



**Calhoun: The NPS Institutional Archive**  
**DSpace Repository**

---

Theses and Dissertations

1. Thesis and Dissertation Collection, all items

---

2022-06

# TARGET POSE ESTIMATION VIA DEEP LEARNING FOR MILITARY SYSTEMS

Heath, Raven S.

Monterey, CA; Naval Postgraduate School

---

<http://hdl.handle.net/10945/70684>

---

This publication is a work of the U.S. Government as defined in Title 17, United States Code, Section 101. Copyright protection is not available for this work in the United States.

*Downloaded from NPS Archive: Calhoun*



Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

**Dudley Knox Library / Naval Postgraduate School**  
**411 Dyer Road / 1 University Circle**  
**Monterey, California USA 93943**

<http://www.nps.edu/library>



**NAVAL  
POSTGRADUATE  
SCHOOL**

**MONTEREY, CALIFORNIA**

**THESIS**

**TARGET POSE ESTIMATION VIA DEEP LEARNING  
FOR MILITARY SYSTEMS**

by

Raven S. Heath

June 2022

Thesis Advisor:

Jae Jun Kim

Co-Advisor:

Brij N. Agrawal

**Approved for public release. Distribution is unlimited.**

THIS PAGE INTENTIONALLY LEFT BLANK

<b>REPORT DOCUMENTATION PAGE</b>			<i>Form Approved OMB No. 0704-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC, 20503.				
<b>1. AGENCY USE ONLY (Leave blank)</b>		<b>2. REPORT DATE</b> June 2022	<b>3. REPORT TYPE AND DATES COVERED</b> Master's thesis	
<b>4. TITLE AND SUBTITLE</b> TARGET POSE ESTIMATION VIA DEEP LEARNING FOR MILITARY SYSTEMS			<b>5. FUNDING NUMBERS</b>	
<b>6. AUTHOR(S)</b> Raven S. Heath				
<b>7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)</b> Naval Postgraduate School Monterey, CA 93943-5000			<b>8. PERFORMING ORGANIZATION REPORT NUMBER</b>	
<b>9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)</b> N/A			<b>10. SPONSORING / MONITORING AGENCY REPORT NUMBER</b>	
<b>11. SUPPLEMENTARY NOTES</b> The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
<b>12a. DISTRIBUTION / AVAILABILITY STATEMENT</b> Approved for public release. Distribution is unlimited.			<b>12b. DISTRIBUTION CODE</b> A	
<b>13. ABSTRACT (maximum 200 words)</b>  Target pose estimation and aimpoint selection is crucial in direct energy weapon systems, as it allows the system to point to a specific and strategic area of the target. However, it is a challenging task because a dedicated attitude sensor is required. Motivated by new emerging deep learning capabilities, the present work proposes a deep learning model to estimate a target spacecraft attitude in terms of Euler angles. Data for the deep learning model were experimentally generated from 3D UAV models, incorporating effects such as atmospheric backgrounds and turbulence. The targets pose was derived from the training, validation, and prediction of 2D keypoints. With a keypoint detection model it is possible to detect interest points in an image, which allows us to estimate pose, angles, and dimensions of the target in question. Utilizing a weak-perspective direct linear transformation algorithm, the pose of a 3D object with respect to a camera from 3D to 2D correspondences could be determined. Additionally, from this correspondence, an aimpoint, mimicking laser tracking could be determined on the target. This work evaluates these methods and their accuracy against experimentally generated data with simulated real-world environments.				
<b>14. SUBJECT TERMS</b> deep learning, pose estimation, machine learning, artificial intelligence, aimpoint selection, high energy laser			<b>15. NUMBER OF PAGES</b> 87	
			<b>16. PRICE CODE</b>	
<b>17. SECURITY CLASSIFICATION OF REPORT</b> Unclassified	<b>18. SECURITY CLASSIFICATION OF THIS PAGE</b> Unclassified	<b>19. SECURITY CLASSIFICATION OF ABSTRACT</b> Unclassified	<b>20. LIMITATION OF ABSTRACT</b> UU	

THIS PAGE INTENTIONALLY LEFT BLANK

**Approved for public release. Distribution is unlimited.**

**TARGET POSE ESTIMATION VIA DEEP LEARNING  
FOR MILITARY SYSTEMS**

Raven S. Heath  
Ensign, United States Navy  
BS, United States Naval Academy, 2021

Submitted in partial fulfillment of the  
requirements for the degree of

**MASTER OF SCIENCE IN ASTRONAUTICAL ENGINEERING**

from the

**NAVAL POSTGRADUATE SCHOOL  
June 2022**

Approved by: Jae Jun Kim  
Advisor

Brij N. Agrawal  
Co-Advisor

Garth V. Hobson  
Chair, Department of Mechanical and Aerospace Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

## ABSTRACT

Target pose estimation and aimpoint selection is crucial in direct energy weapon systems, as it allows the system to point to a specific and strategic area of the target. However, it is a challenging task because a dedicated attitude sensor is required. Motivated by new emerging deep learning capabilities, the present work proposes a deep learning model to estimate a target spacecraft attitude in terms of Euler angles. Data for the deep learning model were experimentally generated from 3D UAV models, incorporating effects such as atmospheric backgrounds and turbulence. The targets pose was derived from the training, validation, and prediction of 2D keypoints. With a keypoint detection model it is possible to detect interest points in an image, which allows us to estimate pose, angles, and dimensions of the target in question. Utilizing a weak-perspective direct linear transformation algorithm, the pose of a 3D object with respect to a camera from 3D to 2D correspondences could be determined. Additionally, from this correspondence, an aimpoint, mimicking laser tracking could be determined on the target. This work evaluates these methods and their accuracy against experimentally generated data with simulated real-world environments.



THIS PAGE INTENTIONALLY LEFT BLANK

# TABLE OF CONTENTS

<b>I.</b>	<b>INTRODUCTION.....</b>	<b>1</b>
<b>A.</b>	<b>OVERVIEW.....</b>	<b>1</b>
<b>B.</b>	<b>OBJECTIVES.....</b>	<b>2</b>
<b>C.</b>	<b>APPROACH.....</b>	<b>2</b>
<b>D.</b>	<b>EXPERIMENT INTRODUCTION.....</b>	<b>3</b>
<b>E.</b>	<b>SCOPE.....</b>	<b>4</b>
<b>F.</b>	<b>THESIS ORGANIZATION.....</b>	<b>4</b>
<b>II.</b>	<b>LITERATURE REVIEW.....</b>	<b>5</b>
<b>A.</b>	<b>ARTIFICIAL INTELLIGENCE.....</b>	<b>5</b>
1.	Machine Learning.....	5
2.	Deep Learning.....	7
3.	CNNs.....	8
<b>B.</b>	<b>DETECTION ALGORITHMS.....</b>	<b>9</b>
1.	Mask R-CNN.....	9
2.	Keypoint R-CNN.....	10
<b>C.</b>	<b>MILITARY APPLICATIONS.....</b>	<b>13</b>
1.	Pose Estimation.....	13
2.	Aimpoint Tracking.....	14
<b>III.</b>	<b>PROBLEM FORMULATION AND METHODOLOGY.....</b>	<b>17</b>
<b>A.</b>	<b>SOFTWARE.....</b>	<b>17</b>
1.	PyTorch.....	17
2.	Jupyter Notebook.....	17
3.	MATLAB.....	18
<b>B.</b>	<b>DATASETS.....</b>	<b>19</b>
1.	No Atmospheric Effects Dataset.....	20
2.	Atmospheric Effects Dataset.....	21
<b>C.</b>	<b>PHASE 1: KEYPOINT DETECTION.....</b>	<b>23</b>
1.	Data Augmentation.....	23
2.	Algorithm Implementation.....	25
3.	Training.....	25
4.	Predictions and Evaluations.....	28
<b>D.</b>	<b>PHASE 2: POSE ESTIMATION.....</b>	<b>30</b>
1.	Data Augmentation.....	30
2.	Keypoint Implementation.....	31
3.	Predictions and Evaluations.....	32

E.	<b>PHASE 3: AIMPOINT SELECTION AND TRACKING .....</b>	<b>32</b>
1.	<b>Inference .....</b>	<b>33</b>
2.	<b>Evaluations and Predictions.....</b>	<b>33</b>
IV.	<b>SIMULATION RESULTS AND DL PERFORMANCE EVALUATION .....</b>	<b>35</b>
A.	<b>PHASE 1: KEYPOINT DETECTION.....</b>	<b>35</b>
1.	<b>Bounding Boxes.....</b>	<b>35</b>
2.	<b>Keypoints .....</b>	<b>38</b>
B.	<b>PHASE 2: POSE ESTIMATION .....</b>	<b>46</b>
1.	<b>Dataset 1.....</b>	<b>47</b>
2.	<b>Dataset 2.....</b>	<b>51</b>
C.	<b>PHASE 3 AIMPOINT SELECTION .....</b>	<b>55</b>
1.	<b>Dataset 1.....</b>	<b>55</b>
2.	<b>Dataset 2.....</b>	<b>58</b>
V.	<b>CONCLUSION .....</b>	<b>61</b>
A.	<b>SUMMARY .....</b>	<b>61</b>
B.	<b>FUTURE RESEARCH.....</b>	<b>61</b>
	<b>LIST OF REFERENCES.....</b>	<b>63</b>
	<b>INITIAL DISTRIBUTION LIST .....</b>	<b>67</b>

## LIST OF FIGURES

Figure 1.	Machine Learning Techniques. Source: [1].	6
Figure 2.	Neural Network Layered Node Structure. Source: [4].	6
Figure 3.	Neural Network Node Function. Source: [4].	7
Figure 4.	CNN Architecture Example. Source: [4].	8
Figure 5.	Deep Learning Algorithms Timeline. Source: [6].	9
Figure 6.	Mask R-CNN Output. Source: [8].	10
Figure 7.	Mask Encoding Examples (Left: Mask R-CNN. Right: Keypoint R-CNN.). Source: [9].	10
Figure 8.	Keypoint R-CNN Architecture. Source: [9].	11
Figure 9.	OKS Example. Source: [9].	12
Figure 10.	Weak-Perspective Direct Linear Transformation Diagram. Source: [17].	14
Figure 11.	Jupyter Notebook Interface.	18
Figure 12.	MATLAB Interface.	19
Figure 13.	NOAT Dataset Images.	20
Figure 14.	NOAT Dataset Annotation File.	21
Figure 15.	AT Dataset Images.	22
Figure 16.	AT Dataset Annotation File.	22
Figure 17.	Keypoint Model Code.	25
Figure 18.	Hyperparameters Code.	26
Figure 19.	Heatmap Output.	29
Figure 20.	3D Target Points Annotation File.	30
Figure 21.	WPDLT MATLAB Function. Source: [12].	31
Figure 22.	NOAT Model Bounding Box Predictions.	36

