predict object locations was further reinforced by the publication from Kong et al. [13].

The medical AI community was surveyed due to the dearth of publications in the remote sensing AI community. The medical AI community had several recent publications involving the use of 3D ConvNets to detect anomalies using computed tomography (CT) and magnetic resonance image (MRI) scans. CT and MRI scans are single channel images and are comparable to single channel EO/IR images. Dou et al. used 3D ConvNets to detect cerebral microbleeds in MRIs [14] and Huang et al. used 3D ConvNets to detect lung nodules in CT scans [15]. More importantly, these papers demonstrate the effectiveness and viability of using sequential images to detect small anomalies in single channel images.

The medical AI field was continually consulted and referenced throughout the research. The technique of semantic image segmentation using U-Net matched the ANN profile needed to solve unresolved object detection in less constrained environments. U-Net is a proven and popular architecture based on subsequent research within the medical AI field. Research proving the viability of using U-Net for image segmentation include: biologically-informed brain tumor segmentation [16] and retina

blood vessel segmentation [17]. There has even been success at expanding U-Net into a recurrent residual network [18].

2.7 Background Summary

This chapter presented an overview of ASSET, 3D ConvNets, LSTMs, and U-Nets, which also provided insight into why these ANN architectures were chosen for this research. It also presents the history of ANNs in the field of remote sensing and the related research found in the medical AI community. This thesis is the culmination of the knowledge gained by combining the techniques from both the remote sensing and medical AI communities.

III. Methodology

3.1 Problem/Objective

This thesis aims to provide the initial exploration of using synthetically generated EO/IR datasets to train ANNs in-performing unresolved object detection. It documents the EO/IR data generation settings using ASSET, details the ANN data pre-processing steps, explores several ANN architectures created specifically for unresolved object detection, and evaluates the performance impacts of training with different EO/IR datasets. The objectives of the thesis are to answer these research questions:

1. Can ANNs be used to perform EO/IR unresolved object detection?
2. Which ANN architecture could be used?
3. How does the object signal-to-noise ratio (SNR), overall background to object pixel ratio, and the number of scenarios within the dataset impact ANN performance?

This research methodology followed the Spiral model, a software engineering design process model. This method was chosen to reduce the overall risk of not producing viable results. The goal was to increase the difficulty of the problem with each cycle or phase. Figure 1 lists the high level steps used to guide this research.

In total, two phases were completed using this methodology. The goal of Phase 1 was to determine the viability of using synthetically generated EO/IR datasets to train ANNs to perform unresolved object detection and prediction. A small scaled problem was created and successfully solved using a synthetic dataset. The success of Phase 1 was critical in order to proceed with using ASSET to generate EO/IR datasets.

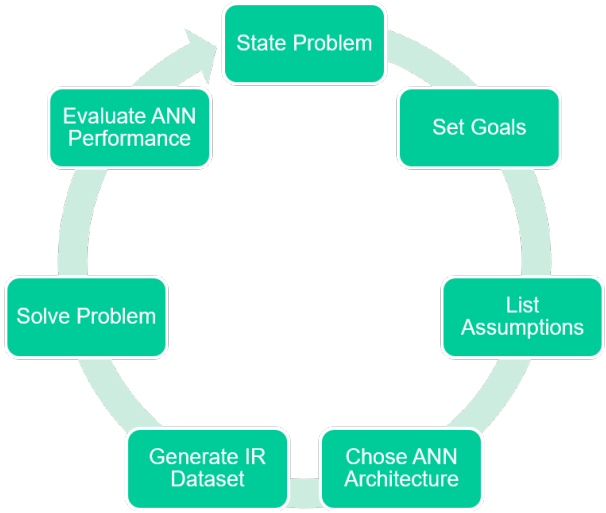


Figure 1. Spiral model

The goal of Phase 2 was to analyze the performance of an ANN solving the problem of unresolved object detection trained using different EO/IR datasets. Each experiment used an untrained ANN as the baseline. Transfer learning was not used to train any ANN. Phase 2 also focused on the impact of dataset generation using ASSET, while relaxing several small scaled problem constraints identified in Phase 1. Due to the relaxation of the problem constraints, a different approach was needed to solve the problem of EO/IR unresolved object detection. In particular, Phase 1 constraints guaranteed an unresolved object would be present in every frame of a video. Phase 2 removed this constraint which renders the ANN architecture used for Phase 1 obsolete. Therefore, Phase 2 solves the unresolved object detection problem using semantic image segmentation.

3.2 Phase 1

Phase 1 was the initial exploration to determine the viability of using ANNs to detect unresolved objects. Phase 1 combined the techniques of extracting features, using 3D ConvNets, and predicting future trajectories, using LSTMs, into one pipeline.

This phase was needed to ensure EO/IR data generated by ASSET contained the appropriate information needed to perform machine learning.

Assumptions.

In Phase 1, the scenarios' variables were constrained to create a small scaled problem. Listed below were the assumptions used for Phase 1:

- ASSET was used to create EO/IR motion imagery data with noise, clutter, and atmospheric conditions representative of a generic space-based sensor.
- There was no attempt to reduce noise and clutter in the motion imagery generated by ASSET. There are proven techniques of suppressing background clutter to amplify the object signals using ANNs, and noise reduction was not a focus of Phase 1 [10, 19].
- In the scenarios considered, atmospheric effects such as clouds and attenuation were set to nominal and did not influence object detection.
- Each scenario was limited to one moving object that is present in each frame. This avoids the added complexity of first determining whether or not an object is present.
- All generated objects moved at a constant speed with either a linear or slightly curved trajectory. Figure 2 shows example linear and curved trajectories using the same background image.
- All scenarios were generated using the same background image. Having the same background image is notionally consistent with a sensor viewing a fixed swath on the earth.

- Since the objective is to demonstrate an architecture capable of detection and prediction, the object signals were well above the background noise pixel signals.
- Even with these constraints, the unresolved objects were not visibly apparent within an image frame due to the contrast between their relatively low signal and bright background (see figure 3).

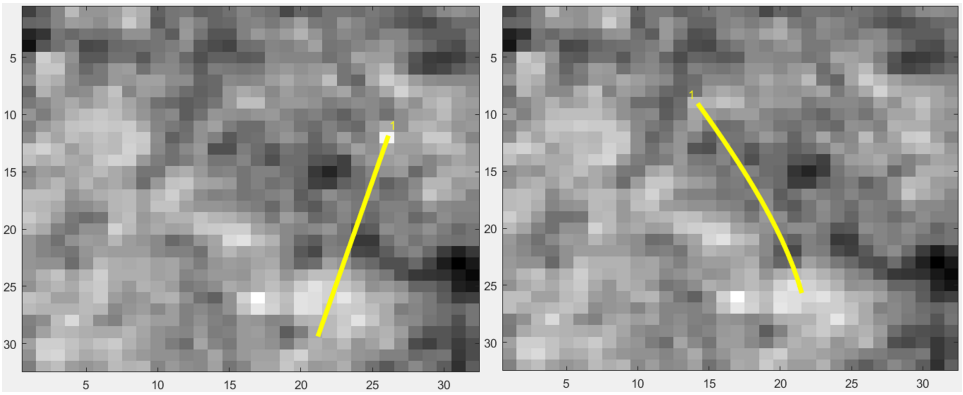


Figure 2. Example trajectories created by ASSET

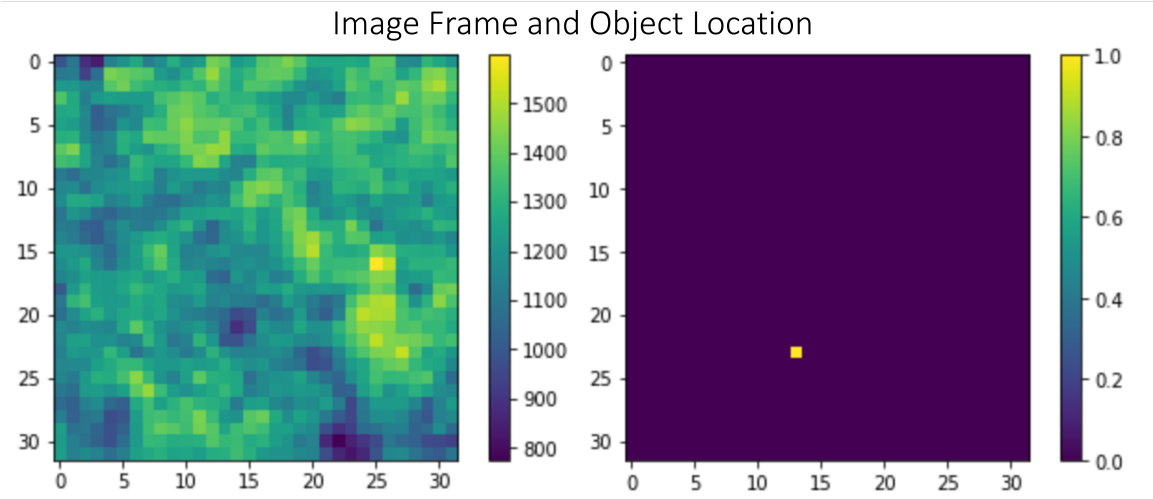


Figure 3. Example image frame with object location

ANN Architectures.

Figure 4 shows the details of the 3D ConvNet architecture and figure 5 shows the details of the LSTM architecture. Each of the models' layers used the rectified linear unit (ReLU) activation function with alpha set to 0.0, and max value set to 'None'. Both architectures were also compiled using the mean squared error (MSE) loss function and the 'Adam' optimizer with a learning rate of 0.001, beta 1 of 0.9, beta 2 of 0.999, epsilon set to 'None', decay rate of 0.0, and amsgrad set to 'False'.

The 3D ConvNet consisted of seven layers and had a total of 658,167 trainable parameters. Based on the image pixel dimensions, the 3D ConvNet layers were inspired by LeNet-5 [20]. The LSTM was smaller and only consisted of two layers, with a total of 92,102 trainable parameters.

ANN Pipeline.

Figure 6 shows a high level diagram of the ANN training and testing pipeline. First the scenarios were randomly separated using the the standard train/test split of 80/20. The data set was split by scenarios vs image frames to maintain the temporal information. Then, the 3D ConvNet was trained using the image files as the input data and the spatial tuples (x, y) found in the object files as the labeled data. The LSTM was trained only on the object location files. Finally for testing, the 3D ConvNet and LSTM predictions were separately compared against the object truth data for analysis.

Dataset Generation.

ASSET generates scenarios by using three sources of information:

1. Background image - The ASSET default background image used for Phase 1 is shown in appendix A.

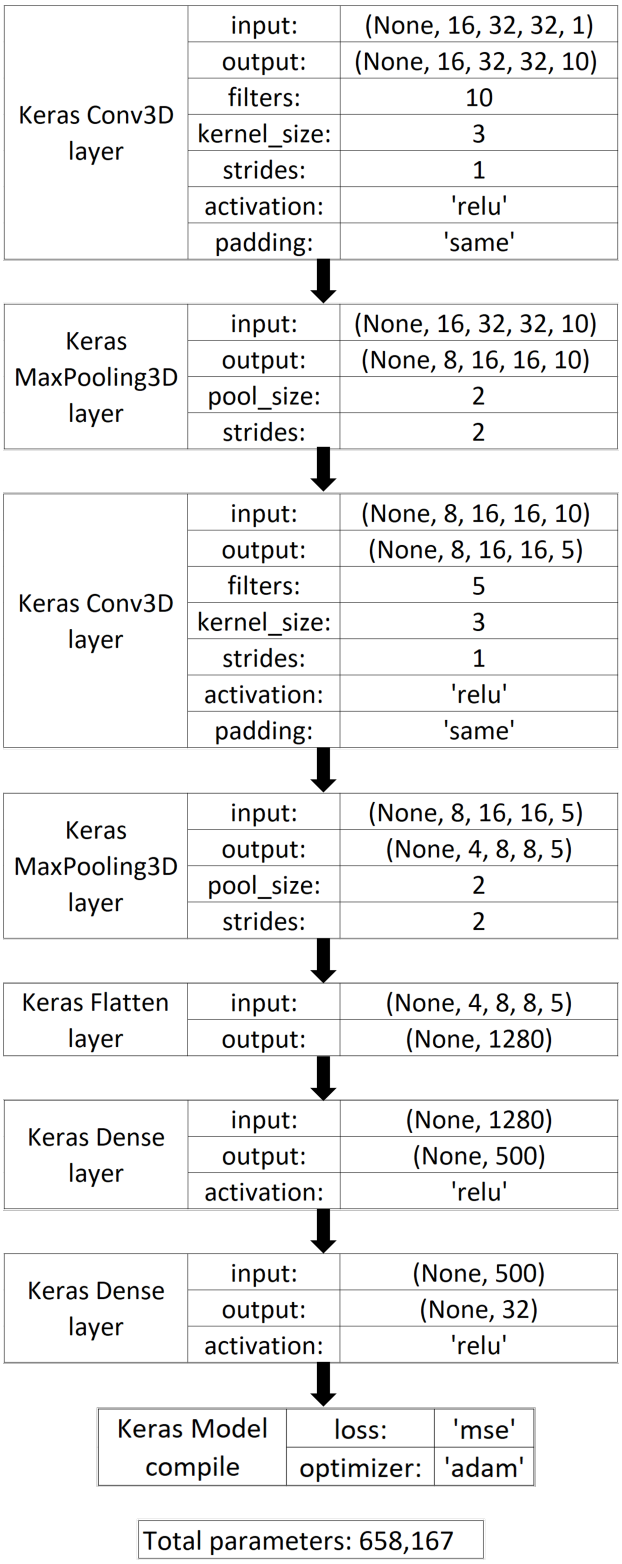


Figure 4. Phase 1: 3D ConvNet architecture

Keras LSTM layer	input:	(None, 5, 2)
	output:	(None, 150)
	nodes:	150
Keras Dense layer	input:	(None, 150)
	output:	(None, 2)
	activation:	'relu'
Keras Model compile	loss:	'mse'
	optimizer:	'adam'

Total parameters: 92,102

Figure 5. Phase 1: LSTM architecture

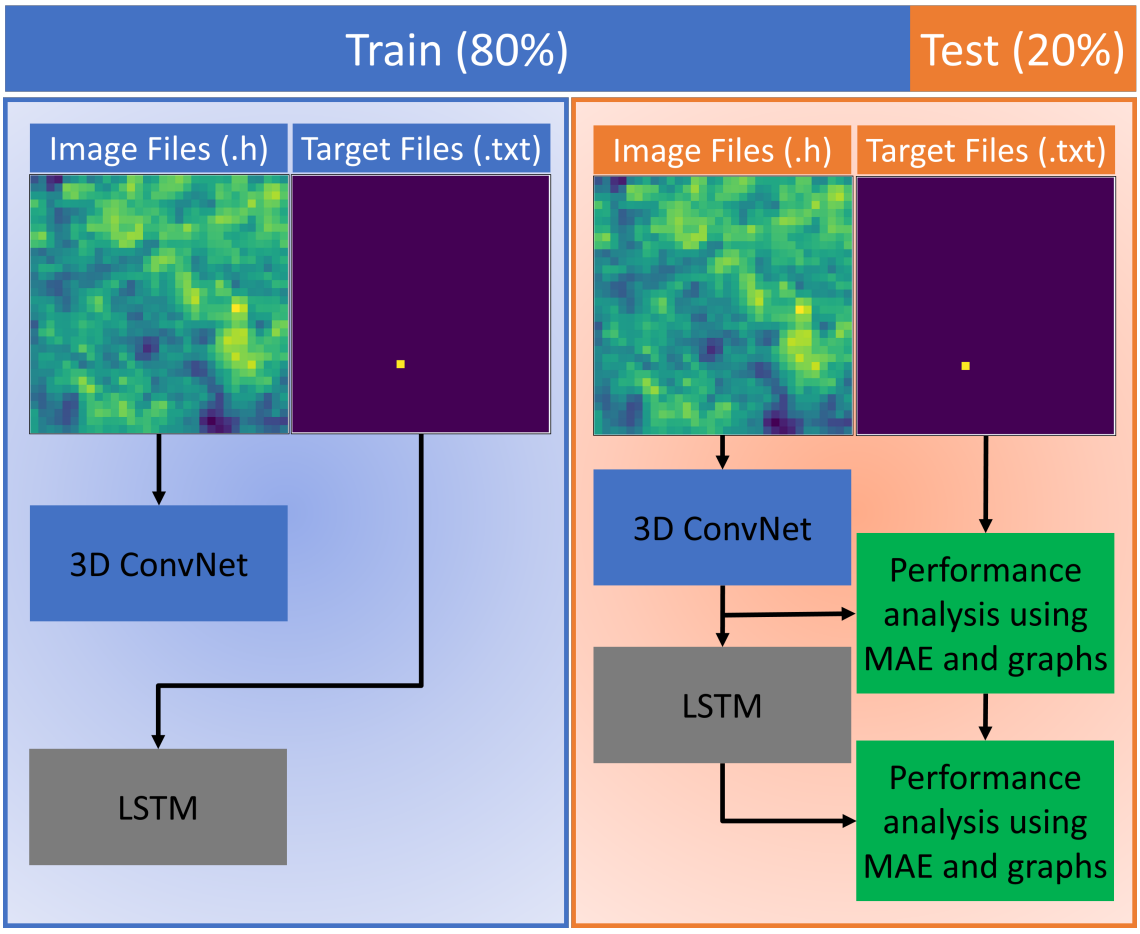


Figure 6. Phase 1: ANN Pipeline

2. Configuration files - A custom and default configuration file were used. The custom configuration file points to the default configuration file and updates the parameters listed in the custom configuration file. The default configuration file is packaged within ASSET and contains a generic configuration.