Figure 23.	Optimal Bounding Box Predictions from AT Model. ....	36
Figure 24.	Avery Bounding Box Predictions from AT Model.....	37
Figure 25.	Examples of No Predictions from AT Model. ....	37
Figure 26.	NOAT Model Default Keypoint Predictions. ....	39
Figure 27.	NOAT Default Keypoint Confusion Matrix. ....	40
Figure 28.	Optimal Heatmap Keypoint Predictions from NOAT Model.....	41
Figure 29.	Poor Heatmap Keypoints Predictions from NOAT Model.....	41
Figure 30.	NOAT Heatmap Keypoint Confusion Matrix.....	42
Figure 31.	Optimal Default Keypoint Predictions from AT Model. ....	43
Figure 32.	Poor Default Keypoint Predictions from AT Model.....	43
Figure 33.	AT Default Keypoint Confusion Matrix.....	44
Figure 34.	AT Model Heatmap Keypoint Predictions. ....	45
Figure 35.	AT Heatmap Keypoint Confusion Matrix. ....	46
Figure 36.	NOAT Pose Estimation from Default Keypoints. ....	47
Figure 37.	Poor Pose Estimation of Default Keypoints. ....	48
Figure 38.	Optimal Pose Estimation of Heatmap Keypoints. ....	49
Figure 39.	Poor Pose Estimation of Heatmap Keypoints.....	49
Figure 40.	NOAT Pose RMSE Results. (Left: Default Keypoint. Right: Heatmap Keypoints). ....	50
Figure 41.	AT Pose Estimation from Default Keypoints. ....	52
Figure 42.	AT Pose Estimation from Heatmap Keypoints.....	53
Figure 43.	AT Pose RMSE Results. ....	54
Figure 44.	NOAT Aimpoint Selection from Default Keypoints. ....	55
Figure 45.	NOAT Aimpoint Selection from Heatmap Keypoints.....	56
Figure 46.	NOAT Aimpoint RMSE Results. ....	57

Figure 47.	AT Aimpoint Selection from Default Keypoints.....	58
Figure 48.	AT Aimpoint Selection from Heatmap Keypoints. ....	59
Figure 49.	AT Aimpoint RMSE Results. ....	60

THIS PAGE INTENTIONALLY LEFT BLANK

## LIST OF TABLES

Table 1.	Hyperparameters of Models.....	27
Table 2.	Validation Performance of Finals Models .....	27
Table 3.	3D Target Aimpoints. ....	33
Table 4.	Summary of Bounding Box Prediction Results.....	35
Table 5.	Mean RMSE of NOAT Pose Results.....	50
Table 6.	AT Pose Mean RMSE.....	54
Table 7.	NOAT Aimpoint Mean RMSE .....	57
Table 8.	AT Aimpoint Mean RMSE.....	60



THIS PAGE INTENTIONALLY LEFT BLANK

## LIST OF ACRONYMS AND ABBREVIATIONS

AI	artificial intelligence
AP	average precision
AT	atmospheric effects
CNN	Convolution Neural Networks
DL	deep learning
ISR	intelligence, surveillance, and reconnaissance
JSON	JavaScript Object Notation
LM	Lockheed Martin
LWS	laser weapons system
MATLAB	Matrix Laboratory
ML	machine learning
NN	Neural Networks
NOAT	no atmospheric effects
OKS	Object Keypoint Similarity
PnP	Perspective-n-Point
R-CNN	Region based Convolutional Neural Network
RMSE	root mean squared error
ROI	region of interest
SGD	stochastic gradient descent
WPDLT	weak-perspective direct linear transfer

THIS PAGE INTENTIONALLY LEFT BLANK

## **ACKNOWLEDGMENTS**

I would like to thank my thesis advisors, Dr. Jae Kim and Dr. Brij Agrawal, for their assistance and guidance. I would also like to thank Lockheed Martin, especially Mr. Gary Chern, for providing their data and knowledge on the topic of machine learning.

THIS PAGE INTENTIONALLY LEFT BLANK

# I. INTRODUCTION

## A. OVERVIEW

Unmanned aerial vehicles (UAVs) have become a serious security threat to the military within the last decade. Counter offensive and defensive operations towards drones are paramount as the dynamic of warfare changes. Developments in drone detection and tracking are becoming a hallmark operation in the fleet for intelligence, surveillance, and reconnaissance (ISR). One of the most recent developments is the Navy's high energy laser weapons system (LWS). The LWS has the potential to advance ship and battle aerial defense as it supports in the selection of targets for engagement. The challenges from the LWS are a consequence of a series of decisive factors. These factors include the target's characteristics (range, position, velocity) and atmospheric conditions and their effect on the laser beam. Decision making for the LWS is dependent on these factors and can affect battle readiness.

The use of artificial intelligence (AI) could support warfighters to address the challenges in weapon guidance systems. Researchers have been developing deep learning (DL) algorithms within the last two decades, that have been able to imitate the functions of the human cortex and brain. However, the use of these types of autonomous systems within a military domain has not been thoroughly evaluated. There are several challenges that come with AI implementation. The development of autonomous cars exhibit the limitations of cognitive decision making for AI. The primary cause of these limitations is the high volume of data exposure that DL algorithms require to have a performance comparable to human functions. However, AI implementation has numerous counts of success that are now a part of daily life. AI and DL could be beneficial in advancing weapon guidance systems such as LWS in terms of autonomy and accuracy.

The specific applications of the LWS that AI implementation could transform is aimpoint selection and tracking. The LWS system's decision making involves detecting, tracking, and engaging an aerial target. This process is analogous to current DL detection and pose estimation algorithms. Although the majority of these algorithms were developed

for specific problems such as human pose or satellite imagery, they can be reconstructed to fit this problem domain. The goal of this research is to conduct experimentation in exploring the application of DL algorithms on weapon guidance systems against aerial targets. This goal was accomplished by evaluating two datasets that are comprised of synthetic imagery of the Mongoose UAV. These datasets factor in real world limitations such as atmospheric affects, target turbulence and detection ambiguity due to pose. This research focused on the use of a keypoint detection DL algorithm and weak-perspective direct linear transformation (WPDLT) pose estimation algorithm to determine aimpoint selection and tracking. This research studied various DL algorithms to obtain the most appropriate fit for the problem. Finally, this study evaluated the performance of the AI implementation for the purpose of identifying its potential effectiveness for systems like the LWS.

## **B. OBJECTIVES**

The primary objective of this research was to evaluate if AI and DL methods could improve the target detection, pose estimation and aimpoint tracking of naval weapon guidance systems. Subsidiary research goals were:

- To study DL methods to identify and evaluate algorithms suitable for improving Naval directed energy weapon system guidance
- To study how DL algorithms developed for human pose can be transferred into tracking systems of aerial targets
- To demonstrate the application of DL to this problem domain through algorithm training and modeling.
- To identify and evaluate the limitations of the DL through synthetic data

## **C. APPROACH**

This thesis approached the research objective through a literature review, development of an open-source DL algorithms to fit the problem domain, and experimentation through training and evaluation of algorithm models. The research began

with a literature review to accumulate information on artificial intelligence (including machine learning (ML), DL and Convolutional Neural Networks (CNNs)), object detection algorithms and DL applications for Naval weapon systems. This information was the foundation for augmenting and evaluating a DL algorithm within the scope of the problem.

This research used synthetic data developed and provided by Lockheed Martin. This data simulated current limitations faced in pose estimation and aimpoint tracking applications such as image noise or clutter, and target ambiguity. These datasets were used to train a DL algorithm to perform keypoint detection, and inference pose estimation and aimpoint tracking. The DL algorithm modeling process was conducted following the five phases: data augmentation, adaptation, training, improving, and validating. The research approach followed five primary steps:

1. Identify DL methodology and techniques
2. Augment datasets to fit DL algorithm format
3. Train and evaluate a keypoint detection model on dataset
4. Determine and evaluate 3D pose from 2D predicted keypoints
5. Determine and evaluate aimpoint accuracy from predicted 3D pose

#### **D. EXPERIMENT INTRODUCTION**

The experimentation for this thesis followed three major phases based on the DL methodology discussed in Chapter III.

- Phase 1 (Keypoint Detection): Developed an open-source DL algorithm for keypoint detection to fit provided datasets. Algorithm was trained to predict keypoints of aerial targets. Prediction accuracy was evaluated and compared to ground-truth keypoints of data.
- Phase 2 (Pose Estimation): Implemented a pose estimation algorithm to determine the 3D-2D pose correspondences. Pose accuracy was evaluated and compared to ground-truth pose of data.



- Phase 3 (Aimpoint Tracking): Utilized pose estimation outcomes to determine aimpoint tracking capability and accuracy.

## **E. SCOPE**

This research focused on the use of DL algorithms to improve acquisition, tracking, and pointing performance of Naval weapon systems that requires target detection and aimpoint determination. The scope of this thesis was intended to support the investigation of how AI methods can be applied to encourage autonomy and accuracy for the directed energy weapon system guidance. Autonomic target detection and pose estimation provide challenges that researchers in AI and DL have been evaluating for the last two decades. Implementation of AI into technology only advances capabilities which is mutually beneficial for military applications as the dynamic of warfare progresses. The training and evaluation of DL algorithms for target detection and pose estimation support this research's demonstration of DL capability in military applications. The scope of this study covered a narrow methodology specifically for UAV targets, which is discussed in detail in Chapter III.

## **F. THESIS ORGANIZATION**

This thesis is organized into five chapters. Chapter I provides the overview of this research, its objectives, and the general approach for experimentation. Chapter II addresses the background and theory of this research in a literature review. Chapter III details the methodology that was implemented to carry out the experimentation. Chapter IV presents the results of the research as well as a survey of the limitations found. Chapter V recounts the conclusion and findings of the research and discusses potential future work.

## II. LITERATURE REVIEW

This literature review presents the background information required for the foundation of this thesis research. There are three main topics that are pertinent to the study: (1) an overview of artificial intelligence, including ML, DL and CNNs, (2) a review of detection algorithms, and (3) a review of military applications for this research, including pose estimation and aimpoint tracking.

### A. ARTIFICIAL INTELLIGENCE

#### 1. Machine Learning

ML is a subset of AI. The concept of ML was birthed in the 1950s, when psychologist Frank Rosenblatt cultivated a team to build a machine capable of recognizing the letters of the English alphabet [1]. His inspiration for this machine was the human nervous system. Within the next decade, Rosenblatt's study transformed into computational learning and pattern recognition based on probability and statistics [1]. Since then, ML has only progressed and is now one of the most heavily researched areas within computer science.

Present day ML algorithms are used to make predictions on data [2]. The significance of ML can be seen in its countless applications. ML has been used by the medical community for patient data analysis, predicting probabilities of diseases [2]. It can be utilized to survey and predict poverty in countries from satellite imagery [3]. ML has no bounds to its applications and is beneficial to all areas of interest as technology advances. The advantages from ML come from its several techniques which are illustrated in Figure 1.

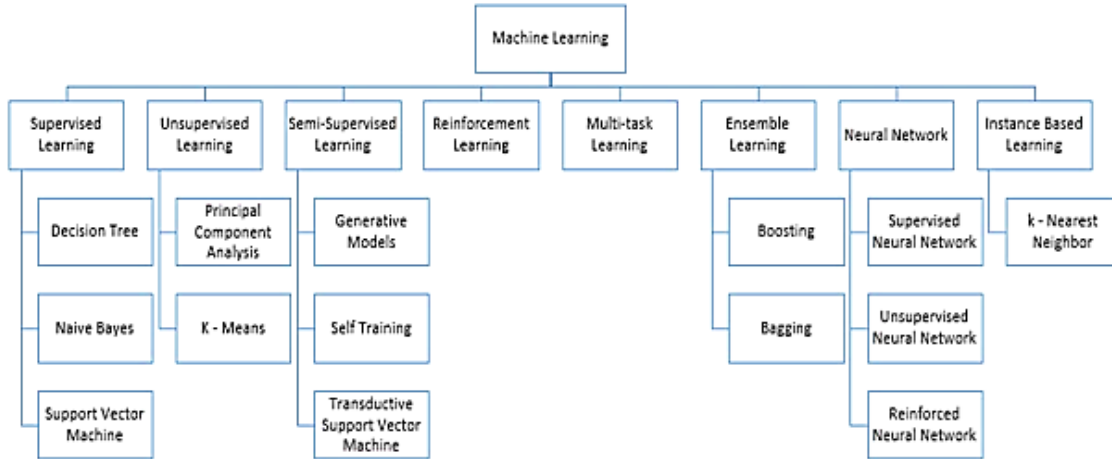


Figure 1. Machine Learning Techniques. Source: [1].

ML is the art of designing algorithms that give a computer the ability to learn. The process for this is dependent on the input data and desired output of the specific problem [2]. The techniques of ML that are utilized in this thesis research are neural networks (NNs). NNs mimic the mechanisms of the brain [4]. They are comprised of node connections analogous to the brain’s web of neurons [4]. NNs can be created in a variety of ways depending on how the nodes are connected or layered. An example of a layered structure of nodes is shown in Figure 2.

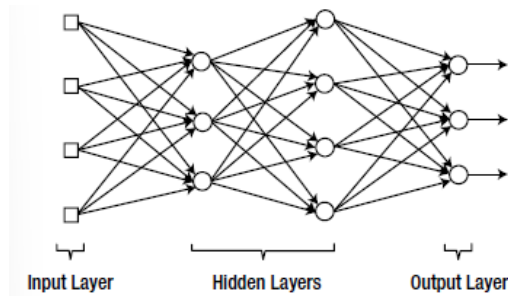


Figure 2. Neural Network Layered Node Structure. Source: [4].

The square nodes on the left Figure 2 represent the input and connect to the first layer of nodes [4]. The arrows represent signal flow, and the outer layer represents the result of the NN [4]. The layers in between these two are defined as hidden layers as they

are inaccessible outside of the network [4]. The node within each layer carries out the mechanism of collecting the information the NN learns from. The NN stores this information in the form of weights and bias [4]. To understand the function of a node in a neural network, Figure 3 illustrates an example of its relationships with weights and biases.

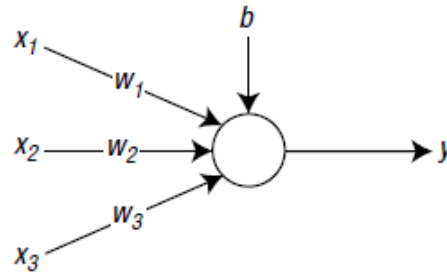


Figure 3. Neural Network Node Function. Source: [4].

From the diagram above, the  $x$  variables are the input signals, the  $w$  variables are the weights, the  $b$  variable is the bias, and the  $y$  variable is the output [4]. The node, circle in figure, reads in the input signals multiplied by the weights [4]. They are then summed within the node with the bias. The weighted sum is then sent to the output. The output equation can be seen below [4]:

$$y = \varphi(v)$$

where  $y$  is the output,  $\varphi$  is the activation function and  $v$  is the weighted sum. The activation function depicts the node's behavior [4]. There are a variety of activation functions for NNs depending on the problem it is solving. The overall result is a trained model on the input, based on weights minimizing the error between in its output and the correct output.

## 2. Deep Learning

DL is a ML technique that utilizes multi-layered NNs of two or more layers [4]. DL became a commonly used technique in the early 2000s as it solved the primary limitation seen from single-layered NNs, the lack of learning from incomplete training [4]. DL developments showed that by strategically layering NNs to increase the complexity of an algorithm, the performance could be improved during training [4]. These strategies

expanded the capabilities of NNs when it came to imitating human brain functions [5]. The capabilities that will be implemented in this study are in computer vision.

### 3. CNNS

Convolutional Neural Networks (CNNs) are a type of deep learning neural network (DLNN) that is specialized for image recognition [4]. CNNs are unique in the fact that they mimic the relationship between the visual cortex and the human brain for recognizing objects [4]. Image recognition for CNNs includes a set of tasks such as image classification, object localization and object detection. These tasks are seen every day from facial recognition in iPhones, to autonomous vehicles and self-service machines [6]. CNNs work by processing images to compare its features. Figure 4 provides an example of a CNN architecture.

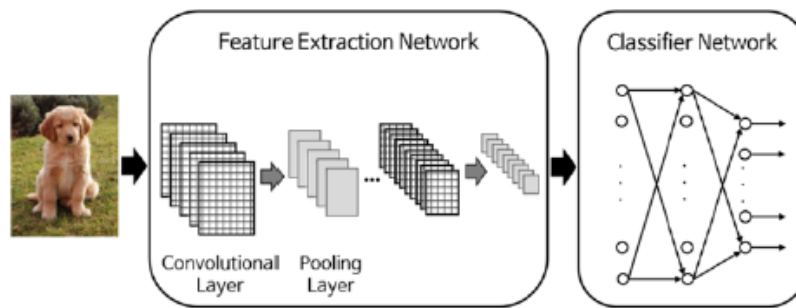


Figure 4. CNN Architecture Example. Source: [4].

The feature extractor in the example CNN is a special type of NNs, comprised of multiple convolutional or pooling layers [4]. The convolutional layer is a collection of filters whereas the pooling layer combines neighboring pixels into one [4]. The deeper the feature extraction network, the higher yield of accuracy for the CNN. The classification network in the example CNN determines the desired output for the CNN which is the task of classification. This is analogous to the eye and brain relationship of seeing an object and determining what kind of object it is. From the diagram, the object in question is the classification of a dog. The tasking of CNNs that is the focus of this thesis is object detection. Object detection is based on image classification and localization, which is

locating a known object within the given image. The algorithm must identify the class of the object as well as locate it by marking it with a bounding box [6] Object detection is an application that has been heavily improved in the last decade with a variety of available algorithms to use. The development of DL detection algorithms can be seen in Figure 5.

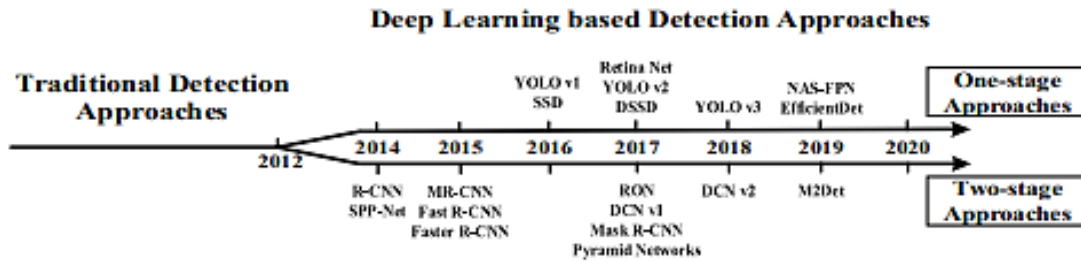


Figure 5. Deep Learning Algorithms Timeline. Source: [6].

From years of object detection competition, research teams have been able to build and expand off one another to create a family of open-source DL algorithms. This study determined the best algorithm for Phase 1 of this research was an extension of the Mask Region-based convolutional neural network (R-CNN): Keypoint R-CNN.

## B. DETECTION ALGORITHMS

### 1. Mask R-CNN

Mask R-CNN is a variant of R-CNN, a special type of object detection model. R-CNN became a notable algorithm due to its feature extractor method of region proposals [7]. The method R-CNN use involves extracting a set number of proposed object regions within the image and evaluating the CNN independently for each region of interest (ROI) [8]. Mask R-CNN differs from the prototype model by having three outputs instead of two [8]. Formerly, object detection algorithms have two methods: one-stage or two-stage [6]. R-CNN is a two-stage method by having an output of a class label and a bounding box for each object [7]. Mask R-CNN introduced a third branch that outputted the object's mask. A mask for an object is its spatial layout [8]. Figure 6 shows examples of the mask output.

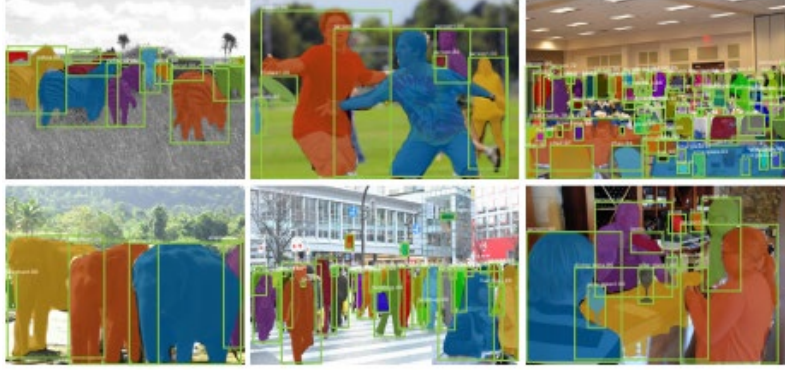


Figure 6. Mask R-CNN Output. Source: [8].

The colored overlays of the people within the example images are the mask encoded from Mask R-CNN. This algorithm can have this additional output by having a ROI-Align layer versus a ROI-Pooling layer which is seen in R-CNN. Additionally, Mask R-CNN has a Mask head that outputs the mask corresponding to a specific class [8]. This new layer was created due the need for higher accuracy in image segmentation [8].

## 2. Keypoint R-CNN

The specific algorithm this thesis uses for Phase 1 of its experimentation is Keypoint R-CNN, which is an adaptation of Mask R-CNN. Keypoint R-CNN differs by one-hot encoding a keypoint instead of an object mask [8]. An example of this encoding is seen below in Figure 7.



Figure 7. Mask Encoding Examples (Left: Mask R-CNN. Right: Keypoint R-CNN.). Source: [9].

On the left side of Figure 7 is an example of how Mask R-CNN encodes its mask for training. There is a mask for the object class and for the background class, and the result is a channel for the two respectively [9]. This encoding method is the approach used in Keypoint R-CNN. However, it focuses on a specific point in the object rather than it entirely as a mask. On the right side of the diagram is the example of Keypoint R-CNN mask encoding. Each mask has its own keypoint class [9] resulting in a channel for each specific keypoint. Keypoint R-CNN is also originally trained for identifying keypoints in a person, so it is limited to only one object class for detection [8]. The architecture of Keypoint R-CNN is provided in Figure 8.