3. Object file - This file identifies the number of objects and its attributes. For Phase 1, only one object was created and had either a linear or curved trajectory. Signals move linearly or with a slightly curved path at approximately 1.21 pixels per frame.

ASSET then saves all output image information as a `HDF5 (.h)` file and the associated object truth information as a `.txt` file. The ASSET `HDF5 (.h)` file contains all of the sensor image data, meaning there is one `.h` file for each scenario.

The dataset used for Phase 1 was 242MB in size and was customized given the assumptions presented for Phase 1. Initially, 3,000 scenarios with linearly moving objects and 3,000 scenarios with slightly curved moving objects were generated (see figure 2 for example trajectories). The dataset was then downsized to only keep scenarios with a object present in each frame. This complies with the assumption that an object is present in every frame of a scenario. Downsizing was needed due to ASSET's inherent object generation algorithm, as generated objects did not always appear in every frame of the scenarios.

The final dataset used to train the ANNs contained 2,786 scenarios (`.h` files). 1,724 of the scenarios had one moving object with a linear trajectory and 1,062 of the scenarios had one moving object with a curved trajectory. Each scenario had 16 frames (image sequence) and each image frame was 32x32 (width x height) pixels. The number of frames and image sizes were chosen to keep the neural network architectures small and reduce the training times needed.

Like the `.h` files, there is one object truth file (`.txt`) generated for each scenario.

Each object file contains 16 spatial coordinate pairs (one spatial coordinate pair for each frame in the input sequence), and the final dataset contains 2,786 object `.txt` files. Having labeled truth information is essential to the success of any supervised learning problem. As an example, table 1 shows the object column and row pixel coordinates for the linear and slightly curved trajectories corresponding to figure 2.

Dataset Pre-processing.

All dataset processing was perform in Python and used the 5,572 (2,786 `.h` and 2,786 `.txt`) files as input. The pre-processing steps are described in this section.

Step 1: Analyze .h and .txt Files.

Within the `.h` file is a key, saved as a `Dataset` named `CalRawData`. All of the required training information was extracted from `CalRawData`. The data is stored in the shape (frame, column, row), with each scenario having 16 frames, 32 columns, and 32 rows.

The `.txt` file contained nine columns and 16 rows of object information. The columns are: `Number, time, frame, row, column, signal, peak, NET, and SET`. The rows provided details of the object's position, signal level, and background noise in each frame. Training the 3D ConvNet only required using the `row and column` details. An example object (.txt) file is attached to appendix B.

Step 2: Import Into Python.

The `.h` files were used as the image data, and the object `.txt` files were used as the labeled data. First, the `.h` primitive data type was converted from unsigned int32 to signed float32. After conversion, the scenario information was saved into NumPy arrays. The NumPy arrays were then reshaped into the input shape needed

Table 1. Example Coordinates Found in .txt Files

	Linear		Slightly Curved	
frame	**column (x)**	**row (y)**	**column (x)**	**row (y)**
1	26.12	11.83	14.20	9.06
2	25.79	13.00	14.81	10.11
3	25.46	14.17	15.40	11.17
4	25.13	15.34	15.99	12.24
5	24.80	16.51	16.56	13.31
6	24.47	17.68	17.13	14.38
7	24.15	18.85	17.67	15.47
8	23.82	20.02	18.20	16.56
9	23.49	21.19	18.72	17.67
10	23.16	22.36	19.21	18.78
11	22.83	23.53	19.67	19.90
12	22.50	24.70	20.11	21.03
13	22.17	25.87	20.51	22.18
14	21.84	27.04	20.89	23.34
15	21.52	28.21	21.22	24.50
16	21.19	29.38	21.51	25.68

for a Keras Conv3D layer: (batch, conv_dim1, conv_dim2, conv_dim3, channels). This translates into: (number of scenarios, number of frames per scenario, pixel height, pixel width, number of channels). For reference, the `.h` dataset had the shape (2786, 16, 32, 32, 1) with the one gray-scale channel.

Next, the row and column information was imported from the object files, also as float32, into NumPy arrays. The object arrays were reshaped into (number of scenarios, number of frames per scenario*2). The "number of frames per scenario*2" represents each frame having one row and one column coordinate. Each scenario is comprised of a vector: $[x_1, y_1, x_2, y_2, ..., x_n, y_n]$ (n = number of frames per scenario).

Step 3: Create Train/Test Sets.

3D ConvNet Train/Test Sets.

The 3D ConvNet was trained using the standard train/test split of 80/20. After splitting, the image data training and testing values were normalized to a range

between [0, 1]. Normalizing ensures the pixel values are evenly evaluated regardless of spatial location [21]. These were the image data and label shapes after the split:

- **Training data**
  - conv3d_image_train: (2228, 16, 32, 32, 1)
  - conv3d_image_train: (2228, 32)
- **Testing data**
  - conv3d_label_test: (558, 16, 32, 32, 1)
  - conv3d_label_test: (558, 32)

LSTM Train Set.

Preparing the LSTM training set took several more steps compared to the 3D ConvNet. A LSTM learns from a sequence of inputs to predict the next output. There are no standard values for look back and prediction window lengths, as they both are dependent on the given problem and data available. Given that the dataset has 16 frames per scenario, the look back window was set to five and the prediction window was set to one. Having additional frames per scenario would have increased the look back window. Also, the LSTM was trained only on the training data available to the 3D ConvNet.

Here is the Python style pseudocode to create a LSTM training set:

Algorithm 1 created 24,508 scenarios from the 2,228 scenarios available in $conv3d\_y\_train$ set. The input parameters and training shapes resulting from Algorithm 1 are:

- Dataset = conv3d_label_train.reshape(2228, 16, 2)
- lb (look back window) = 5
- pw (prediction window) = 1

Algorithm 1 Pseudocode to create LSTM training set

```
procedure CREATE_TRAINING(Dataset)
    lb = 5                                              ▷ look back window
    pw = 1                                              ▷ prediction window
    lookback_train, pred_train = [ ], [ ]               ▷ empty lists
    for i in number of scenarios do
        for j in number of frames do
            lookback_train.append(Dataset[i, j:(j+lb)])
            pred_train.append(Dataset[i, (j+lb):(j+lb+pw))
        end for
    end for
    return lookback_train, pred_train
end procedure
```

- lstm_lookback_train: (24508, 5, 2)
- lstm_pred_train: (24508, 2)

LSTM Test Set.

For testing, a combination of 3D ConvNet and LSTM predictions were used. The 3D ConvNet had the task of detecting the object using the first 10 frames, and the LSTM was given the task of predicting the future coordinates for the final six frames. The predictions were then compared to the truth coordinates.

The steps to process LSTM inputs are:

1. Take the last 10 (number of frames*2) values from the 3D ConvNet output. Only the last five coordinates will be used as LSTM inputs (based on the look back window). Output goes from shape (558, 32) $\Rightarrow$ (558, 10).

2. Reshape the 10 values again to match the input shape defined by the LSTM model. Shape goes from (558, 10) $\Rightarrow$ (558, 5, 2).

3. The LSTM then outputs the predicted coordinates for the following window(s). The number of predicted coordinates can be set by the user. For this research,

the LSTM prediction window was set to one, so the resulting output shape was (558, 2).

4. For additional LSTM predictions, continually use the last five frames to predict the following frame.

Analysis Strategy.

Phase 1 was evaluated by analyzing the ANN training loss plots, quantitatively calculating mean absolute error (MAE), and qualitatively examined by graphing the predicted coordinates.

ANN Training Loss Plots.

ANN training loss plots provided insight into the training process. Of interest are the number of epochs required for training and the minimum/maximum loss metrics achieved. The validation line also a good indicator of test set performance.

Mean Absolute Error.

Performance results were quantitatively evaluated by plotting and then calculating the median MAE of the histograms. As opposed to mean, the median value is less influenced by extreme outliers and is preferable when comparing different histograms. A perfect prediction would equate to a total MAE of `0`.

MAE was calculated by sequentially placing all x and y coordinates into separate arrays. The set $predicted\_X$ contained all predicted x coordinates, and the set $predicted\_Y$ contained all the predicted y coordinates. The same was done for the truth x and y coordinates and placed into sets $truth\_X$ and $truth\_Y$. The formulas below detail the steps used to calculate MAE between the x and y coordinates:

$$MAE\_X = \frac{1}{n}\sum_{i=1}^{n} |truth\_X_i - predicted\_X_i| \tag{1}$$

$$MAE\_Y = \frac{1}{n}\sum_{i=1}^{n} |truth\_Y_i - predicted\_Y_i| \tag{2}$$

$$MAE\_Total = MAE\_X + MAE\_Y \tag{3}$$

where i represents the iteration through each element of the set and n represents the total number of elements in the set. Finally, MAE_X and MAE_Y were combined to create the total MAE for one scenario. This process was repeated for each test set scenario and then graphed on a histogram.

Graph Predictions.

ANN prediction results were also qualitatively evaluated by graphing the predicted and truth coordinates on the same plots. Visual representations of the predictions provide instant feedback into the ANN's prediction performance.

3.3 Phase 2

Phase 2 was the next cycle of this research. Phase 2 removed several assumptions presented in Phase 1 to create a more difficult problem. As a result the 3D ConvNet and LSTM architectures were no longer viable solutions. By leaning on the works published in the medical AI field, U-Net was chosen as the solution.

For Phase 2, U-Net was tasked with receiving 32 images (one scenario) as input and producing 32 semantic image segmentation masks. There is one semantic image segmentation mask for each frame (or image) in a scenario. Each pixel within the mask was either identified as background or object (see figure 7 for a visual example).

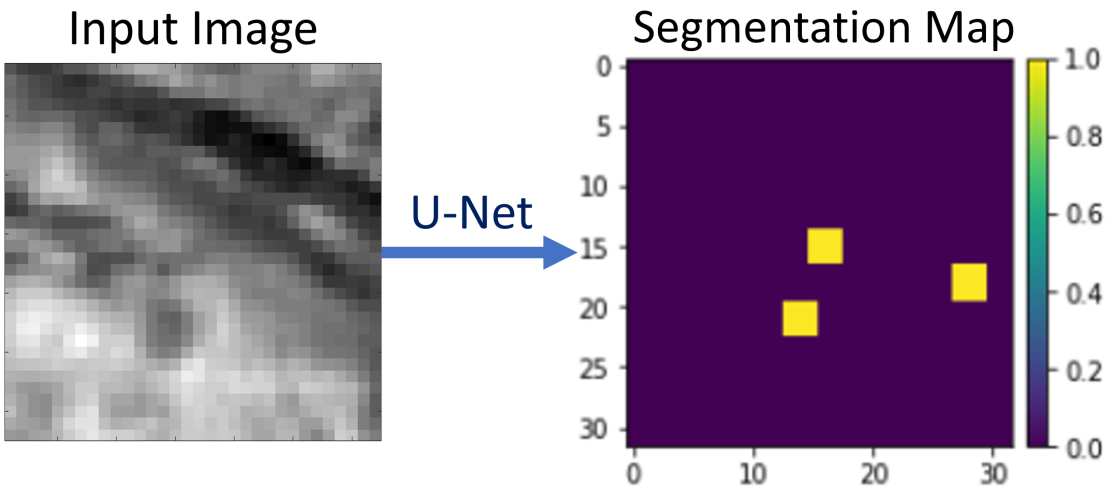


Figure 7. Phase 2: U-Net example

Phase 2 was designed around the two research questions:

1. Which ANN architectures could be used?
2. How does the object SNR, overall background to object pixel ratio, and the number of scenarios within the dataset impact ANN performance?

Experiments 1-3 used a control dataset to identify best ANN architecture, and experiments 4-6 used a control ANN architecture to identify dataset impacts (see figure 8).

Assumptions.

Phase 2 relaxed several small scaled problem constraints previously identified in Phase 1. The updated assumptions are provided below. Note: unlike Phase 1, Phase 2 did not include object prediction.

- ASSET was used to create EO/IR motion imagery data with noise, clutter, and atmospheric conditions representative of a generic space-based satellite sensor.

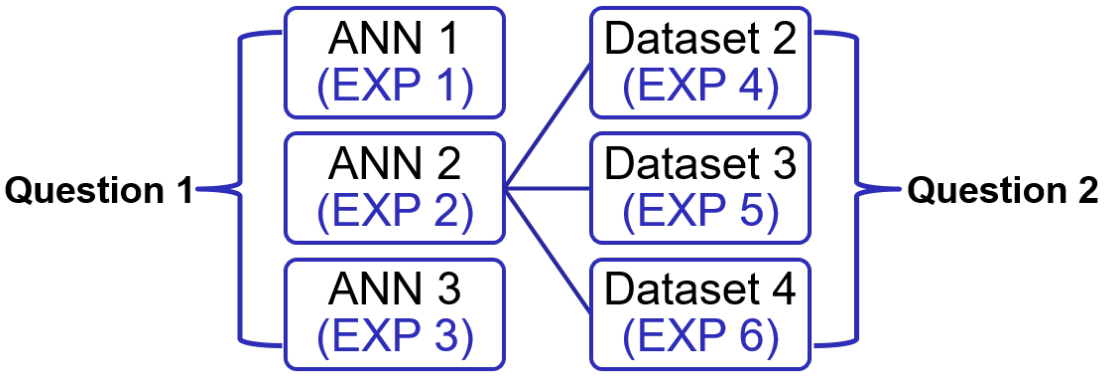


Figure 8. Phase 2: Research questions and experiments

- There was no attempt to reduce noise and clutter in the motion imagery generated by ASSET.
- In the scenarios considered, atmospheric effects such as clouds and attenuation were set to nominal and did not influence object detection.
- Allow scenarios to have multiple and/or zero objects in any given image frame. For example, an object may appear in frame 2 and disappear in frame 10. There will be no constraints to keep the object in all frames of a scenario.
- Scenarios will have randomized background images (figure 11 showcases the randomized background images).
- Objects will be initialized with random trajectories, speed, and acceleration. After initialization, the object will maintain a constant speed and acceleration, and trajectory will be formed based on those two parameters. As an example, if an object is moving right, it will continue to moving right at a constant speed and acceleration. The object will never veer left or slow down.
- Removed object prediction goal, focused on detection.
- Even with these constraints, the unresolved objects were not visibly apparent

within an image frame due to the contrast between their relatively low signal and bright background (see figure 3).

ANN Architecture.

Three U-Net variants were chosen as the main ANN architectures for Phase 2. These variants were created to answer the second research question, "Which ANN architectures could be used?"