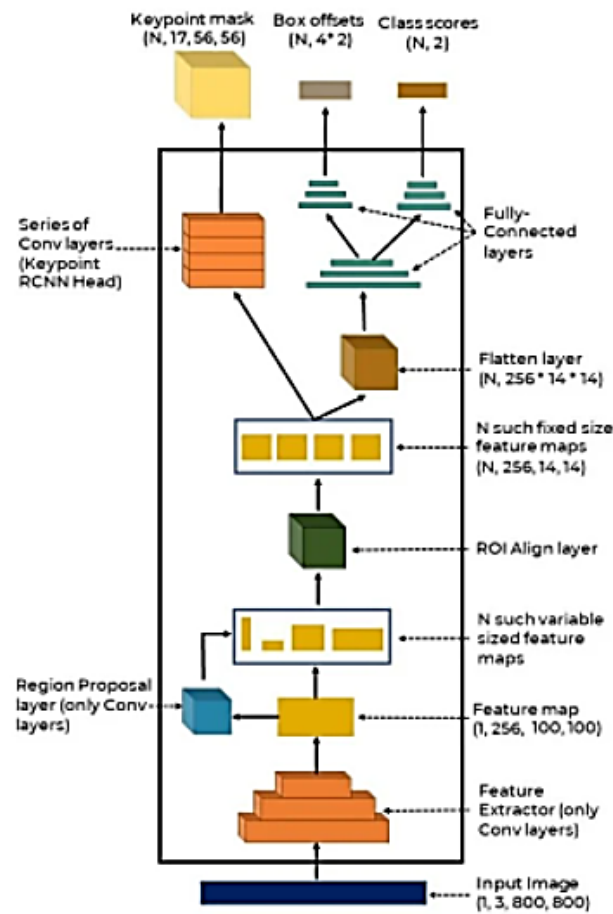


Figure 8. Keypoint R-CNN Architecture. Source: [9].



Keypoint R-CNN was originally developed to determine 17 keypoints for human pose. Therefore, this algorithm does not support multi-class object detection since it was developed for specifically the human class. The metric of Keypoint R-CNN is called Object Keypoint Similarity (OKS) [10]. This metric quantifies the error of the predicted keypoint location against its ground-truth keypoint location. OKS ranges from 0 to 1 [9]. The closer the prediction is to the ground-truth, the closer the OKS will be to 1. OKS is a factor of the object's scale. Lower penalization should be for larger objects and hence its OKS will be better [9]. A visual example of how OKS works is illustrated in Figure 9.

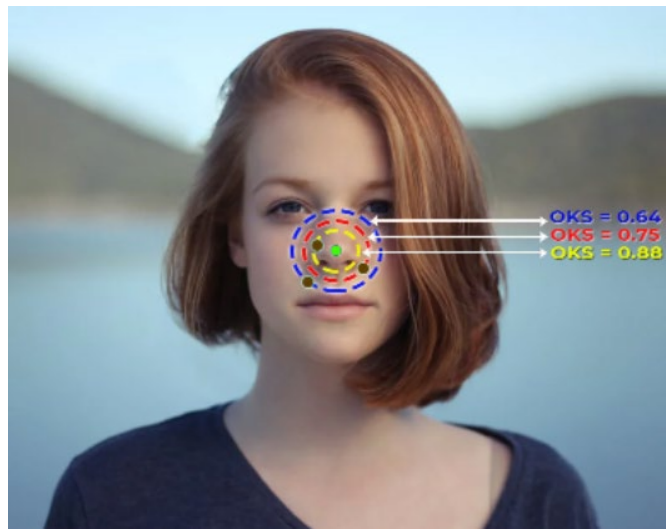


Figure 9. OKS Example. Source: [9].

From the figure above, the green dot is the truth keypoint, the brown dots are predicted keypoints; the dashed circles depict different strength values of OKS for prediction. As the OKS value increases to 1, the dashed circles shrink. This illustrates how Keypoint R-CNN measures prediction accuracy. The benefit of this algorithm is that as an object detection model it additionally provides keypoint detection and prediction. Keypoints can be utilized in applications of weapon guidance systems such as pose estimation and aimpoint tracking. The advantages and limitations of implementing keypoints in these applications will be evaluated in this study.

## C. MILITARY APPLICATIONS

The capability of AI, specifically DL is being applied to improve and advance technologies for all systems today. As the dynamic of military warfare progresses into a technologically advanced environment, implementation of DL systems could be beneficial for military readiness. Naval systems in development such as the LWS are analogous to the functions of DL algorithms. This section provides an overview of the applications of DL algorithms that can be applied to military weapon systems to improve performance and encourage autonomy.

### 1. Pose Estimation

Pose estimation is a popular application researched in DL for computer vision [11]. Pose estimation is the process of detecting and tracking 2D semantic keypoints to determine 3D pose of an object in imaging data [12]. This concept has been heavily researched for human pose estimation to understand human behavior and improve the limitations of autonomous driving [11]. However, this can be applied to any object or system. The problem of pose estimation is commonly known as the Perspective-n-Point problem (PnP) [12]. The PnP problem originates from calibrated camera pose; the goal is to estimate the 6 degrees of freedom (DOF) pose of the camera with respect to a world frame from 3D pose of an object and its 2D image corresponding points [13]. Developments of solutions for PnP are current [13, 14, 15], and widely reviewed. The solution approach this research is adapting on is WPDLT. WPDLT is based on 3D projection theory, specifically orthographic projection. Orthographic projection is the mathematical formulation for projection 3D objects into a 2D space. The difference is that WPDLT requires a scaling factor to ensure object appear the correct size in the projection [16]. For example, a closer object would appear larger in the projection. WPDLT is advantageous for cases where the scale of the object with respect to line of sight is small [13]. This data of study utilizes 3D models of UAVs therefore WPDLT is most efficient solution method to adopt. Figure 10 shows a visual depiction of WPDLT.

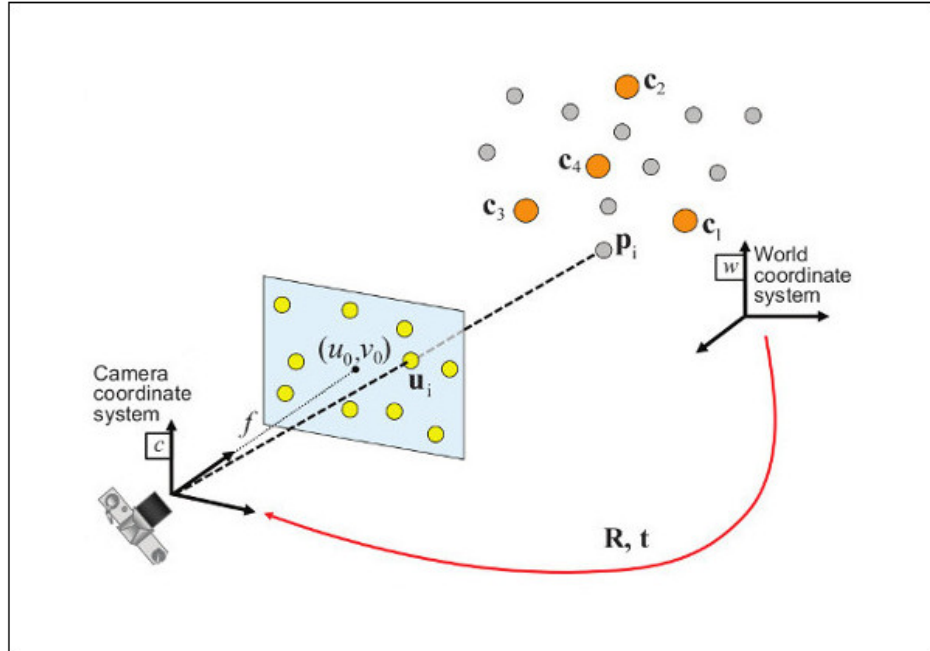


Figure 10. Weak-Perspective Direct Linear Transformation Diagram.  
Source: [17].

The 3D pose angles expressed in the world frame are casted in the image plane using a projection matrix and the cameras parameters [13]. The projection matrix is determined from known 3D pose of the object, UAV model, and its 2D image correspondences [13]. These 2D points would be the predicted keypoints from the detection algorithm mentioned above. The limitations in this method are due to scene clutter or texture ambiguity of the object in the image. These challenges make it more difficult for the target to be separated from its background which lowers accuracy [12]. However, the implementation and evaluation of DL algorithms in this military application could benefit tracking systems by receiving real-time pose data of aerial targets.

## 2. Aimpoint Tracking

Selection of a desired point on an aerial target to ensure tracking is essential to weapon guidance accuracy [18]. This application is commonly recognized in missile guidance. Aimpoint selection and tracking works in conjointly with pose estimation. From pose estimation, the resultant projection matrix can be applied to that desired point and

project the aimpoint onto the target in the imaging data. DL implementation into this application would be the most beneficial to weapon guidance systems such as the LWS. Incorporating image processing techniques into weapon guidance systems could lead to autonomy in target determination of position and velocity, and target interception [19]. The limitations for aimpoint tracking from DL are homogenous to pose estimation. Image noise or clutter limits performance as it causes errors in computing the target aimpoint [19]. The primary goal of this research is to evaluate these two applications mentioned above through DL efforts and maximize their performance.

THIS PAGE INTENTIONALLY LEFT BLANK

### III. PROBLEM FORMULATION AND METHODOLOGY

The beginning of this section describes the software components and the implementation of them for the experimental approach. After the overview of the software design, the data utilized in the experimentation is detailed as well as the steps of augmenting it to fit the software. There are two datasets, both provided from Lockheed Martin. The last section expresses the three-phase method of the experiment outlined in section 1.C Approach: Phase 1 keypoint detection, Phase 2 pose estimation and Phase 3 aimpoint selection.

#### A. SOFTWARE

##### 1. PyTorch

As stated above in section 2.A, the AI community has developed a library of open-source ML and DL algorithms and implementations. The software utilized for Phase 1 of the experimentation is PyTorch. PyTorch is an open-source framework based on the Torch library. This framework was developed by Facebook's AI Research lab and its primary used for computer vision applications [20]. Implementation of this software only requires installation on a computer from its repository on GitHub. PyTorch is versatile in all programming languages. Python was the chosen language for its application in Phase 1.

##### 2. Jupyter Notebook

Jupyter Notebook is another open-source prototype in the AI community [21]. It is commonly used for being a web-based computer environment. The environment's interface, or known as a notebook, allows for real-time manipulation and execution of algorithms by being connected to a kernel [21]. Kernels handle the user's request by executing and replying. The advantage of this interface is the capability of having immediate results which improves efficiency in code implementation. An example of a notebook in the Jupyter Notebook interface is shown in Figure 11.

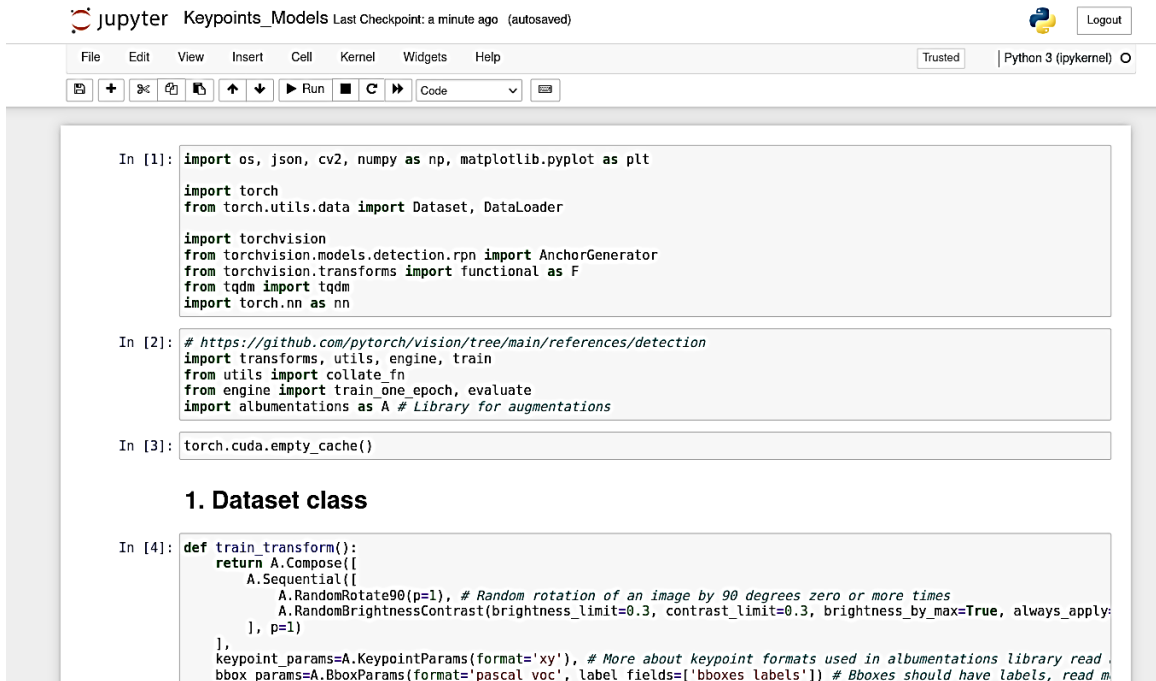


Figure 11. Jupyter Notebook Interface.

The conjunction of Jupyter Notebook and PyTorch provided the design to execute Phase 1 of the experimentation. Jupyter Notebook is also versatile in all programming languages. Using Jupyter Notebook as the environmental interface, the repository of Pytorch was able to be accessed. From the repository, the Keypoint R-CNN algorithm, mentioned in Chapter II, was implemented in a notebook. There were adjustments to the original source code of this algorithm to fit the bounds of the experimentation. These adjustments will be further explained in sections below on Phase 1 of experimentation. While Keypoint R-CNN is utilized and implemented for its applications, the design of Phase 1 results in an authentic coding program specifically for this research.

### 3. MATLAB

The Matrix Laboratory (MATLAB) was the software utilized for Phase 2 and 3 of experimentation. Similar to Jupyter Notebook, MATLAB is a computational environment. However, this software is numeric-based while Jupyter Notebook was web-based. MATLAB was developed by MathWorks with the original intent of being an interactive matrix calculator [22]. Recent developments in AI have transformed MATLAB into one

of the most common computer environments for ML and DL applications. MATLAB is equivalent to Jupyter Notebook in terms of operating and executing code. However, its programming languages are limited to C or C++ [22]. Figure 12 shows an example of the MATLAB interface.

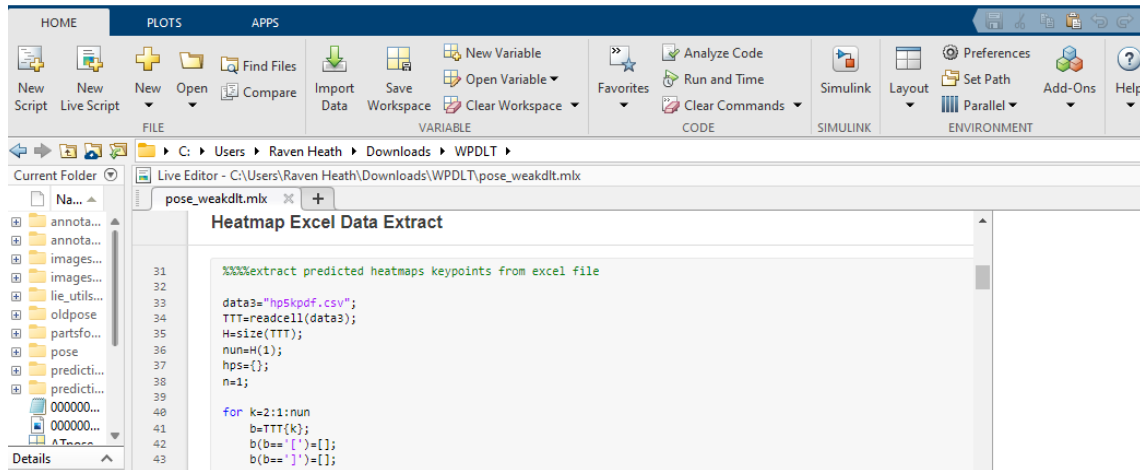


Figure 12. MATLAB Interface.

Phase 2 and Phase 3 of experimentation are based on pose estimation, mentioned in section 2.C above. The method chosen for pose estimation is derived from WPDLT. WPDLT determines the 3D-2D point correspondences from given pose of a camera. This method was the center of the coding program designed for Phase 2 and 3. All implementation and execution of program was carried through MATLAB.

## B. DATASETS

The data for the experiment was synthetically created to simulate real-world expectations. UAV models were captured at different poses in a testbed. The resulting images were then altered to simulate different imaging limitations that are expected for real-world applications. These limitations include but are not limited to atmospheric effects such as weather and lighting and blurring of the model to simulated turbulence. There are two sets of data that were evaluated in the experimentation. Both datasets were provided



by LM. This section details the format of the datasets, their characteristics. Steps taken to augment the data for experimentation will be discussed in the Phase sections.

### 1. No Atmospheric Effects Dataset

This dataset will be referenced to as the no atmospheric effects (NOAT) dataset. This dataset is comprised of 100,000 synthetic images of a modeled Mongoose UAV in an unaffacting background. All images are in grayscale. Each image is a size of 255x255 pixels. The Mongoose UAV model is at different poses but consistent sizing throughout the images. Figure 13 shows examples of the NOAT dataset.



Figure 13. NOAT Dataset Images.

Along with every image is an annotation file. These files are JavaScript Object Notation (JSON) files, a common format for DL algorithms. In them contains the required pixel data for training the DL algorithms. A visual example of the dataset's annotation files can be seen in Figure 14.

```

{
  "image id" : "085008",
  "reflectance" : 0.537699,
  "distance" : 4000,
  "fried param" : -1,
  "pose" : [1.87565, -0.677549, -1.78229, -2.27374e-13, 4.54747e-13, 4000],
  "bbox [col1, col2, row1, row2]" : [122, 141, 94, 171],
  "parts" : {
    "nose" : [128.432, 111.2],
    "right_wing" : [126.898, 97.0924],
    "left_wing" : [130.466, 162.28],
    "tail" : [125.661, 157.626],
    "tail_top" : [139.552, 158.842],
    "right_horizontal_stabilizer" : [125.211, 145.176],
    "left_horizontal_stabilizer" : [126.262, 167.542]
  }
}

```

Figure 14. NOAT Dataset Annotation File.

The image id is the image number. For the example above, this JSON file is associated with image 85,008. The bbox is the parameters of the bounding box for object detection. Typical bounding box formats provide the x-y pixel coordinates for two opposing corners on the bounding box, lower left, and upper right. The parts component is comprised of the keypoints for the NOAT dataset. This dataset has 7 keypoints: nose, right wing, left wing, tail, tail top, right horizontal stabilizer and left horizontal stabilizer. Each keypoint has a x-y pixel coordinate. The pose component of the JSON file describes the 2D pose correspondence for this specific image which will be utilized in Phase 2 and 3. The data provided in the JSON files are the ground-truths for the experimentation models and the baseline when assessing the algorithms' performance.

## 2. Atmospheric Effects Dataset

This dataset will be referenced as the atmospheric effects (AT) dataset. It has the same general format as the NOAT dataset since both are provided from LM. Example images of the AT dataset can be seen in Figure 15.



Figure 15. AT Dataset Images.

The difference between the NOAT and AT datasets are the synthetic atmospheric effects applied to alter the image and UAV model. From the figure, these effects include lighting, object blurring, simulated weather, and differing model sizing. These effects present the challenge of training in-disguisable pixels between the object and background. Inconsistent model sizing presents the challenge of different perspectives of ranges for the algorithms. The AT dataset is comprised of 100,000 images of the mongoose UAV model. The image size is the same as the NOAT dataset. Figure 16 shows an example of the JSON files associated with this dataset.

```
{
  "id" : "0085001",
  "target" : 3,
  "distance" : 2299.63,
  "contrast" : 43.9984,
  "noise" : 5.1,
  "pose" : [-1.92126, -0.356193, -0.276951, -0.0705069, 0.0207578, 2299.63],
  "bbox [col1, col2, row1, row2]" : [108, 135, 122, 134],
  "parts" : {
    "nose" : [134.114, 129.029],
    "center" : [122.636, 127.024],
    "wing_right" : [131.03, 122.028],
    "wing_left" : [119.113, 133.082],
    "tail" : [108.384, 125.752]
  }
}
```

Figure 16. AT Dataset Annotation File.

The general format is the same as seen above with the NOAT JSON files. The difference is the keypoints. For the AT dataset, there are five keypoints provided: nose,

center, right wing, left wing and tail. This dataset provides a keypoint for the center of gravity of the UAV model and removes keypoints of features such as the tail top and stabilizers. Similar to above, the data provided in each JSON file is the ground-truth data to evaluate against the accuracy of the algorithms in the experimentation.

The implementation of these datasets are separated into the task of training and evaluating. The datasets were split 85–15 for training and testing. As mentioned above in Chapter II, the larger the volume for training, the higher the accuracy of the resultant model. The next section breaks down the specific methodology of each phase in the experimentation including data augmentation and implementation, and design overview.

### **C. PHASE 1: KEYPOINT DETECTION**

Phase 1 of the experiment developed the keypoint detection model for this study. The data augmentation methodology discusses the implementation of the datasets to fit the format of Keypoint R-CNN. The algorithm implementation section is an overview breakdown of the adaptation of Keypoint R-CNN to fit the scope of the study. Two unique models resulted in the training of the DL algorithm on the LM datasets. Training was conducted for multiple iterations to optimize the models. Training results determined two scenarios of predictions, keypoints and heatmaps. The evaluation and predictions section discusses the methodologies of obtaining the predictions and evaluating the performance of the models against the ground-truth keypoints.

#### **1. Data Augmentation**

To fit the datasets into the format of Keypoint R-CNN, tools from Pytorch were utilized. Pytorch has an object called the Dataset class which allows for easy manipulation to alter data into its expected format. The Dataset class object for Keypoint R-CNN is broken down below [20, 23, 24]:

- boxes ([N, 4]): a channel of the pixel coordinates of the N number of bounding boxes in [x0, y0, x1, y1] (lower left, upper right) format.
- area ([N]): a channel of the pixel area of the N number of bounding boxes.

- keypoints ( $[N, K, 3]$ ): a channel for each one of the  $N$  number of objects in an image. This channel contains the  $K$  number of keypoints in  $[x, y, v]$  pixel coordinate format.
- labels ( $[N]$ ): a channel of the class label for each  $N$  number of bounding boxes. 0 always represents the background class. 1 represents the object class which in this case is “Mongoose.”
- image\_id ( $[1]$ ): a channel of the image number.

Keypoint R-CNN has a unique visibility factor for keypoint detection. This visibility factor, denoted as  $v$  above, can be the number: 0, 1 or 2. Denoted as 0 dictates that the keypoint is not visible and is not marked in the image [24]. Visibility denoted as 1 dictates that the keypoint is marked but is not visible and denoted as 2 dictates that it is both marked and visible in the image [24]. Since both datasets were provided and not generated as a part of the experimentation, assumptions had to be made about this visibility factor. For the NOAT dataset, all keypoints were denoted as 2 except the tail top, and right and left horizontal stabilizers; those keypoints were denoted as 1. The factor that made this decision is due to these three keypoints are not always visible in each image due to the pose of the object. For the AT dataset, visibility was denoted as 2 for all keypoints. The task of going through each image and editing its associate JSON file to have complete accuracy of the visibility factor was not determined necessary for the experimentation. Additional code was added to The Dataset class object source code to fit the LM datasets to this format.