All U-Nets were based on the works presented by Ronneberger et al. [7] and Milletari et al. [22]. Ronneberger et al. developed the popular U-Net architecture. Milletari et al. showed using the Dice-coeffiecient as the loss function on the last layer performs well on image segmentation tasks with strongly imbalanced binary classes. The Dice-coefficient function was calculated using The true positive (TP), false positive (FP), and false negative (FN) values and is shown in equation 4. A DSC of 1.0 represents a perfect match between the truth and predicted sets.

$$DSC = \frac{2 * TP}{2 * TP + FP + FN} \tag{4}$$

Here are the descriptions of the three ANN architectures created for Phase 2:

1. ANN 1 (2D_U-NET_Dice) - This ANN was modeled after the original U-Net and is considered a FCN. The 2D U-Net performs a convolution over the width and height of the input volume. Figure 9 shows the network architecture using Keras functional application program interface (API) style notation. There were a total of 34,512,193 trainable parameters in ANN 1. The input shape matched the Keras 4D tensor (batch, rows, cols, channels) and the output shape was (batch, new_rows, new_cols, filters). To keep figure 9 relatively small, these additional hyperparameters were not shown in the figure:

- All Conv2D layers, except Layer_36, used (kernel_size = 3, padding = 'same', activation = 'relu').
- Layer_36 used (kernel_size = 1 and padding = 'valid'). Due to having only one class, the sigmoid activation function was used on this final layer. Dice-coefficient was used as the loss function.
- All MaxPooling2D layers used (pool_size = 2 and strides = 2)
- All UpSampling2D layers used (size = 2).

2. ANN 2 (3D_U-NET_Dice) - This ANN was almost an exact copy of ANN 1. The 3D U-Net performs a convolution over the width, height, and depth/frames of the input volume. Using figure 9 as a reference, simply replace all 2D layers with 3D layers (Conv2D $\rightarrow$ Conv3D, MaxPooling2D $\rightarrow$ MaxPooling3D, UpSampling2D and $\rightarrow$ UpSampling3D). There were a total of 103,522,753 trainable parameters in ANN 2. The input shape matched the Keras 5D tensor (batch, depth, rows, cols, channels) and the output shape was (batch, new_depth, new_rows, new_cols, filters). The hyperparameters used for ANN 2 were:
   - Conv3D, MaxPooling3D, and UpSampling3D used the same hyperparameters, loss function, and activation function as ANN 1.
   - Concatenation layers used axis = 4, instead of axis = 3.
3. ANN 3 (3D_U-NET_BCE) - This ANN was created solely to compare the performance difference between the loss functions Dice-coefficient and binary cross-entropy (BCE). ANN 3 uses the exact same hyperparameters as ANN 2 and has the same number of trainable parameters. The only difference is that ANN 3 was compiled using binary cross-entropy as the loss function and accuracy was used as the metric.

4. The architecture used for ANN 2 and 3 can be viewed at appendix F.

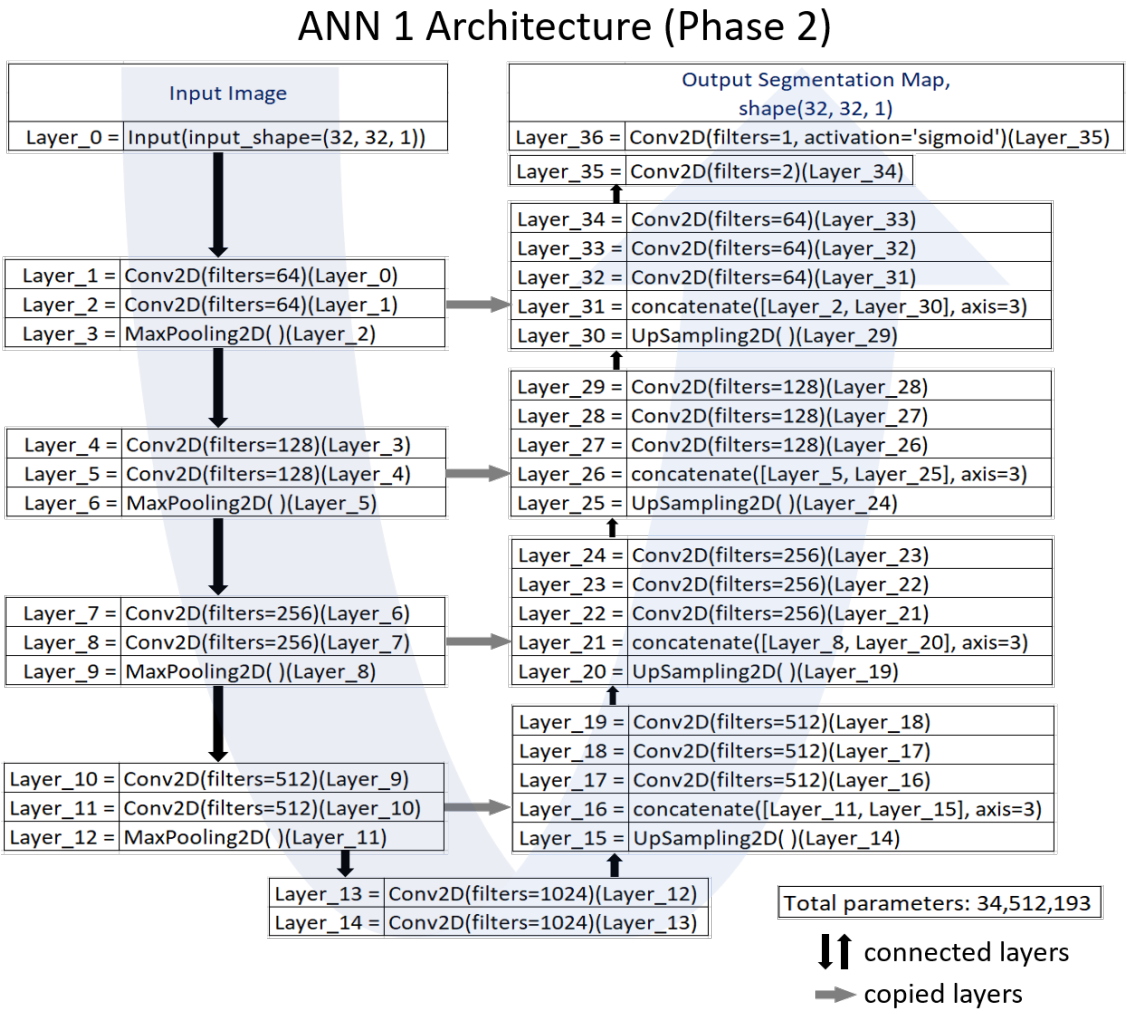


Figure 9. Phase 2: 2D U-Net architecture

All ANNs were compiled using the 'Adam' optimizer with a beta 1 of 0.9, beta 2 of 0.999, epsilon set to 'None', decay rate of 0.0, and amsgrad set to 'False'. The LR was adjusted pending the dataset used and was inversely proportional to the dataset size. The datasets with 500/600 scenarios used a learning rate of 0.0001 and the dataset with 3,000 scenarios used a learning rate of 0.00001.

Dataset Generation.

Similar to Phase 1, ASSET needed three sources of information for Phase 2:

1. Background image - Each scenario used a different background image for this phase. These images were created by randomly taking 32x32 pixel samples from the original background image. The scenario characteristics were also randomly assigned within ASSET to emulate different times of day, time of year, temperature, and satellite longitude. Appendix E shows a snapshot of 40 scenarios with random satellite generation parameters. The largest dataset required 3,000 scenario generation parameters.

2. Configuration files - A new custom configuration file was created for Phase 2.

3. Object file - This file identifies the number of objects and its attributes. For Phase 2, all objects were generated using the same base configuration; because objects are generated randomly, their trajectories are different in each dataset but have similar speed, acceleration, and signal levels.

Phase 2 was also tasked with answering the third research question, "how does the object SNR, overall background to object pixel ratio, and the number of scenarios within the dataset impact ANN performance?". Several variables were hypothesized as having a significant impact to ANN performance. These were the variables changed for each dataset generation:

- Object signal-to-noise ratio (SNR) - Users can define the object's SNR prior to scenario generation. The object's SNR takes into account both the background image and noise signals present in the scenario.

- Overall background to object pixel ratio - This ratio is influenced by the number of objects present in the scenario. A higher number of objects equate to a lower ratio.

- Number of scenarios - All scenarios are the same size: 32 frames, 32 pixel width, and 32 pixel height. The number of objects within the scenario does not impact the scenario size.

In total, four datasets were created for Phase 2. Each dataset was successively designed based on the knowledge of ASSET to test the boundaries of an ANN's performance at detecting unresolved objects.

All datasets were generated to have 32x32 (row x column) pixels per image and with 32 image frames per scenario. 32x32 pixels per image was chosen to reduce ANN training times and reduce the overall background to object pixel ratio. For example, a 64x64 image has 4,096 pixels, whereas a 32x32 image has 1,024 pixels. Generation of the scenario backgrounds and object trajectories were determined by ASSET's randomization algorithm (see figure 11 for an example).

In addition, each dataset contains image masks identifying the locations of the objects within the images. Image masks are needed by U-Net to perform training, and they are the same dimensions as the input image. For this research, a 1-pixel mask was created around the object using the object locations found in the `.txt` files. Note: the image mask is not representative of the actual object signals found within the scenario, objects are not 3x3 pixels in size. This naive solution simply serves to decrease the background to object pixel ratio, thereby reducing the difficulty of the problem. Figure 10 shows a visual of implementing this solution by increasing the mask of the object from 1-pixel to 9-pixels.

Figure 11 shows the randomization of the image backgrounds and object motion paths found in the scenarios. The datasets in the figure were aggregated based on scenario generation parameters.

Here, the datasets are described in the order created:

1. Dataset 1 (500Scenarios_10-Objects_20SNR) - This was the initial, baseline,

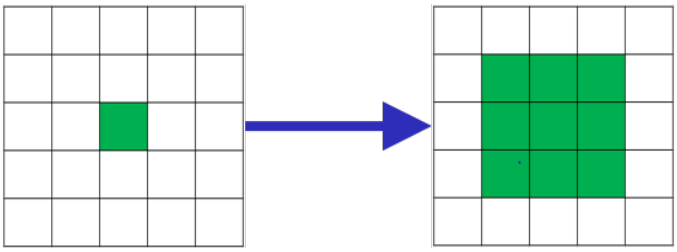


Figure 10. Phase 2: Object masking visual

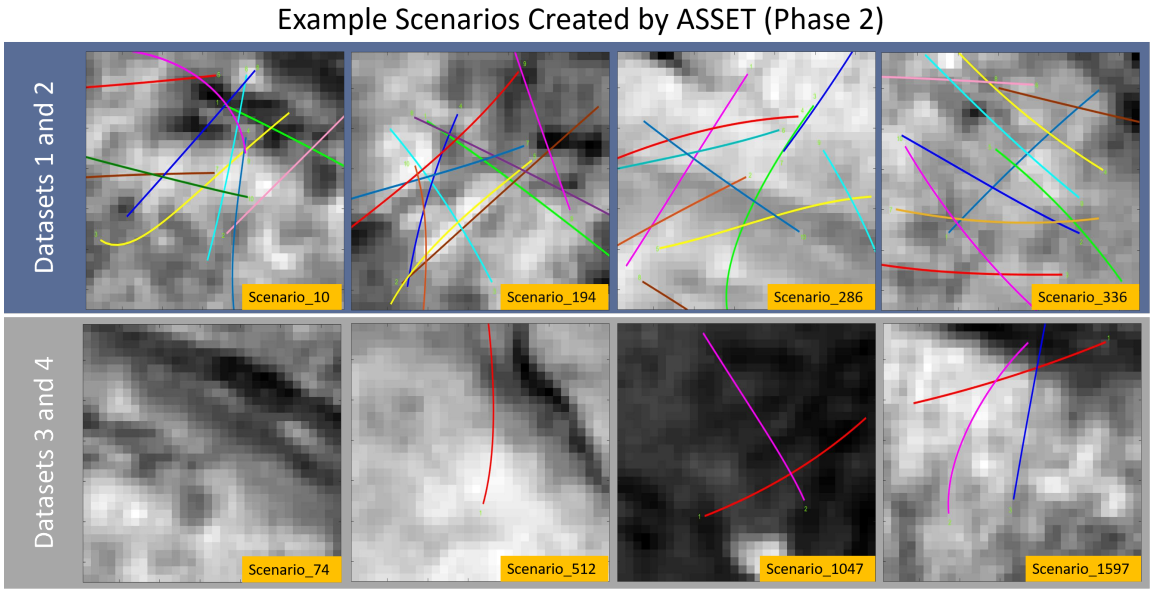


Figure 11. Phase 2: Example backgrounds and object motion paths

dataset containing 500 scenarios and was 86 MB in size. The intent was to maximize the ANN's probability of detecting objects. This was achieved by setting the maximum number of objects generated to 10 and with each object having a SNR of 20. The overall background to object pixel ratio was 14:1 (15,320,927:1,063,073 pixels).

2. Dataset 2 (500Scenarios_10-Objects_10SNR) - This was the second dataset created and was created with the same generation parameters as Dataset 1, except for the object's SNR. The SNR was dropped from 20 to 10 to test if the ANN could detect objects with a lower SNR. Dataset 2 was also 86 MB in size, and the overall background to object pixel ratio was 14:1 (15,312,432:1,071,568 pixels).

3. Dataset 3 (3000Scenarios_0-5-Objects_20SNR) - This was the third dataset created. This dataset lowered the maximum number of objects present from 10 to 5. Lowering the number of objects reduces the background to object pixel ratio, thus increasing the difficulty of the problem. To compensate for the increased difficulty, additional scenarios were created. Objects' SNR remained set to 20.

    - 500 scenarios were generated with zero objects
    - 500 scenarios were generated with one object
    - 500 scenarios were generated with two objects
    - 500 scenarios were generated with three objects
    - 500 scenarios were generated with four objects
    - 500 scenarios were generated with five objects

    In total, 3,000 scenarios where generated with the maximum number of objects in any scenario being five. Dataset 3 was 472 MB in size, and the overall background to object pixel ratio was 60:1 (96,637,352:1,666,648 pixels).

4. Dataset 4 (600Scenarios_0-5-Objects_20SNR) - This was the fourth and final dataset created. This dataset was created to determine if lowering the training data would still provide enough information to train an ANN. Scenarios were selected from Dataset 3 to create Dataset 4.

    - 100 scenarios were selected with zero objects
    - 100 scenarios were selected with one object
    - 100 scenarios were selected with two objects
    - 100 scenarios were selected with three objects
    - 100 scenarios were selected with four objects
    - 100 scenarios were selected with five objects

    In total, 600 scenarios where selected with the maximum number of objects in any scenario was five. Dataset 4 was 98 MB in size, and the overall background to object pixel ratio was 57:1 (19,323,398:337,402 pixels).

Analysis Strategy.

Similar to Phase 1, Phase 2 used ANN training loss plots and qualitatively evaluated the ANN predictions using plots. Quantitative evaluations were performed using the DSC, accuracy, and calculating the precision-recall (PR) area under the curve (AUC).

ANN Training Loss Plots.

ANN training loss plots provided insight into the training process. Of interest are the number of epochs required for training and the minimum/maximum loss metrics achieved. The validation line also a good indicator of test set performance.

Dice-coefficient Score and Accuracy.

Performance results were quantitatively evaluated by calculating the average DSC (equation 4) or average accuracy (equation 5. A perfect prediction would have a DSC or accuracy equal to 1.0. Accuracy was calculated using the TP, FP, true negative (TN), and FN values.

$$Accuracy = \frac{(TP + TN)}{(TP + FP + TN + FN)} \tag{5}$$

When working with strongly imbalanced classes, it is good to note that using accuracy to measure performance can be misleading. For example, here are the results of calculating DSC and accuracy using representative values as seen from the experiments.

- TP = 4,802
- FP = 12,277
- TN = 3,052,327
- FN = 207,394

$$DSC = \frac{2 * 4,802}{2 * 4,802 + 12,277 + 207,394} = 0.04$$

$$Accuracy = \frac{(4,802 + 3,052,327)}{(4,802 + 12,277 + 3,052,327 + 207,394)} = 0.93$$

This example shows how accuracy can be near 1.0, while DSC is near zero.

Precision-recall AUC.

An ideal ANN would achieve a PR AUC calculation of 1.0, and worst case performance would equate to a 0.0. PR AUC was chosen as the quantitative metric over receiver operating characteristic (ROC) curves due to the strong class imbalance present in all datasets.

ROC curves are created using the true positive rates (TPRs) and the false positive rate (FPR)s (FPRs). Since 90%+ of the pixels is background, ROC AUC scores will inherently score high. On the other hand, PR AUC calculations also account for the false negative rates (FNRs), which account for miss-classifying objects as background. TP, FP, TN, FN valuess were used to calculate PR AUC.

$$PR = \frac{Precison}{Recall} = \frac{PositivePredictedValue}{TruePositiveRate} = \frac{\frac{TP}{TP+FP}}{\frac{TP}{TP+FN}} = \frac{TP+FN}{TP+FP}$$

To calculate TP, FP, TN, and FN, predictions on the test set were repeatedly made while adjusting the threshold from 0.0 to 1.0 in 0.01 increments. These numbers were then stored in a dataframe. Table 2 displays the first five rows of an example dataframe.

Table 2. Example Dataframe Values

threshold	TP	FP	TN	FN	recall	precision
0.00	68992	3863168	0	0	0.983	0.018
0.01	53592	1862234	2000934	15400	0.747	0.028
0.02	36202	909142	2954026	32790	0.502	0.038
0.03	27180	564903	3298265	41812	0.378	0.0457
0.04	22649	420006	3443162	46343	0.316	0.051

The PRC is then graphed after the dataframe is fully populated. AUC can then

be calculated using the scikit-learn and NumPy libraries [23, 24] (see Algorithm 2 for an example).

Algorithm 2 Code to calculate PR AUC using Python style notation

```
x = dataframe['recall']
y = dataframe['precision']
print('PR AUC (using trapezoid rule) =', np.trapz(x,y))
```

Graph Predictions.

ANN prediction results were also qualitatively evaluated by graphing the predicted and truth images side by side. Visual representations of the predictions provide instant feedback into the ANN's prediction performance.

IV. Results & Analysis

4.1 Overview

This chapter describes the training results and presents the prediction results for Phases 1 and 2. The results have been grouped by phases and will be presented accordingly.

4.2 Phase 1

For Phase 1, the 3D ConvNet had the goal of predicting the object's location within each frame. The LSTM had a goal of predicting the object's location in the sixth image frame, given five previous image frame locations.

Both ANNs were initialized with the Keras callbacks EarlyStopping and ModelCheckpoint. These two callbacks were used to pinpoint the required number of epochs needed for training. During model training, 'validation_split' was set to 0.1, which reallocates 10% of the training set as the validation set. Creating the validation set from the training set allowed the testing set to remain untouched until final performance evaluation. As a reminder, MSE was used as the loss function between the training and validation sets. For the 3D ConvNet, EarlyStopping monitored validation loss and patience was set to 300. For the LSTM, EarlyStopping also monitored validation loss and patience was set to 20. This allowed us to set the training epochs excessively high without having to guess a good number. For both models, ModelCheckpoint monitored validation loss and only saved the best model, which disregarded the unnecessary training epochs shown in figures 12 and 16.

3D ConvNet Training Results.

Figure 12 displays the training results carried out to 1,000 epochs. The training curve decreases from 160 down to 10 MSE in under 100 epochs; however, it took another 300 epochs before reaching near 0 MSE. Monitoring the validation loss provides insight into the ANN's performance on the test set. A near 0 MSE implies the ANN will also perform well on the test set.

Finding the correct number of training epochs took several tries. The first few training sessions had the max number of training epochs set to 200 and always produce unusable results. A hint for future work: try increasing the training epochs first before adjusting other hyperparameters.

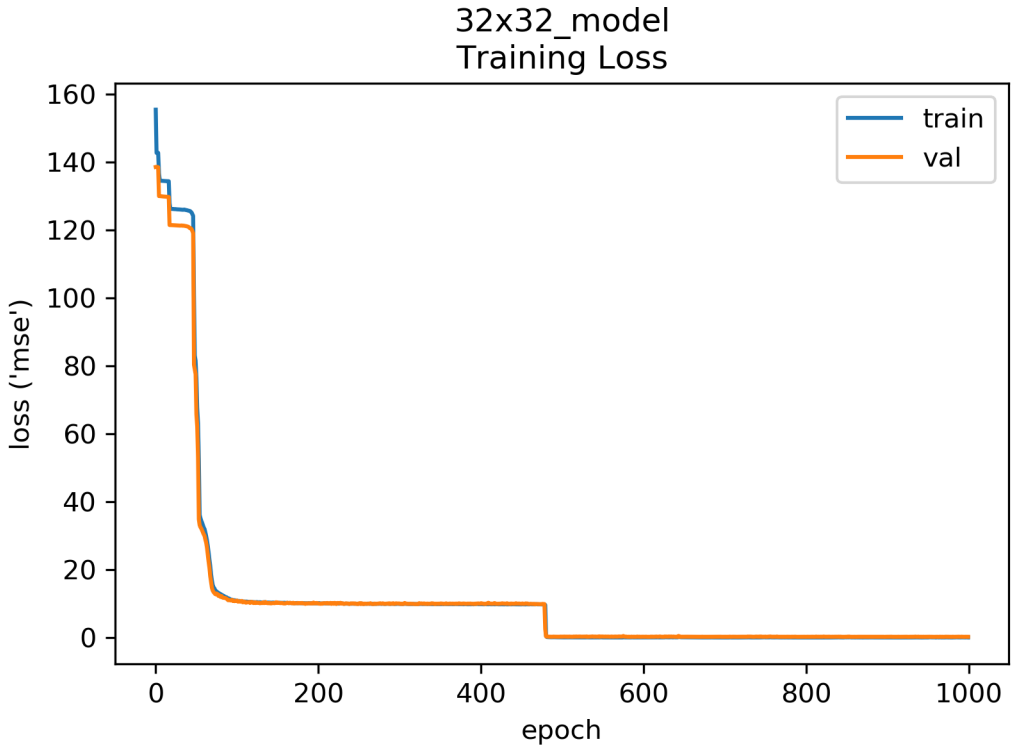


Figure 12. Phase 1: 3D ConvNet training loss plot

3D ConvNet Quantitative Evaluation.

Figure 13 shows the 3D ConvNet MAE results from all the test scenarios. MAE was calculated using the predicted and truth coordinates. The calculated median value found in the histogram was 0.31.

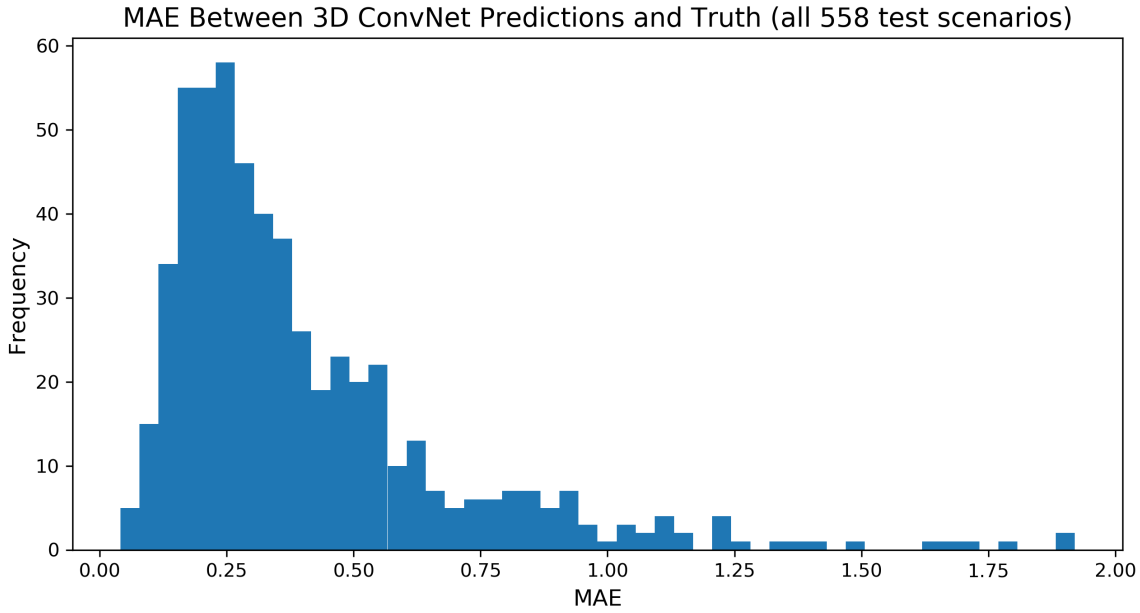


Figure 13. Phase 1: 3D ConvNet MAE histogram

For reference, had the 3D ConvNet always predicted the center pixel as the object's location, where x and y equaled 16, the calculated median value would have been 9.95 (see figure 14 for the side by side comparison of the 3D ConvNet and center pixel prediction MAE histograms). The 3D ConvNet model outperforms the trivial solution of always predicting the center pixel.

3D ConvNet Qualitative Evaluation.

Figure 15 displays the 3D ConvNet's predictions on the first four test set scenarios. The predicted coordinates are close to the truth coordinates, which supports the calculated median MAE of 0.31.

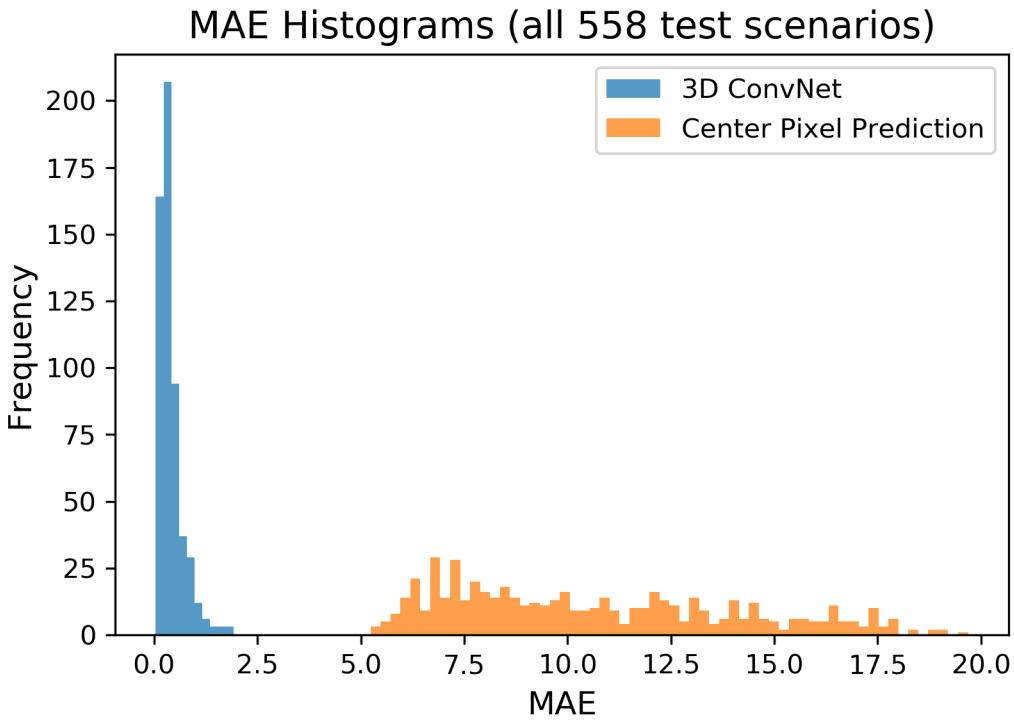


Figure 14. Phase 1: 3D ConvNet vs center pixel MAE histogram

Appendix C provides additional test set predictions.

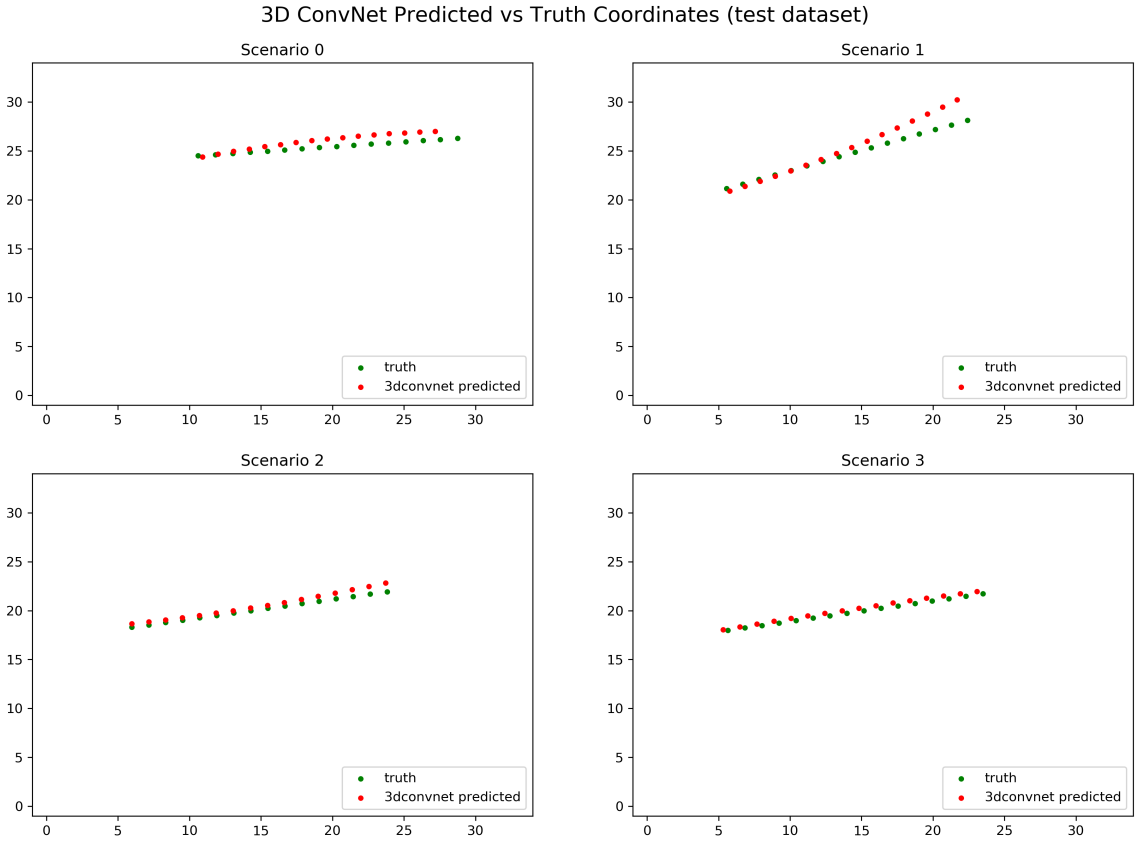


Figure 15. Phase 1: 3D ConvNet visual of the predictions

LSTM Training Results.

Figure 16 displays the training results carried out to 63 epochs. The training curve decreases to near 0 MSE at 43 epochs. The LSTM reaches near 0 validation loss, which indicates the ANN will perform well on the test set.

The LSTM was trained in less time than the 3D ConvNet. This is due to the type of input being fed into the two networks. LSTM received vector arrays, while the 3D ConvNet received image pixels.

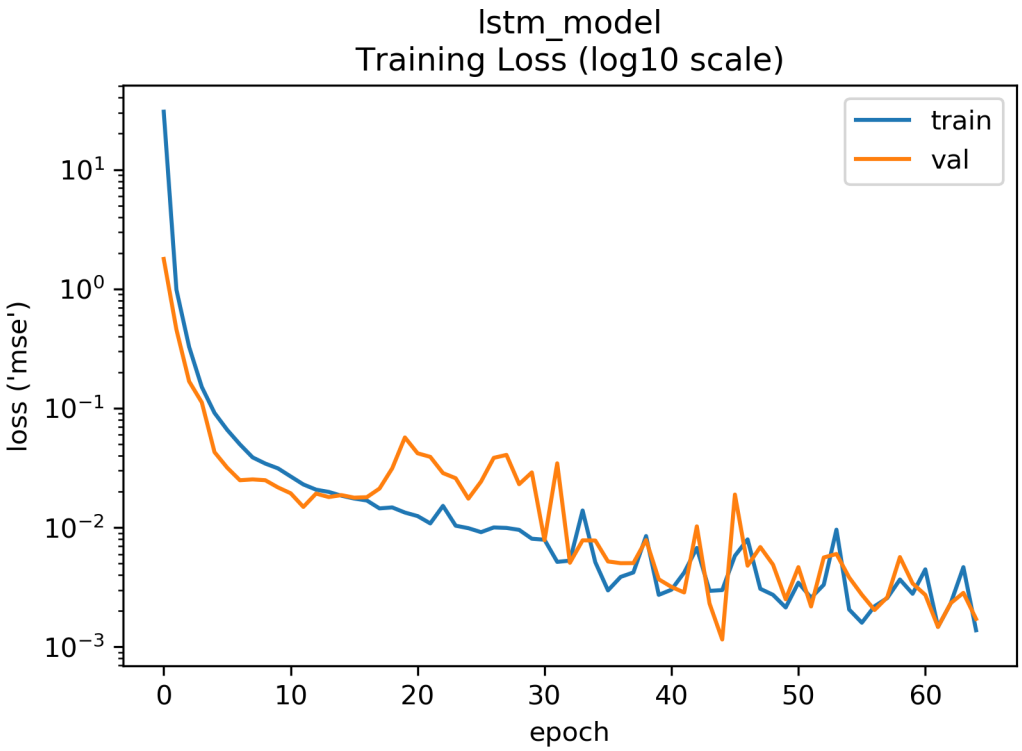


Figure 16. Phase 1: LSTM training loss plot (log scale)

3D ConvNet+LSTM Quantitative Evaluation.