The next step taken for data augmentation was altering the original source code of Keypoint R-CNN to fit its OKS to the LM datasets. As mentioned in Chapter II, this algorithm was trained to predict 17 keypoints for human pose. Thus, the object scale factors for the OKS are fit to human pose. These factors are inputted as coefficients in the source code. The coefficients chosen for the datasets of this study were 0.1 for all keypoints. The assumption was made that there is not large object scaling for the keypoints since the datasets are UAV models. This assumption also limits the algorithm to a closer predicting range around the intended keypoints.

## 2. Algorithm Implementation

The implementation of the Keypoint R-CNN algorithm was done by developing a model custom to the scope of the study. Figure 17 shows an example of the code for the model.

## 5. Training

```
In [9]: def get_model(num_keypoints, train_kptHead=False, weights_path=None):
anchor_generator = AnchorGenerator(sizes=(32, 64, 128, 256, 512), aspect_ratios=(0.5, 1.0, 2.0,))
model = torchvision.models.detection.keypointrcnn_resnet50_fpn(pretrained=False,
                                                              progress=True,
                                                              pretrained_backbone=True,
                                                              trainable_backbone_layers=5,
                                                              num_keypoints=num_keypoints,
                                                              num_classes = 2,
                                                              rpn_anchor_generator=anchor_generator,
                                                              min_size=640)

out = nn.ConvTranspose2d(512, num_keypoints, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
model.roi_heads.keypoint_predictor.kps_score_lowres = out

if weights_path:
    state_dict = torch.load(weights_path)
    model.load_state_dict(state_dict)

return model
```

Figure 17. Keypoint Model Code.

This model line in Figure 17, defines the overall bounds that will be trained. The “pretrained=False” parameter defines that this model will not use the trained Keypoint R-CNN model. The weights would not perform well on the LM datasets since the data is not for human pose. The “pretrained\_backbone,” and “trainable\_backbone\_layers” defines that the intention is to train all of NN layers within the algorithm. “num\_keypoints” defines the number of keypoints that the model will be trained on. There are two separate models for the datasets since they have a different number of keypoints. The “out” line found below the “model” line defines the activation function, mentioned in Chapter II, for the model.

## 3. Training

The initial iterations of training the models were done to optimize the models for maximum performance. Preliminary training resulted in inadequate training accuracy that did not reach above fifty percent. The method to improve the models were altering and

evaluating its hyperparameters. Hyperparameters are used to affect the learning of a NN [25]. Using a combination of hyperparameters can optimize the learning approach the algorithm takes [25]. Figure 18 shows an example of how these parameters are implemented into the algorithm for training.

```
In [12]: model = get_model(num_keypoints = 5, train_kptHead=5)
model.to(device)

params = [p for p in model.parameters() if p.requires_grad]

#lr=0.0025
optimizer = torch.optim.SGD(params, lr=0.2, momentum=0.9, weight_decay=0.0001)
lr_scheduler = torch.optim.lr_scheduler.MultiStepLR(optimizer, milestones=[12,18], gamma=0.1, verbose=True)

num_epochs = 20
```

Adjusting learning rate of group 0 to 2.0000e-01.

Figure 18. Hyperparameters Code.

The hyperparameters that were utilized for developing an optimal learning approach were learning rate, weight decay, momentum, a multi-step learning rate scheduler. These were implemented using a Pytorch optimizer package called “torch.optim” [26]. The specific optimizer used was the stochastic gradient descent (SGD). SGD reduces redundancy in the data by selecting a random batch from the entire dataset to calculate its gradient [27]. The functions of each hyperparameter used is broken down below [25, 27, 28]:

- Learning Rate: Step size of weight updates during training.
- Weight Decay: Penalty factor that is added to the loss function.
- Momentum: Parameter specific for SGD. It increases the speed at which learning is done.
- Multi-Step Learning Rate Scheduler (MSLRS): This is a subset of the optimizer package in Pytorch. Its function is to decay the learning rate by the gamma component once the number of epochs (training iterations) reaches the defined milestones.

The final decision of hyperparameters chosen for the models can be seen in Table 1.

Table 1. Hyperparameters of Models

<b>Model</b>	<b>Epochs</b>	<b>Batches</b>	<b>Learning Rate</b>	<b>Weight Decay</b>	<b>Momentum</b>	<b>MSLRS Milestones</b>
NOAT - 7 Keypoints	15	16	0.2	0.0001	0.9	[7, 13]
AT - 5 Keypoints	20	16	0.2	0.0001	0.9	[12,18]

Table 1 includes the final decision on the epochs and batches for the models as well. Epochs are the number of complete passes that the model is trained on [5]. The batch size is the amount of data samples that are uses in one iteration of training [5]. The preliminary stages of training for an optimal model had 50 epochs to evaluate the behavior of the model. After an epoch of training, the model was then validated against sample test to estimate its trend of loss and accuracy as it learned. The average time for completion of a single epoch and validation was one hour. Table 2 shows the validation performance results of the final models.

Table 2. Validation Performance of Finals Models

<b>Model</b>	<b>Loss</b>	<b>Bounding Box AP</b>	<b>Keypoint AP</b>
NOAT - 7 Keypoints	0.45	0.97	0.78
AT – 5 Keypoints	1.20	0.86	0.65

After the completed training of each model’s set of epochs, the model weights were saved. These weights define the behavior of the model and were used to determine the predictions discussed below.



#### 4. Predictions and Evaluations

The predictions from the models were presented in a similar fashion to the input data format. The breakdown of the output from the models is seen below [24]:

- boxes ( $[N, 4]$ ): The predicted pixel coordinates of the  $N$  number of bounding boxes.
- labels ( $[N]$ ): The predicted class label of the  $N$  number of objects.
- scores ( $[N]$ ): The score of each prediction. The score value ranges from 0 to 1 and defines prediction accuracy. The closer to 1, the more accuracy the prediction is.
- keypoints ( $[N, K, 3]$ ): The predicted locations of the keypoints for each  $N$  number of objects in the image.
- heatmaps ( $[N, K, 56, 56]$ ): A predicted  $56 \times 56$  channel for  $K$  number of keypoints for each  $N$  number of objects.

All predictions from the model are based on the bounding boxes. While there is only a single object in each image, the model could predict several bounding boxes due to confusing the background with UAV. For each bounding box predicted, there will be that same number of scores, keypoints, and heatmaps. The value of the scores determined which bounding box is the most accurate for the prediction. To obtain a single set of predictions for an image, the maximum score that was above 0.7 was selected. If the prediction was high, then this resulted in a set of predictions for a single bounding box. If the prediction was below 0.7, then this resulted in no predictions for that image. There is also the case that the algorithm did not predict for an image due to limitations mentioned in section B.2 of this chapter.

There are two outputs that provide keypoint prediction for the model, keypoints and heatmaps. The keypoint output is based on the default prediction that the model provides for the chosen bounding box. The heatmap output is a scoring map of possible keypoint

locations. This scoring map is analogous to heatmaps seen for weathering imaging. Figure 19 shows examples of the heatmaps.

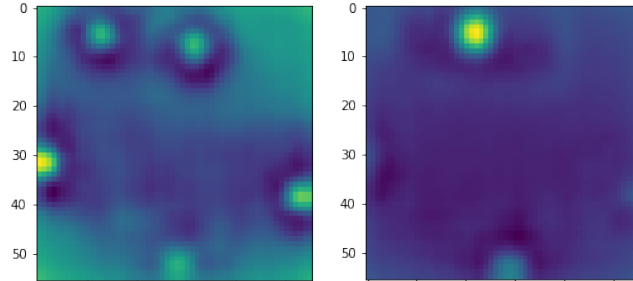


Figure 19. Heatmap Output.

As illustrated in Figure 19, the heatmaps are hotter, or lighter in color, around the locations of the keypoints. There is a heatmap for each keypoint. Since this output is a channel of scores, this maximum heat score for a single keypoint could be converted into the x-y pixel coordinates for that keypoint. This conversion leads to a second set of predicted keypoints from a single model. The reason for utilizing the heatmap out is improve performance of the model. A limitation to accuracy is bearing ambiguity of the UAV. Since the datasets are comprised of the UAV at various poses, there are certain positions that can confuse the algorithm. This confusion leads to the model mistaking keypoints and predicting them in the same location. During the preliminary stages of deriving an optimal model, this behavior was most commonly seen between the right- and left-wing tip keypoints. Since the heatmaps focus on scoring around the location of a keypoint, there is the possibility this limits the confusion from the model.

Training and predictions were conducted on the 85–15 split mentioned above in section B. The modes were trained on 85,000 images and predictions were inferred for the remaining 15,000. All predictions were saved in the same general format as the datasets were received. The predicted and ground-truth bounding boxes and keypoints were overlaid on top of the inferred images and saved respectively. The predicted and ground-truth pixel coordinates of the bounding boxes and keypoints were saved into excel files. These excel files were utilized in Phase 2 and 3. Keypoint prediction accuracy was

evaluated through the metrics of precision and recall. Precision is the number of correct positive predictions; recall is the number of correct positive predictions out of all predictions [29, 30]. The difference between the two metrics is that recall includes false positive predictions [30]. Recall quantifies the bearing ambiguity limitation mentioned above. The result of this evaluation is discussed below in Chapter IV.

## D. PHASE 2: POSE ESTIMATION

Phase 2 in utilized the keypoint predictions to evaluate DL capability in the application of pose estimation. This section provides an overview of the methodologies for augmenting prediction data from Phase 1, modeling pose estimation and evaluating pose performance.

### 1. Data Augmentation

The excel files of the predicted and ground-truth bounding boxes and keypoints were read into MATLAB and adapted to fit the format of the pose estimation model. The annotation files of the LM datasets were also utilized. Mentioned above in section B, the pose line in these annotation files are the ground-truth 2D pose of UAV in the image. Two additional annotation files provided by LM were also utilized for the model. These files contained the 3D points on the UAV in its frame. Figure 20 shows an example of these files.

```
{
  "model" : "mongoose_no_landing_gear.obj",
  "parts" : {
    "nose" : [0.332648, -0.00045693, -0.00469297],
    "center" : [-0.111739, 0, -0.0268089],
    "wing_right" : [-0.0176648, 0.573003, -0.017927],
    "wing_left" : [-0.0176648, -0.573003, -0.017927],
    "tail" : [-0.667325, 0.0068952, -0.00442388]
  }
}
```

Figure 20. 3D Target Points Annotation File.

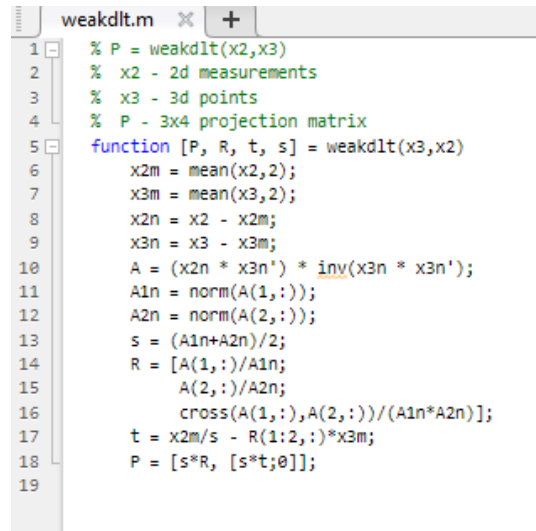
These files are unique to each of the LM datasets. Based on the theory of WPDLT, mentioned in Chapter II section C.1, these are the 3D points needed to calculate the projection matrix.