Figure 17 shows the results of calculating the MAE between the 3D ConvNet+LSTM predicted and truth coordinates for all test scenarios. Based on the histograms, the 3D ConvNet+LSTM MAE also maintains a skewed right distribution with a calculated median at 0.45. Adding the LSTM predictions resulted in an increase in MAE by 45% (0.14), when compared to the 3D ConvNet MAE of 0.31. For comparison, figure 18 displays 3D ConvNet and 3D ConvNet+LSTM side by side.

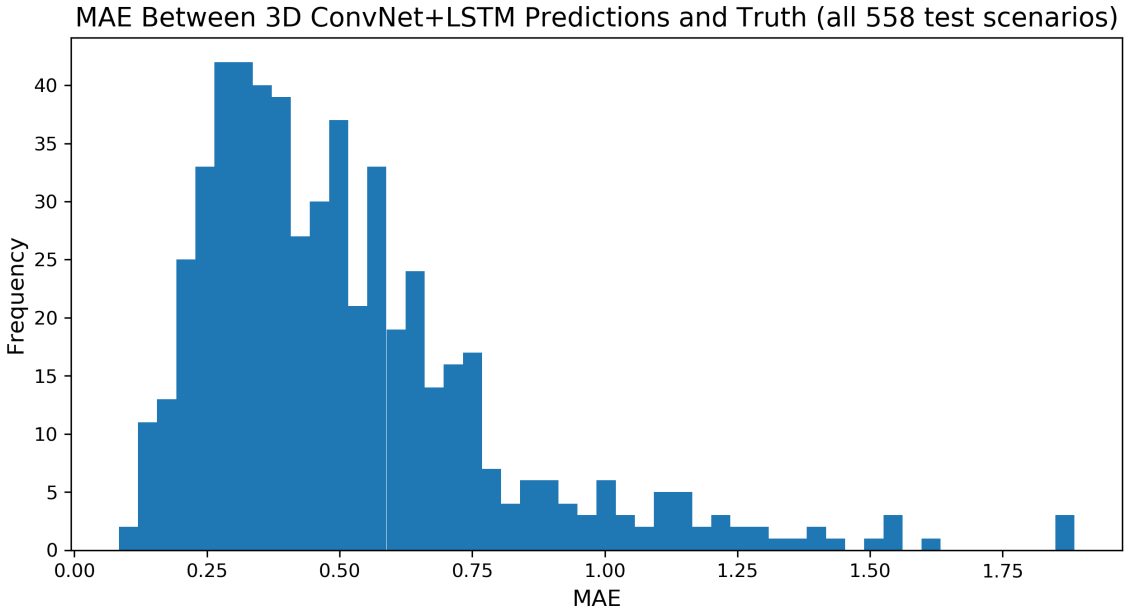


Figure 17. Phase 1: 3D ConvNet+LSTM MAE histogram

3D ConvNet+LSTM Qualitative Evaluation.

Figure 19 displays the LSTM's predictions on the first four test set scenarios. Appendix D provides additional test set predictions. The predictions made by the LSTM continue the 3D ConvNet trajectories as expected. There is a noticeable drop in performance when comparing the results to figure 15.

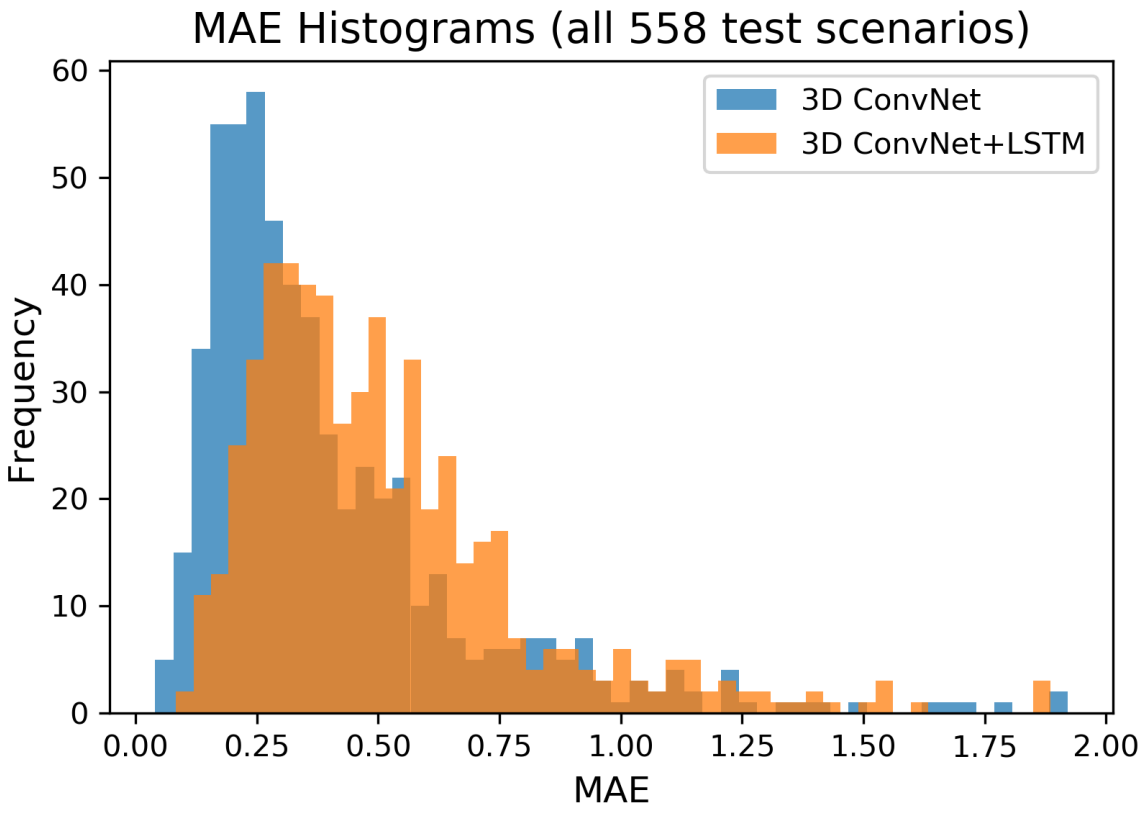


Figure 18. Phase 1: 3D ConvNet and 3D ConvNet+LSTM MAE histogram

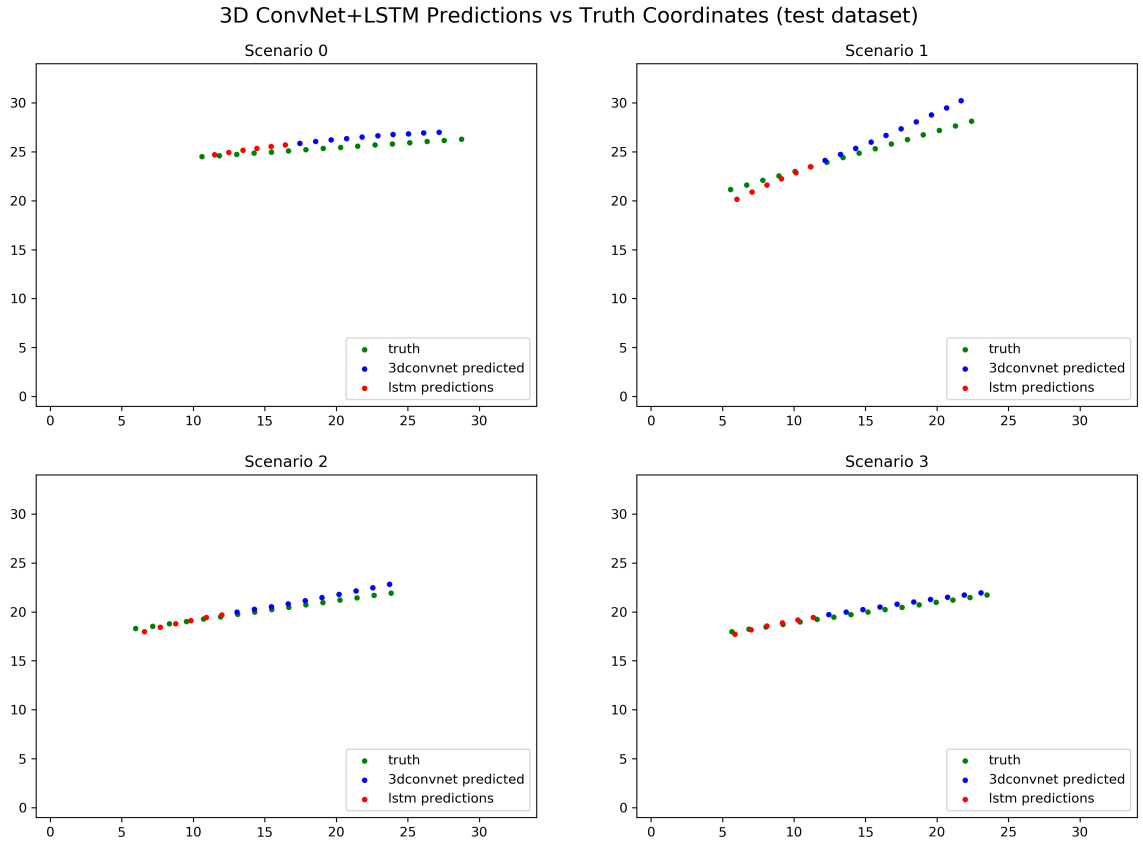


Figure 19. Phase 1: 3D ConvNet+LSTM visual of the predictions

4.3 Phase 2

U-Net was given the task of outputting a segmented image map when given an IR image. As a reminder, Phase 2 was designed around the two research questions:

1. Which ANN architectures could be used?

2. How does the object SNR, overall background to object pixel ratio, and the number of scenarios within the dataset impact ANN performance?

Experiments 1-3 used a control dataset to identify best ANN architecture, and experiments 4-6 used a control ANN to identify dataset impacts to ANN performance (see figure 8). The experimental combinations are shown in table 3. Please note the ranges of the colorbar next to the graphs, as the ranges are not all zero to one.

Table 3. Phase 2 Experimental Combinations

Experiment/Combination	Train set (60%)	Val set (20%)	Test set (20%)	Total (100%)
1.) Dataset 1 + ANN 1	10,240	2,560	3,200	16,000 images
2.) Dataset 2 + ANN 2	320	80	100	500 scenarios
3.) Dataset 3 + ANN 3	320	80	100	500 scenarios
4.) Dataset 4 + ANN 2	320	80	100	500 scenarios
5.) Dataset 5 + ANN 2	1,920	480	600	3,000 scenarios
6.) Dataset 6 + ANN 2	384	96	120	600 scenarios

U-Net Training Results.

For organization, this section displays all six experimental training results together; however, Experiments 1-6 were trained and evaluated chronologically. The Keras callbacks EarlyStopping, ReduceLROnPlateau, and ModelCheckpoint were implemented for the training process. The benefits of EarlyStopping and ModelCheckpoint was previously discussed in Phase 1 results section. ReduceLROnPlateau was used to improve the training process and helped prevent stagnant learning.

The training loss and performance metric graphs are shown for each experiment in figures 20 and 21. Experiments 1, 2, 4, 5, and 6 used the Dice-coefficient as

the training metric, and Experiment 3 used BCE as the training metric. For the experiments using the Dice-coefficient, an ideal model would have validation loss of 0 and a DSC of 1.0. For BCE, an ideal model would also have a validation loss of 0 and an accuracy of 100%.

In Experiment 1, ANN 1 completed 53 training epochs before EarlyStopping (patience = 8) stopped it. In 45 epochs, validation loss declined from 1.1 to 0.65, and the validation Dice-coefficient increased from 0.1 to 0.56.

In Experiment 2, ANN 2 completed 71 training epochs before EarlyStopping (patience = 8) stopped it. In 63 epochs, validation loss declined from 1.1 to 0.17, and the validation Dice-coefficient increased from 0.1 to 0.9. The training results from experiments 1 and 2 were promising and indicated image segmentation is possible using data generated by ASSET. Using the Dice-coefficient as a metric, Experiment 2 outperforms Experiment 1 by 62%. Based on this, the remaining experiments used 3D U-Nets as opposed to 2D U-Nets.

In Experiment 3, ANN 3 completed 18 training epochs before EarlyStopping (patience = 8) stopped it. In 10 epochs, validation loss declined from 0.25 to 0.23, and the validation accuracy remained constant at 0.935. Experiment 3 showed training a model using BCE with highly imbalanced classes does not work well. A naive solution of predicting all pixels as background pixels would also produce an accuracy of 93%. In the training and validation sets, there were 12,256,323 background and 850,877 object pixels. This gave a 14:1 ratio of background to object pixels. Experiment 3 proved using 3D U-Nets with the Dice-coefficient as the loss function will outperform 3D U-Nets using BCE as the loss function.

In Experiment 4, ANN 2 completed 88 training epochs before EarlyStopping (patience = 8) stopped it. The validation loss decreased from 1.3 to 1.1 and validation Dice-coefficient increased from 0 to 0.16 in 10 epochs. The slight improvements

to validation loss from epochs 10-80 prevented the training process from ending it sooner. Based on the training graphs, setting the objects' SNR to 10 was too low, and future experiments kept objects' SNR at 20.

In Experiment 5, ANN 2 completed 149 training epochs before EarlyStopping (patience = 32) stopped it. EarlyStopping patience was increased and the Adam learning rate was decreased for this experiment to compensate for the increased amount of training data. The validation loss decreased from 1.0 to 0.2 and validation Dice-coefficient increased from 0 to 0.9 in 117 epochs. Experiment 5 increased the training and validation sets' background to object pixel ratio to 58:1 (15,460,230:268,410 pixels), while also increasing the training data to compensate. Experiment 5's training results were comparable to Experiment 2, which provided confidence an ANN was still able to detect objects when the background to object pixel ratio increased.

In Experiment 6, ANN 2 completed 52 training epochs before EarlyStopping (patience = 32) stopped it. The validation loss decreased from 1.15 to 1.04 and validation Dice-coefficient increased from 0.025 to 0.051 in 13 epochs. Experiment 6's training results were similar to experiment 4's. The graphs quickly converged and then slightly improved until EarlyStopping ended the training process. Based on the training results, there is not enough information/scenarios present in Dataset 4 to train an ANN.

U-Net Quantitative Evaluation.

Phase 2 ANN test sets performance were quantitatively measured by plotting the PR curves and then calculating the AUCs using the trapezoid rule. An ideal ANN would achieve a PR AUC calculation of 1.0, and worst case performance would equate to 0.0.

Table 4 displays the ANNs' performance on the test sets for each experiment.

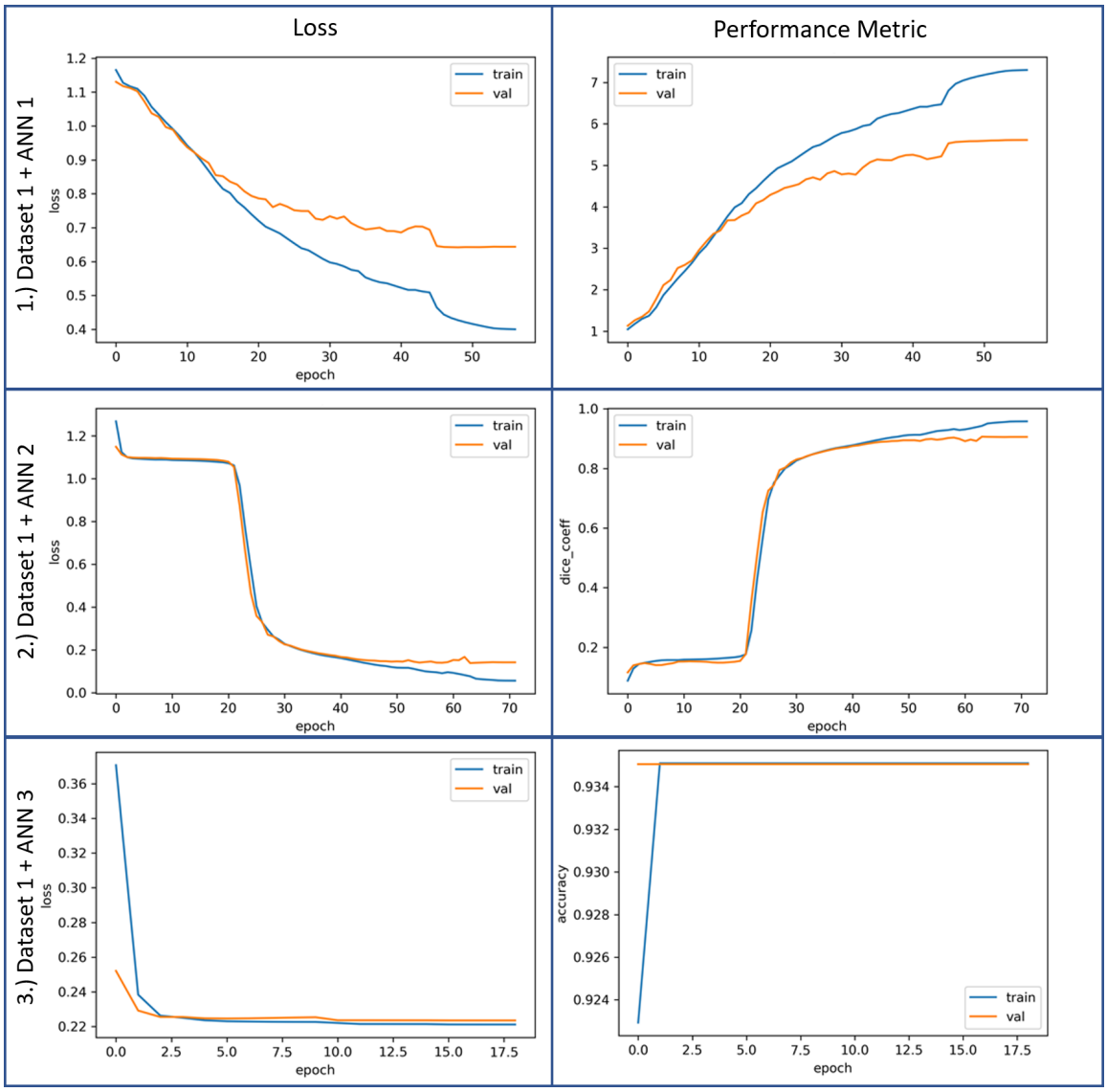


Figure 20. Phase 2: Experiments 1-3 training loss plots

Table 4. Phase 2: Summary of ANNs' Performance

Experiment/Combination	Avg. Dice	Avg. Accuracy	Avg. PR AUC
1.) Dataset 1 + ANN 1	0.567	N/A	0.553
2.) Dataset 2 + ANN 2	0.906	N/A	0.900
3.) Dataset 3 + ANN 3	N/A	0.935	0.085
4.) Dataset 4 + ANN 2	0.161	N/A	0.083
5.) Dataset 5 + ANN 2	0.894	N/A	0.878
6.) Dataset 6 + ANN 2	0.061	N/A	0.027

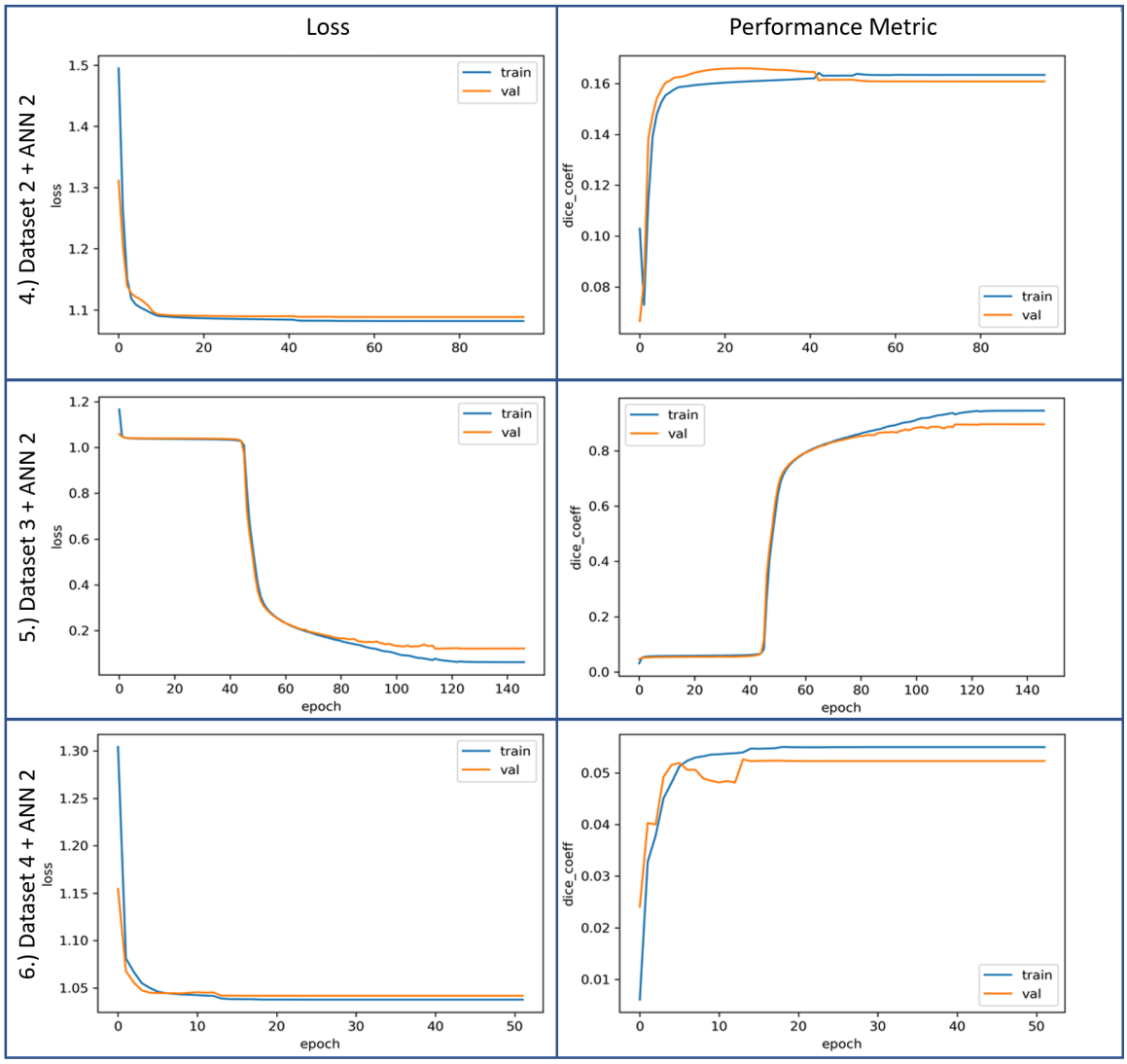


Figure 21. Phase 2: Experiments 4-6 training loss plots

Figure 22 displays the results of plotting and calculating the PR AUC from all six experiments.

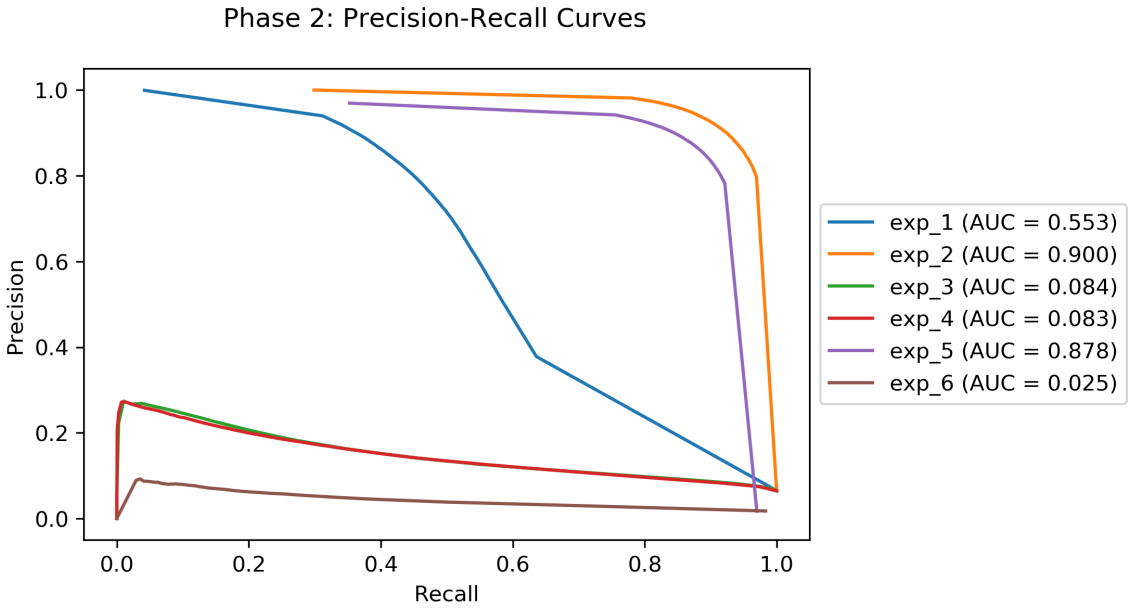


Figure 22. Phase 2: ANN PR AUC graph

In Experiment 1, ANN 1 had an average Dice-coefficient of 0.567 and average PR AUC of 0.553 on the test set scenarios. In Experiment 2, ANN 2 had an average Dice-coefficient of 0.906 and average PR AUC of 0.900 on the test set scenarios. Based on the average PR AUC, it is expected the model used in Experiment 2 will predict 61% more pixels correctly than the model used for Experiment 1.

In Experiment 3, ANN 3 had an average accuracy of 0.935 and average PR AUC of 0.085 on the test set scenarios. Experiment 3 confirmed using BCE as a loss function is not a viable U-Net variant when using data with strong class imbalances. Using the average PR AUC as a metric, Dice-coefficient was 10x more effective than BCE.

In Experiment 4, ANN 2 had an average Dice-coefficient of 0.161 and average PR AUC of 0.083 on the test set scenarios. This experiment reduced the objects' SNR in half, from 20 to 10. This resulted in ANN 2 being unable to correctly predict

the category of the pixels. The prediction results after reducing the object SNR were comparable to those of Experiment 3. There were no attempts to adjust the hyperparameters of ANN 2 to improve the prediction results.

In Experiment 5, ANN 2 had an average Dice-coefficient of 0.894 and average PR AUC of 0.878 on the test set scenarios. This experiment raised the object SNR back up to 20 and showed that increasing the dataset size compensates for the increased background to object pixel ratio. The prediction results of Experiment 5 were comparable to Experiment 2.

In Experiment 6, ANN 2 had an average Dice-coefficient of 0.061 and average PR AUC of 0.027 on the test set scenarios. This experiment showed that the increase in dataset size/number of scenarios are necessary when there is a high background to object pixel ratio. There were no attempts in improving ANN 2's performance on Dataset 4.

U-Net Qualitative Evaluation.

The ANN test set performance is also visually shown in figures 23 and 24. The figures were created by randomly sampling predictions using the test set scenarios and displaying the corresponding images of the truth masks. Note: figures 23 and 24 display the raw pixel value outputs from the ANNs. The raw pixel value outputs represent the probability of that pixel being an object. Lower values equate to lower probability.

The visual graphs show that the training results were good indicators of the ANNs' performance on the held-out test sets. The visual graphs also show there is a relationship between low average PR AUC and poor prediction performance. It can be seen the 3D U-Net performed better than the 2D U-Net, and the loss function using Dice-coefficient provides better ANN predictions than BCE.