## 2. Keypoint Implementation

The pose estimation model was coded in MATLAB as a function of WPDLT. The following equation was used in WPDLT [12],

$$\xi(\theta) = W - s\bar{R} \left( B_0 + \sum_{i=1}^k c_i B_i \right) - \bar{T} \mathbf{1}^T$$

where  $\bar{R}$  is a 2x3 rotation matrix,  $\bar{T}$  is a translation matrix,  $W$  is a matrix of the 2D keypoints and  $s$  is a scalar value. This equation calculates the reprojection error [12]. This equation was implemented into MATLAB as a function and can be seen in Figure 21.



```

1 % P = weakdlt(x2,x3)
2 % x2 - 2d measurements
3 % x3 - 3d points
4 % P - 3x4 projection matrix
5 function [P, R, t, s] = weakdlt(x3,x2)
6     x2m = mean(x2,2);
7     x3m = mean(x3,2);
8     x2n = x2 - x2m;
9     x3n = x3 - x3m;
10    A = (x2n * x3n') * inv(x3n * x3n');
11    A1n = norm(A(1,:));
12    A2n = norm(A(2,:));
13    s = (A1n+A2n)/2;
14    R = [A(1,:)/A1n;
15         A(2,:)/A2n;
16         cross(A(1,:),A(2,:))/(A1n*A2n)];
17    t = x2m/s - R(1:2,:)*x3m;
18    P = [s*R, [s*t;0]];
19

```

Figure 21. WPDLT MATLAB Function. Source: [12].

The WPDLT function takes an input of the 3D points on the target from the file shown in Figure 20, and the 2D keypoints. It then conducts the mathematical operations of WPDLT and outputs a projection matrix, rotation matrix, a translation matrix, and the subject distance. The resultant projection matrix was then multiplied to the 3D points to

determine the corresponding 2D image points. This model ran against the predicted keypoints provided from Phase 1 and the ground-truth keypoints of each dataset. The predicted pose was determined from computing a logarithm of the predicted rotation matrix. This computation resulted in predicted Euler angles for the model.

### 3. Predictions and Evaluations

The two sets of predicted keypoints for each dataset were inferred on the WPDLT model. Each image had its own unique outputted projection and rotation matrix. The predicted keypoints and predicted and ground-truth 2D image points were overlaid on the testing images for visual results. The ground-truth and predicted poses were saved for evaluation. The projection matrix for each inference was also saved for Phase 3. The evaluation metric chosen for pose estimation was the root mean squared error (RMSE). RMSE is the standard deviation of the prediction errors [31]. The equation used to calculate RMSE for the experimentation can be seen below [31]:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n e_i^2}$$

RMSE measures the accuracy of the model. The accuracy in question is the error between the pose prediction from the DL model versus the ground-truth pose values. RMSE was calculated in MATLAB and results will be discussed in Chapter IV.

### E. PHASE 3: AIMPOINT SELECTION AND TRACKING

Aimpoint selection was an extension of the determined pose in Phase 2. Utilizing the projection matrix results from the 2D keypoints, the DL model was evaluated in its usefulness of this application. This section discusses the methodologies of determining aimpoint selection from the results of the two previous phases.

## 1. Inference

To achieve aimpoint selection, 3D points on the target were chosen to represent the intended placement of the aimpoint. These points are in the same format seen in the 3D points annotation file in Figure 20. Table 3 shows the points chosen for the datasets.

Table 3. 3D Target Aimpoints.

Dataset	3D points
NOAT	[-0.225; 0; -0.105]
AT	[0.215; 0; -0.0110]

The 3D point for the NOAT dataset was chosen to be near the center of gravity of the UAV. It is half between the wing tips. The 3D point for the AT dataset was chosen to be under the nose of the UAV. Multiplying these 3D points and the projection matrices from on the predicted 2D keypoints and ground-truth keypoints, the 2D image correspondence for the aimpoint was determined.

## 2. Evaluations and Predictions

The two sets of keypoint predictions for each dataset were evaluated for their potential in aimpoint selection. The ground-truth and predicted aimpoints were overlaid on the test image. The determined x-y pixel coordinates of the aimpoint were saved. The evaluation metric for aimpoint selection was RMSE. The RMSE of this section was calculated from the error between the predicted 2D image correspondences and ground-truth correspondences. The results of this phase will be discussed below in Chapter IV.

THIS PAGE INTENTIONALLY LEFT BLANK

## IV. SIMULATION RESULTS AND DL PERFORMANCE EVALUATION

### A. PHASE 1: KEYPOINT DETECTION

#### 1. Bounding Boxes

The evaluation metric of the bounding boxes was provided from the algorithm after training. The average precision (AP) of the NOAT dataset was higher than the AP of the AT dataset. The NOAT model resulted in the most accurate performance for predicting bounding boxes coordinates matching the ground-truths. The prediction percentage was calculated as an additional metric. Mentioned in Chapter III section C.3, the bounding box that was accepted was based on the highest score above 0.7. If there was no score, then the prediction was not counted as accurate nor further evaluated. Table 4 quantifies the models' performance for bounding boxes.

Table 4. Summary of Bounding Box Prediction Results.

Model	Number of Ground-Truths	Number of Predictions	Prediction Percentage	Average Precision
NOAT	15000	15000	100%	0.97
AT	15000	14984	99.8%	0.86

From the table, the AT dataset did not have predictions 14 images. In comparison, the prediction percentage is efficient for both. The sections below further evaluate the bounding box predictions and their limitations.

#### *a. Dataset 1*

The NOAT dataset bounding boxes were almost identical to that of the ground-truths. Figure 22 shows visual results of the prediction.



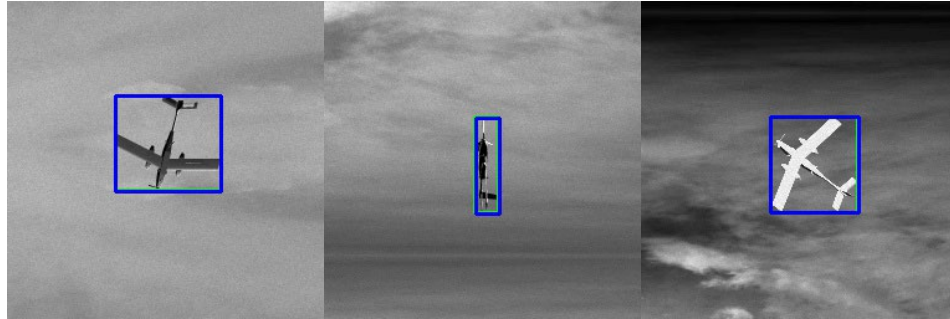


Figure 22. NOAT Model Bounding Box Predictions.

From the Figure 22, blue bounding box is the ground-truth, and the green bounding box is the prediction. As illustrated in the sample images, the bounding boxes are overlapping each other with near perfect symmetry. The NOAT had no false predictions for the bounding box and performed with pronounced accuracy.

***b. Dataset 2***

While the AT model had a high prediction percentage, the actual results are variable. Figure 23 shows quality predictions from the Model.



Figure 23. Optimal Bounding Box Predictions from AT Model.

These results show little to no deviation from the ground-truths. There is clear overlapping of the bounding boxes regardless of the varying sizes of the UAV. Figure 24 shows average predictions from the Model.

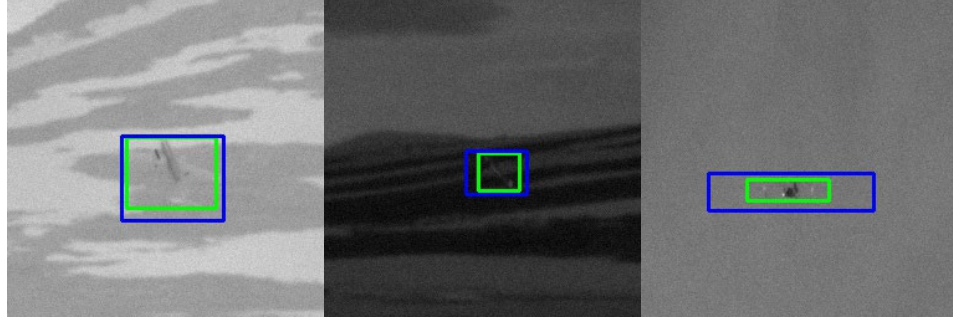


Figure 24. Avery Bounding Box Predictions from AT Model.

These results illustrate the AT model's true accuracy for prediction bounding boxes. An AP of 0.85 is a sufficient level of performance but there is still an average amount of error. As seen in the figure, the green predicted bound boxes still contain the target. However, there is a level of cropping with these predictions which results in a less accurate prediction for the keypoints as well. The first image in Figure 24 shows the prediction cutting off the wing tips of the target. The prediction keypoints for that image will result not at their desired location due to this. The cropping in the predicted bounding boxes were a result of the simulated weather in the dataset. Simulating a background and weather presented a high volume of image noise and blurred the target pixels with surrounding background pixels. This became challenging for the algorithm to learn on and resulted in bounding boxes that overcompensating and cropping closer to the disguisable areas of target. Figure 25 shows the test images where no bounding box predictions were made.



Figure 25. Examples of No Predictions from AT Model.

The images that had no bounding box prediction from the AT model all had extreme simulate backgrounds and weathering affects. As seen above in Figure 25 it would be hard to identify where the UAV was in the image if the ground-truth bounding box was not provided. The AT model resulted in only 16 images not having predictions. While the AP of the model allows for a small margin of error, the prediction capability of the model against the AT dataset showed promising results. The performance of this model is efficient and accurate in detecting targets under normal to extreme conditions of imaging. Overall, the prediction results from both models demonstrated the usefulness of DL for target detection, which is the initial stage for weapon guidance systems.

## **2. Keypoints**

As mentioned in Chapter III, the inference of the models resulted in two sets of keypoints for each. This section will present the visual results of the keypoints predictions and evaluate its accuracy. The validation of each model provided a keypoint AP of 0.78 for the NOAT model and a keypoint AP of 0.65 for AT model. The evaluation metric for the predictions were precision and recall. The results of the evaluation metrics provided the limitations and challenges of keypoint detection.

### ***a. Dataset 1***

#### **(1) Default Keypoints**

The keypoint predictions defaulted from the bounding box predictions for the NOAT model had expected results. There was high accuracy in predicting the keypoints equal to their ground-truths. Figure 26 shows sample keypoint predictions.