Phase 2 Visuals of ANN Predictions

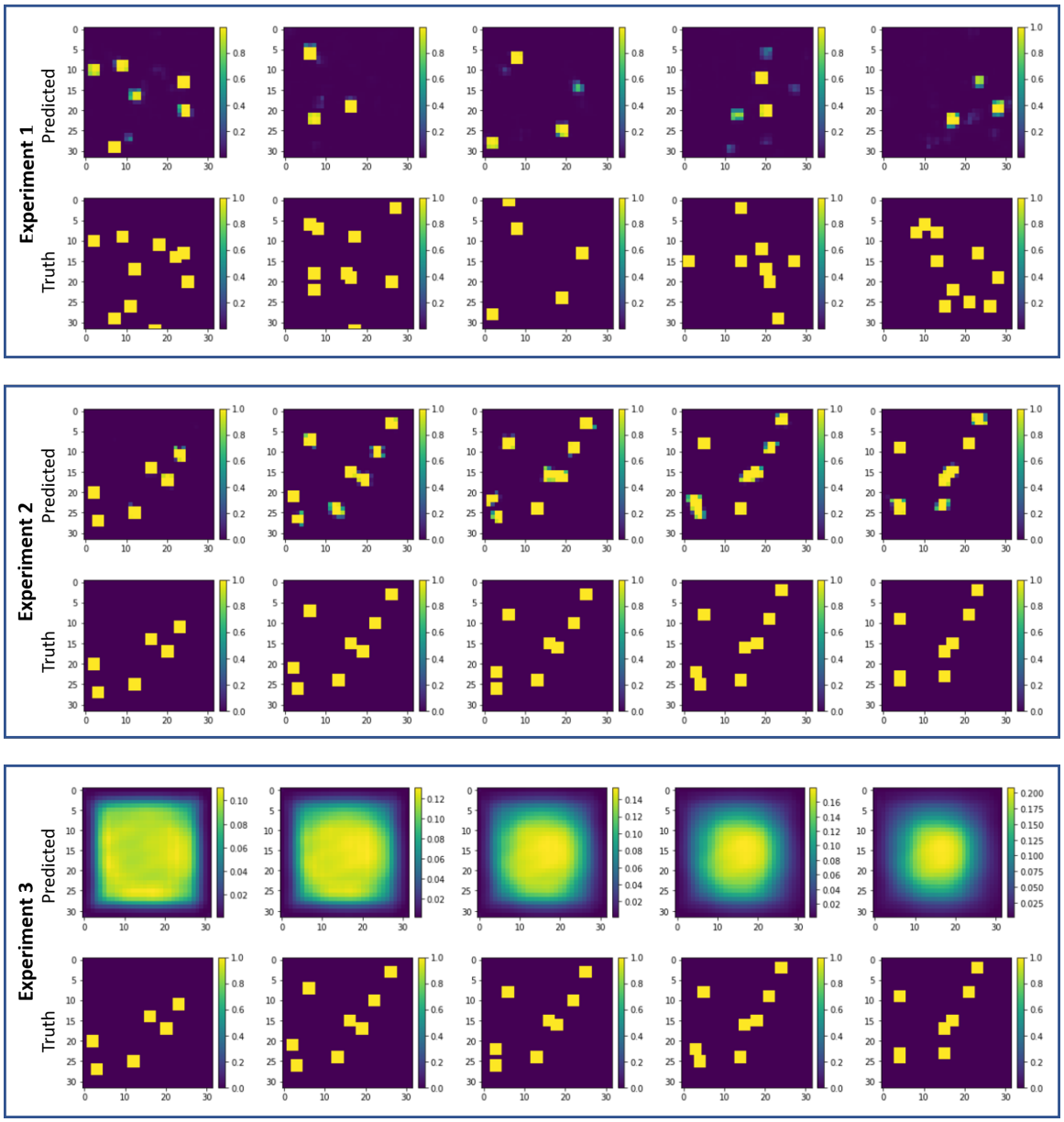


Figure 23. Phase 2: Experiments 1-3 visual results

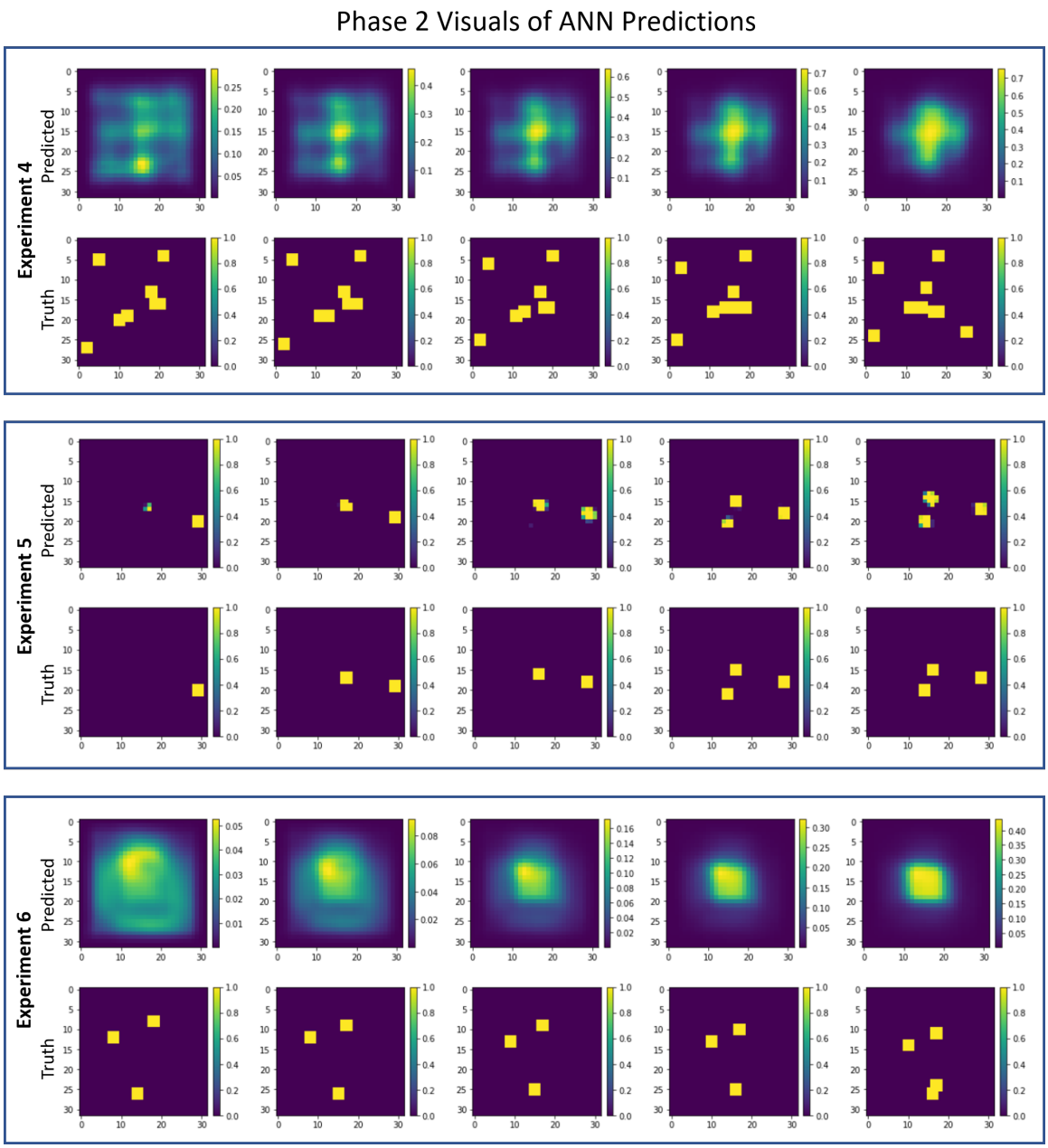


Figure 24. Phase 2: Experiments 4-6 visual results

Figure 23 shows the ANN predictions from Experiments 1 and 2 had comparable average PR AUC. The ANN in Experiment 1 correctly predicted 55% of pixels, while the ANN in Experiment 2 correctly predicted 90% of the pixels. The visual graphs showed that the ANN in Experiment 3 essentially predicted all pixels as background pixels to achieve the 93.5% average accuracy score.

Figure 24 shows the ANN predictions for Experiments 4 and 6 were not successful. The visual results also support the claim average PR AUC calculations are good indications of visual performance. Experiment 5 ANN predictions were on par with Experiment 2 ANN predictions. Both ANNs were able able to correctly label the pixels 88% of the time. Again, the additional scenarios compensated for the increased background to object pixel ratio.

4.4 Results Summary

For Phase 1, the 3D ConvNet was able to generally predict the object's location in each image frame, and adding the LSTM into the pipeline degraded the overall performance. More importantly, Phase 1 showed ANNs will work with ASSET. This is a successful first step towards transfer learning to real sensor data.

For Phase 2, three U-Net variants were created. The 3D U-Net using the Dice-coefficient as the loss function (ANN 2) performed the best on the control dataset. Using ANN 2, the impacts of adjusting dataset parameters were evaluated. Object SNR, number of scenarios in the dataset, background to object pixel ratio all play an important role when generating IR datasets. Increasing the number of scenarios in the dataset compensates for an increase in the background to object pixel ratio. It was also shown PR AUC is a good measure of performance when determining ANN performance on the test set.

V. Conclusion

5.1 Overview

This chapter provides a summary of the research, the results of the experiments, and potential future work.

5.2 Research Conclusions

The research presented in this thesis used a Spiral model approach and two phases were completed. Phase 1 was the initial attempt in training ANNs using ASSET to generate synthetic EO/IR data. Phase 1 explored using 3D ConvNets and LSTMs to perform object detection and prediction. This phase showed a 3D ConvNet and 3D ConvNet+LSTM were generally able to solve the small scaled problem of EO/IR unresolved object detection and prediction with MAE medians of 0.31 and 0.45 respectively. The results also show the 3D ConvNet outperforms the trivial solution of always predicting the center pixel, which had a MAE median of 9.95. Combining the 3D ConvNet and LSTM degraded the overall performance. Improving the performance of Phase 1 was not pursued as the goal of proving the viability of using ASSET to train ANNs was accomplished. In summary, Phase 1 showed ASSET is a viable source of EO/IR data, which can be used to train ANNs to perform EO/IR unresolved object detection and prediction.

The assumptions were then relaxed for Phase 2, which created a harder problem. Mainly, the previous assumption of guaranteeing an object in every image frame no longer applied. Objects were allowed to appear and disappear randomly within the scenarios. This increased the difficulty of the problem and required a new ANN solution, and U-Net was chosen as the ANN solution. In addition, Phase 2 explored the impacts EO/IR datasets had on ANN performance. The results from Phase 2

demonstrated the number of scenarios needed to train an ANN correlates with the background to object pixel ratio of the dataset. The higher the ratio, the more scenarios needed to achieve the same performance on the test set. Experiment 2 and 5 resulted in the best ANN performance with average PR AUCs of 0.900 and 0.878 respectively. Experiment 2 had a background to object pixel ratio of 14:1 and the dataset used contained 500 scenarios. Experiment 5 increased the background to object pixel ratio to 60:1 and also increased the dataset scenarios to 3,000. In summary, Phase 2 showed these four items:

- U-Net is a viable ANN architecture to detect EO/IR objects using semantic image segmentation.
- 3D U-Nets outperform 2D U-Nets. Due to the strong class imbalance between background and object pixels, 3D U-Nets should also be compiled using the Dice-coefficient as the loss function.
- ANN performance is influenced by the number of scenarios in the dataset, the object's SNR, and the overall background to object pixel ratio of the dataset.
- Calculating the PR AUC is a good quantitative measure of how well the ANN performs on the test set.

5.3 Future Work

Future work can be driven in a number of different directions. Below are the recommended paths of improving the work presented in this thesis:

1. Adjust hyperparameters to detect objects with less than 20 SNR. Currently, U-Net has been optimized to detect objects with at least 20 SNR. Real world videos will contain objects with a SNR less than 5. Another option here is to also increase the dataset size to supplement the lowered object SNR.

2. Investigate how background and bias subtraction, implemented during the pre-processing steps, impact ANN performance.

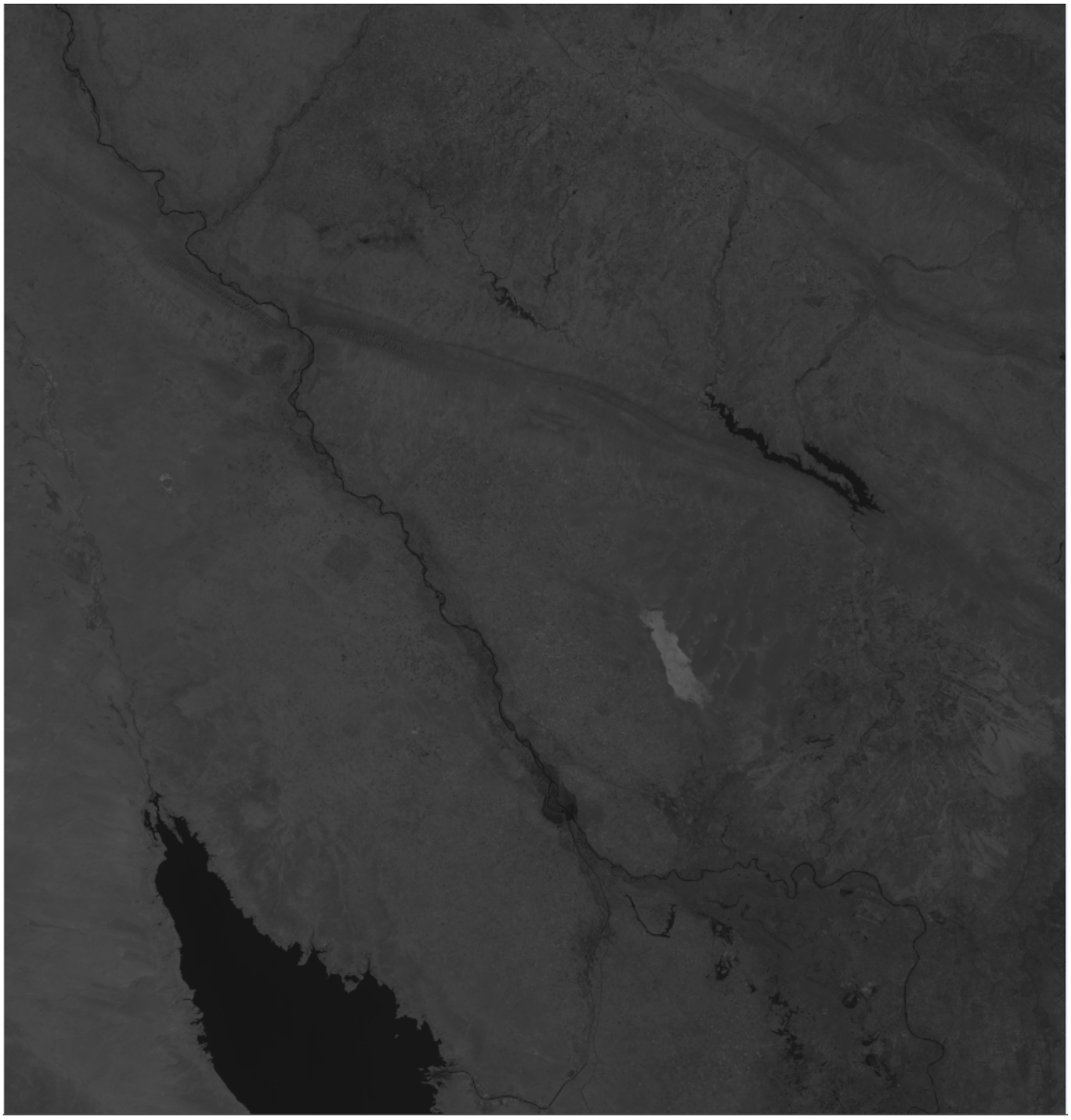
3. Create accurate object masks. The current implementation uses naive approach of adding one object pixel around the object. This is not very accurate; however, it was good enough for the goals of Phase 2.

4. Expand area of interest (AOI) to 64x64 image pixels. This will increase both the ANN training times and the background to object pixel ratios. This path allows higher resolution videos to be processed quicker. The videos can be segmented into 64x64 pixels, instead of segmenting the video into 32x32 pixels.

5. Incorporate addition classification categories. U-Net can be modified to incorporate additional classification categories. The new categories (i.e. kites, planes, and birds) can be based on object area/size, SNR, speed, and acceleration. The number of input/output channels will need to be increased to match the number of categories.

6. Compare U-Net performance to a proven traditional detection and tracking algorithm. First obtain a real world EO/IR video, and use ASSET to generate similar types of objects found in the video. Use the ASSET dataset to train U-Net. Finally, compare the object detection results of U-Net to a traditional detection and tracking algorithm.

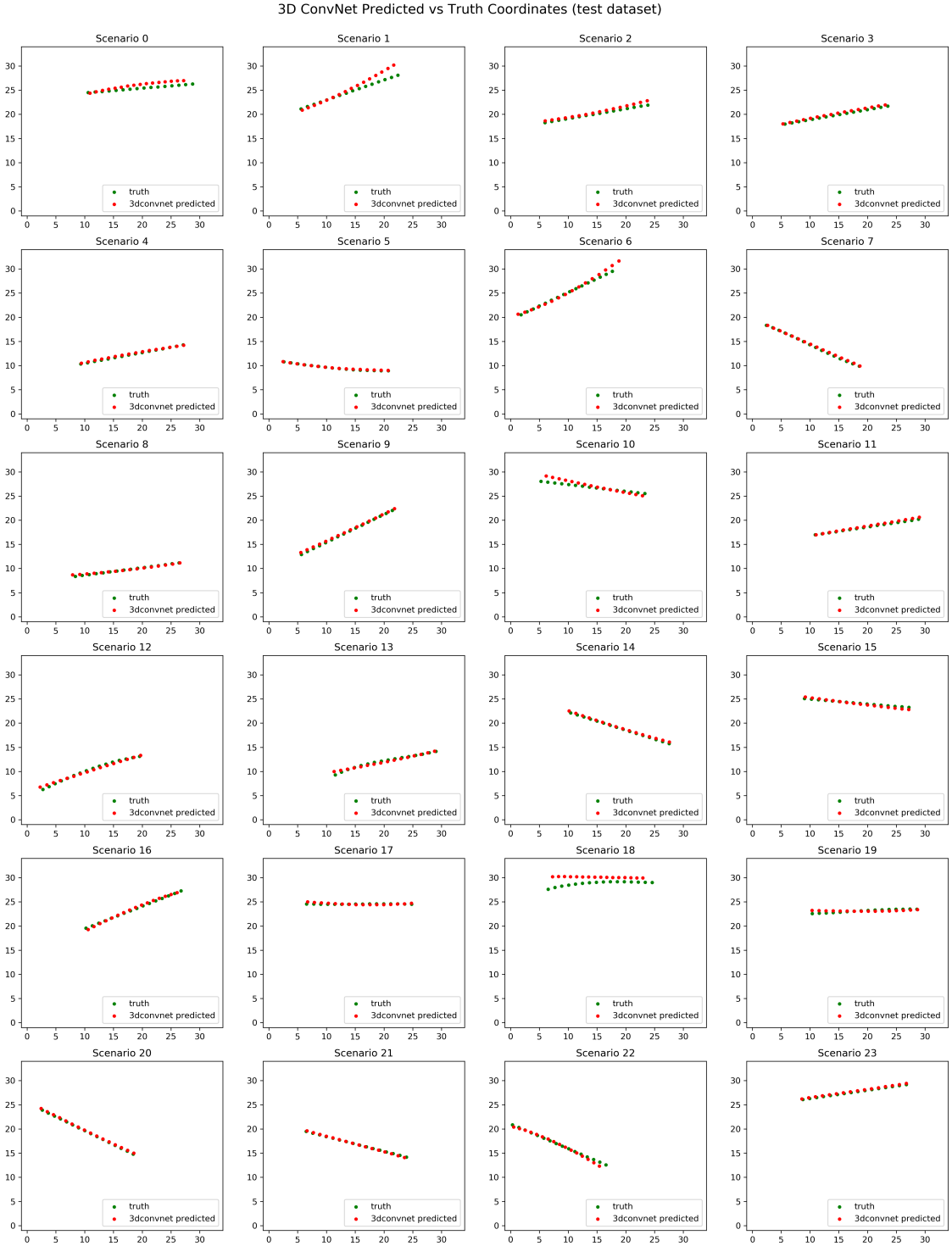
Appendix A. ASSET Default Background Image



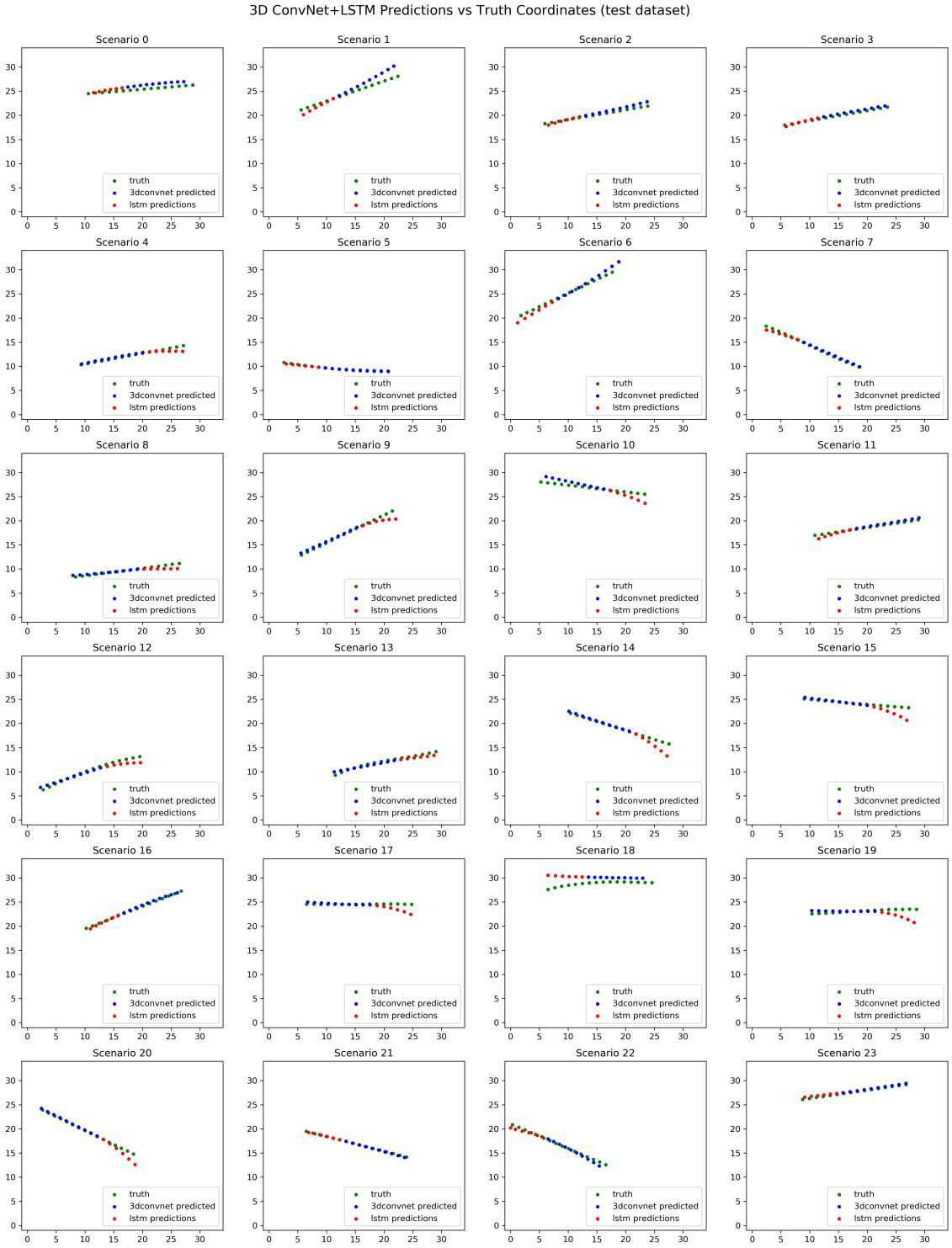
Appendix B. ASSET Object (.txt) File

Number	time	frame	row	column	signal	peak	NET	SET
	[s]		[pix]	[pix]	[cts]	[cts]	[cts]	[cts]
1	0	1	14.32	12.91	437.24	179.20	7.47	7.47
1	1	2	15.38	13.50	439.34	122.62	6.66	6.66
1	2	3	16.45	14.07	437.73	145.92	6.18	6.18
1	3	4	17.54	14.61	439.35	124.68	7.12	7.12
1	4	5	18.64	15.12	437.50	174.21	8.16	8.16
1	5	6	19.76	15.61	438.26	148.80	7.88	7.88
1	6	7	20.88	16.07	435.97	202.57	6.88	6.88
1	7	8	22.01	16.52	437.52	158.26	7.25	7.25
1	8	9	23.15	16.93	436.33	196.52	6.31	6.31
1	9	10	24.30	17.33	438.52	152.32	6.55	6.55
1	10	11	25.46	17.70	439.11	125.21	7.01	7.01
1	11	12	26.63	18.04	437.57	162.28	8.01	8.01
1	12	13	27.80	18.36	438.01	161.38	6.75	6.75
1	13	14	28.98	18.65	437.13	176.75	6.91	6.91
1	14	15	30.17	18.91	436.63	191.87	7.13	7.13
1	15	16	31.36	19.14	433.08	160.54	7.09	7.09

Appendix C. Phase 1: 3D ConvNet Predictions (First 24 Test Scenarios)



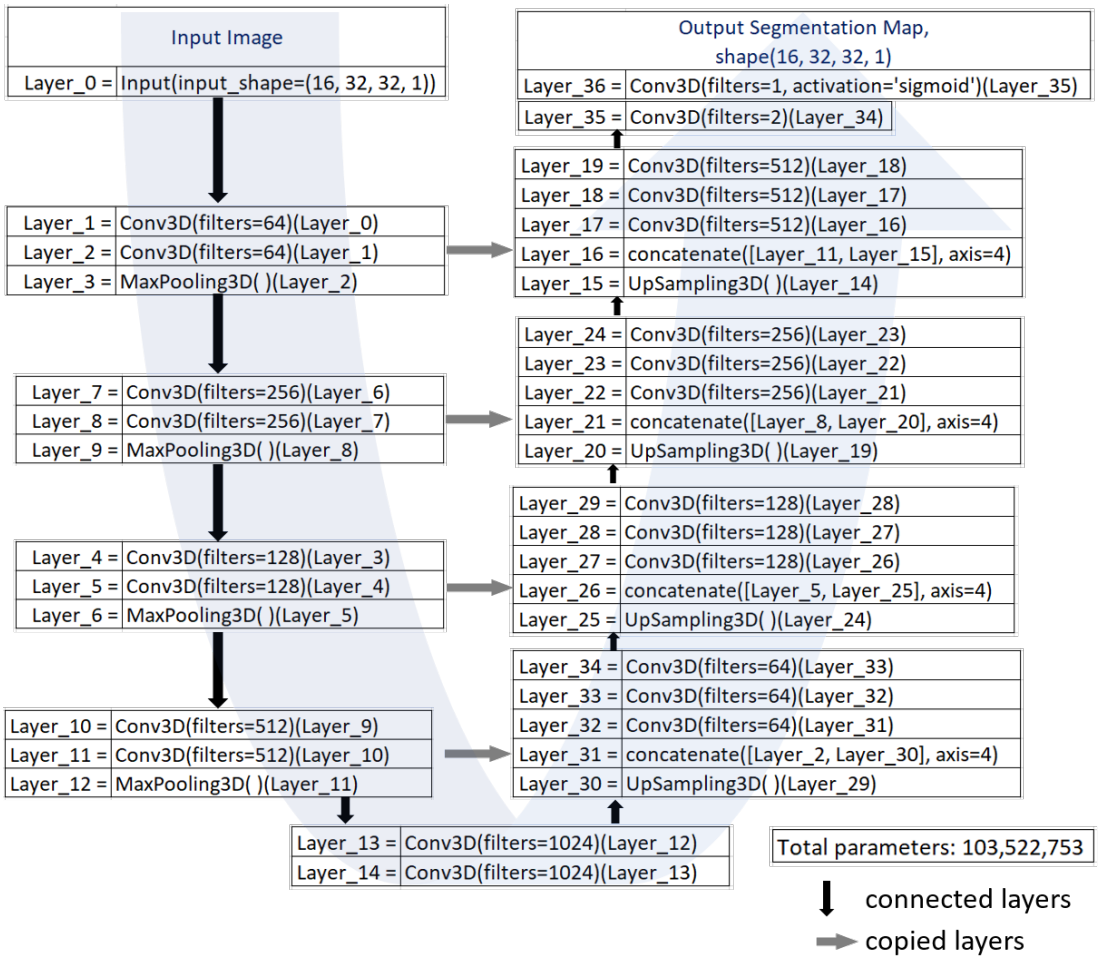
Appendix D. Phase 1: 3D ConvNet+LSTM Predictions (First 24 Test Scenarios)



Appendix E. Phase 2: Background Image Generation Parameters

Save Name	Zulu Hour	Julian Day	Initial Aimpoint	Temperature	Satellite Longitude
Planes1	11.7	58	[0 -74]	283	29
Planes2	10.8	264	[5 14]	286	-42
Planes3	13.8	146	[12 -81]	307	16
Planes4	12	73	[-87 94]	305	24
Planes5	10.2	132	[-9 -51]	314	-41
Planes6	13	355	[57 -67]	289	15
Planes7	10	339	[-20 62]	316	-29
Planes8	10.6	217	[35 56]	307	-45
Planes9	14.5	339	[-74 -35]	317	-37
Planes10	15.3	2	[73 -13]	311	-30
Planes11	10.1	148	[-72 -92]	299	5
Planes12	9.3	208	[62 -66]	315	-29
Planes13	15.1	204	[86 -55]	288	14
Planes14	13.4	360	[68 -99]	316	-11
Planes15	12.6	160	[52 -33]	302	-2
Planes16	14.3	190	[-15 82]	282	23
Planes17	10.2	168	[-75 -20]	316	2
Planes18	12.5	123	[57 -51]	308	11
Planes19	9.9	250	[-92 49]	291	42
Planes20	10.2	341	[-69 49]	298	-39
Planes21	11.8	136	[-17 80]	301	-42
Planes22	10.7	69	[-38 -2]	298	26
Planes23	10.4	212	[54 -57]	303	-4
Planes24	13.5	102	[-45 -99]	302	16
Planes25	12.7	44	[48 -78]	303	14
Planes26	10.7	6	[51 -56]	282	-42
Planes27	12.8	47	[13 20]	311	13
Planes28	15.3	309	[78 -17]	309	7
Planes29	15.2	49	[-35 -72]	316	6
Planes30	10.1	65	[91 9]	280	1
Planes31	9	39	[18 -9]	290	37
Planes32	13.8	269	[93 70]	283	-40
Planes33	11.3	79	[2 80]	310	-26
Planes34	13.4	225	[-59 14]	313	32
Planes35	14.9	66	[-76 -29]	289	-11
Planes36	12.8	139	[-35 68]	315	18
Planes37	11.7	102	[-1 58]	299	-25
Planes38	15	70	[-64 53]	317	-32
Planes39	14.8	54	[-100 25]	305	29
Planes40	15.5	346	[-22 -64]	313	-41

Appendix F. Phase 2: ANN 2 and 3 Architecture



Bibliography

1. M. M. Abdo and L. Hongzuo, "New approach in processing of the infrared image sequence for moving dim point targets detection," *Journal of Signal and Information Processing*, vol. 4, no. 3B, p. 144, 2013.

2. S. R. Young, B. J. Steward, and K. C. Gross, "Development and validation of the afit scene and sensor emulator for testing (asset)," *Proc.SPIE*, vol. 10178, 2017. [Online]. Available: https://doi.org/10.1117/12.2262193

3. S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 1, pp. 221–231, 2013.

4. D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4489–4497.

5. B. Kim, C. M. Kang, J. Kim, S. H. Lee, C. C. Chung, and J. W. Choi, "Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, Oct 2017, pp. 399–404.

6. J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *CoRR*, vol. abs/1411.4038, 2014. [Online]. Available: http://arxiv.org/abs/1411.4038

7. O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015. [Online]. Available: http://arxiv.org/abs/1505.04597

8. D. L. Chenoweth, "Infrared search and track signal processing: a potential application of artificial neural computing," in *1989 First IEEE International Conference on Artificial Neural Networks, (Conf. Publ. No. 313)*, Oct 1989, pp. 270–274.

9. R. Liou and M. R. Azimi-Sadjadi, "Dim target detection and clutter rejection using modified high order correlation neural network," in *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, vol. 4, June 1992, pp. 289–294 vol.4.

10. M. V. Shirvaikar and M. M. Trivedi, "A neural network filter to detect small targets in high clutter backgrounds," *IEEE Transactions on Neural Networks*, vol. 6, no. 1, pp. 252–257, Jan 1995.

11. L. Yun and R. Zhou, "A novel method for infrared small targets detection," in *2010 Asia Pacific Conference on Postgraduate Research in Microelectronics and Electronics (PrimeAsia)*, Sept 2010, pp. 111–114.

12. J. Hu, Y. Hu, and X. Lu, "A new method of small target detection based on neural network," *Proc.SPIE*, vol. 10608, 2018. [Online]. Available: https://doi.org/10.1117/12.2285178

13. Y.-L. Kong, Q. Huang, C. Wang, J. Chen, J. Chen, and D. He, "Long short-term memory neural networks for online disturbance detection in satellite image time series," *Remote Sensing*, vol. 10, no. 3, 2018. [Online]. Available: http://www.mdpi.com/2072-4292/10/3/452

14. Q. Dou, H. Chen, L. Yu, L. Zhao, J. Qin, D. Wang, V. C. Mok, L. Shi, and P. Heng, "Automatic detection of cerebral microbleeds from mr images via 3d convolutional neural networks," *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1182–1195, May 2016.

15. X. Huang, J. Shan, and V. Vaidya, "Lung nodule detection in ct using 3d convolutional neural networks," in *2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017)*, April 2017, pp. 379–383.

16. A. Beers, K. Chang, J. M. Brown, E. Sartor, C. P. Mammen, E. R. Gerstner, B. R. Rosen, and J. Kalpathy-Cramer, "Sequential 3d u-nets for biologically-informed brain tumor segmentation," *CoRR*, vol. abs/1709.02967, 2017. [Online]. Available: http://arxiv.org/abs/1709.02967

17. W. Xiancheng, L. Wei, M. Bingyi, J. He, Z. Jiang, W. Xu, Z. Ji, G. Hong, and S. Zhaomeng, "Retina blood vessel segmentation using a u-net based convolutional neural network," in *Procedia Computer Science: International Conference on Data Science (ICDS 2018), Beijing, China, 8-9 June 2018*, 2018.

18. M. Z. Alom, M. Hasan, C. Yakopcic, T. M. Taha, and V. K. Asari, "Recurrent residual convolutional neural network based on u-net (r2u-net) for medical image segmentation," *arXiv preprint arXiv:1802.06955*, 2018.

19. M. Zeng, J. Li, and Z. Peng, "The design of top-hat morphological filter and application to infrared target detection," *Infrared Physics & Technology*, vol. 48, no. 1, pp. 67 – 76, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1350449505000757

20. Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.

21. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

22. F. Milletari, N. Navab, and S. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," in *2016 Fourth International Conference on 3D Vision (3DV)*, Oct 2016, pp. 565–571.

23. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

24. T. Oliphant, "NumPy: A guide to NumPy," USA: Trelgol Publishing, 2006–. [Online]. Available: http://www.numpy.org/

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
21-03-2019	Master's Thesis	June 2017 - March 2019

4. TITLE AND SUBTITLE

Unresolved Object Detection Using Synthetic Data Generation And Artificial Neural Networks

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Sinn, Yong, U, Capt

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering an Management (AFIT/EN)
2950 Hobson Way
Wright-Patterson AFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENG-MS-19-M-055

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Intentionally Left Blank

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This research presents and solves constrained real-world problems of using synthetic data to train artificial neural networks (ANNs) to detect unresolved moving objects in wide field of view (WFOV) electro-optical/infrared (EO/IR) satellite motion imagery. Objectives include demonstrating the use of the Air Force Institute of Technology (AFIT) Sensor and Scene Emulation Tool (ASSET) as an effective tool for generating EO/IR motion imagery representative of real WFOV sensors and describing the ANN architectures, training, and testing results obtained. Deep learning using a 3-D convolutional neural network (3D ConvNet), long short term memory (LSTM) network, and U-Net are used to solve the problem of EO/IR unresolved object detection. U-Net is shown to be a promising ANN architecture for performing EO/IR unresolved object detection. In two of the experiments, U-Net achieved 90% and 88% pixel prediction accuracy. In addition, the results show ASSET is capable of generating sufficient information needed to train deep learning models.

15. SUBJECT TERMS

ANN, ASSET, Deep Learning, Remote Sensing, Synthetic EO/IR Data

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Kenneth M. Hopkinson, AFIT/ENG
U	U	U	UU	77	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3636 x4579; kenneth.hopkinson@afit.edu