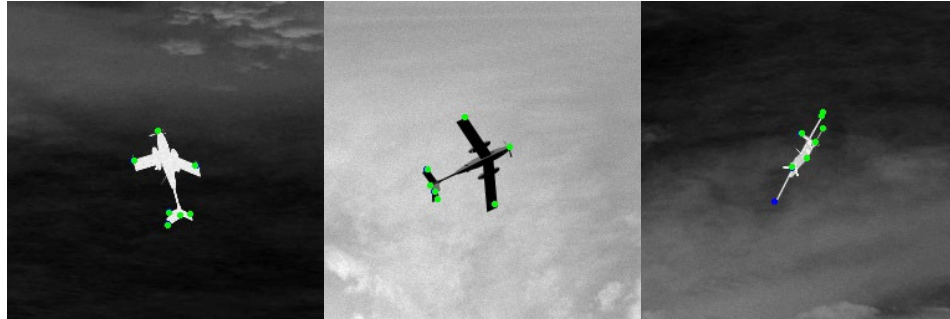


Figure 26. NOAT Model Default Keypoint Predictions.

In the figure, the green points are the predicted keypoints and the blue points are the ground-truths. The predictions are almost undisguisable from the ground-truths as they overlap exactly. The only limitation of these predictions is shown in the right most image of Figure 26. There are 7 keypoints predicted in the image but only 6 are shown. The algorithm placed two keypoints on one location. Explained in detail in Chapter III, the primary challenge faced for keypoint detection is bearing and pose ambiguity caused by the symmetric shape of the UAV target. Since the target is at a unique pose for every image in the dataset, learning to differentiate various poses is limited the accuracy of the model.

The precision and recall of the predicted keypoints was calculated by finding true and false positives. A true positive is a prediction that matches its ground-truth correctly. A false positive is a prediction that mistakes itself for another class. These metrics are typically used for classification and not detection. However, these metrics were implemented by setting a class to each keypoint value based on the ground-truths. If the predicted keypoint of the corresponding index was within a 2% margin of the ground-truth, then it had the same keypoint class. If the prediction was not within this margin, then its value was assessed against the 14 ground-truths, x-y coordinates of the 7 keypoints, for the corresponding image. The closest ground-truth value to that prediction was the assigned class for the prediction. Figure 27 illustrates the results in a confusion matrix.

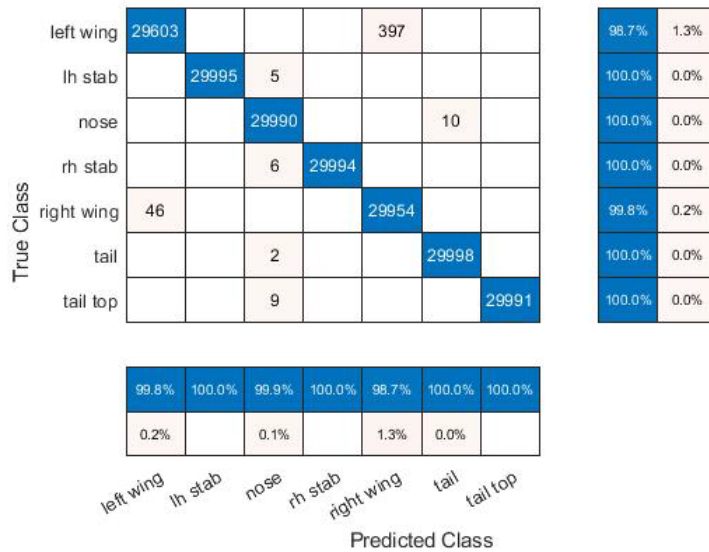


Figure 27. NOAT Default Keypoint Confusion Matrix.

The confusion matrix shows the total number of predictions of the specific keypoint in each cell. Values in the diagonal indicate the true positives [29]. Values outside the diagonal indicate the false positives [29]. The rows of the matrix are the ground-truths, and the columns are the predicted keypoint classes. The matrices on the right and bottom of the chart summaries the prediction accuracy for the NOAT model. The confusion matrix showed that the model has high accuracy for detecting keypoints. The most incorrect predictions were for the left and right wings. However, the limitations from bearing ambiguity are not significant within this model. Obtaining an accuracy of 98.7% or higher for detection is exceptional performance for an AI system based on DL.

## (2) Heatmap Keypoints

The process of converting the predicted heatmaps into keypoints coordinates was discussed in Chapter 3. The purpose of obtaining these keypoints was to evaluate if heatmaps could be a potential solution to the bearing and pose ambiguity of targets in images. Figure 28 shows visual predictions of heatmap keypoints.

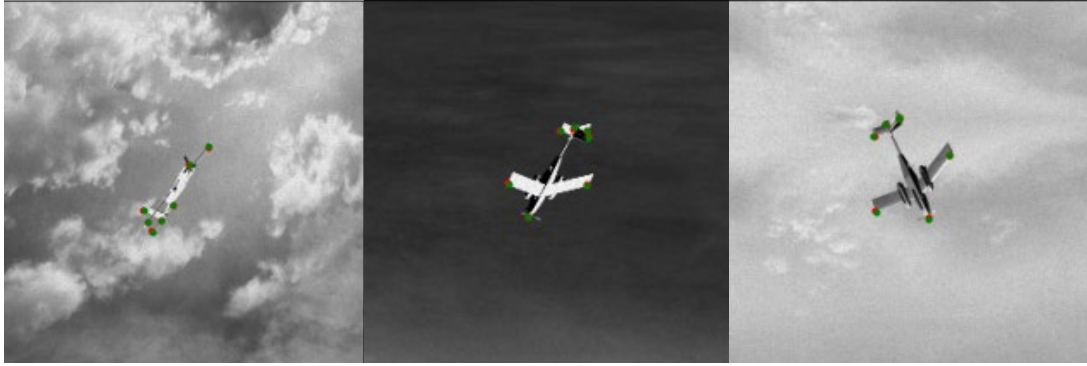


Figure 28. Optimal Heatmap Keypoint Predictions from NOAT Model.

In Figure 28, the green points are the ground-truths, and the red points are the predicted heatmap keypoints. These results show similar performance to the default keypoint prediction. The accuracy seen in the images above show that the heatmap prediction provided the desired keypoint location. However, not all of the heatmap predictions resulted in this level of accuracy. Figure 29 shows poor results from these predictions.

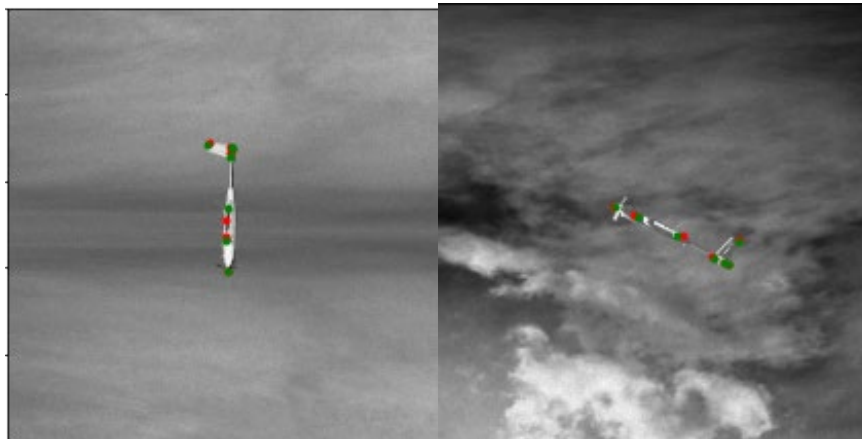


Figure 29. Poor Heatmap Keypoints Predictions from NOAT Model.

The images in Figure 29 illustrate how heatmaps can be limited to bearing ambiguity. The keypoint predictions for the heatmaps were accurate for target pose when all of the keypoints could be seen. However, pose that presented a side profile of the target resulted in lower performance from the NOAT model. The predicted keypoint, red, can be

seen clustering towards the middle of the target in the figure. The model was compensating for a keypoint that was not visible. Visibility was a factor in model training that was discussed in Chapter III. For this dataset, all wing keypoints were labeled as visible and tail features were labeled as marked but not visible as an assumption. This assumption limited the potential accuracy of this model and its heatmaps. The precision and recall performance metrics quantified this limitation. Figure 30 is the confusion matrix for the heatmap keypoints.

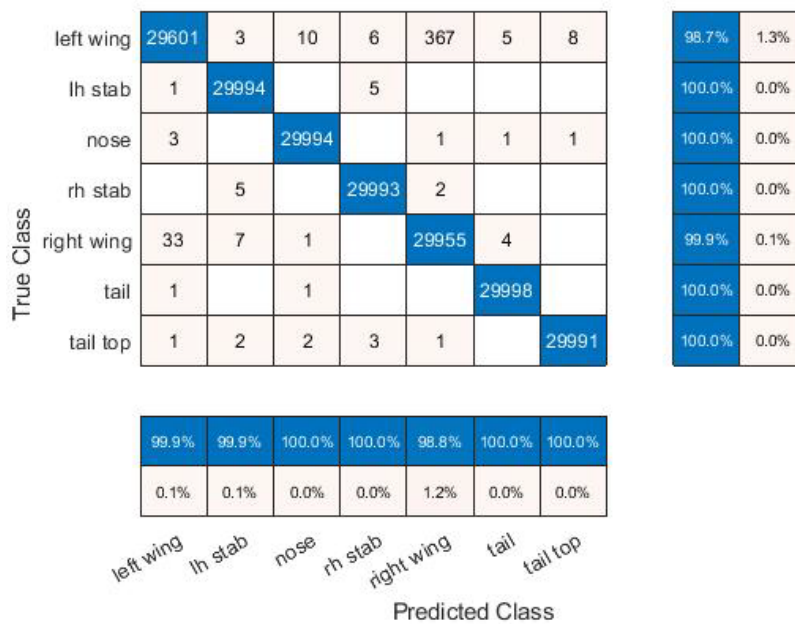


Figure 30. NOAT Heatmap Keypoint Confusion Matrix.

The confusion matrix shows that there are more false positives for the heatmap keypoints versus the keypoints from the bounding box. While the false positives between the wing tips has decreases, the model incorrectly predicted the wing tips as other features of the UAV. The left and right horizontal stabilizers have a much higher false prediction rate than seen with the default keypoints. However, the accuracy of the model in terms of prediction did not decrease. The accuracy is still 98.7% or higher for true positives. The limitation from bearing ambiguity did not have a significant effect on the heatmaps for the NOAT dataset.



*b. Dataset 2*

(1) Default Keypoints

The AT model's validation AP after training was significantly lower than the NOAT model. The evaluation in section 1.B discussed certain limitations seen in the bounding box performance that could have a secondary effect on keypoint detection. Figures 31 and 32 display those effects on the prediction of the default keypoints.

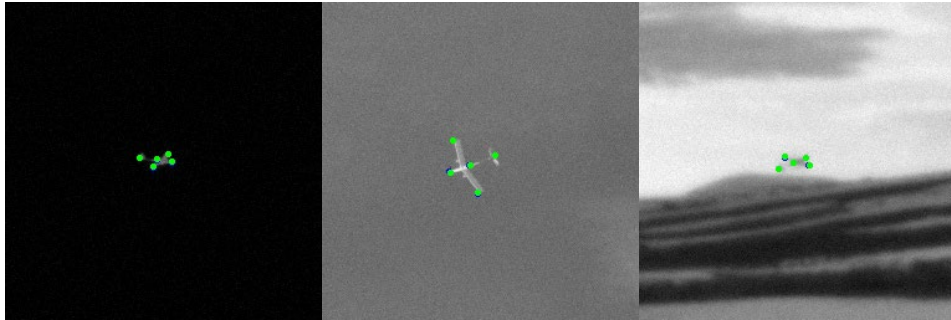


Figure 31. Optimal Default Keypoint Predictions from AT Model.

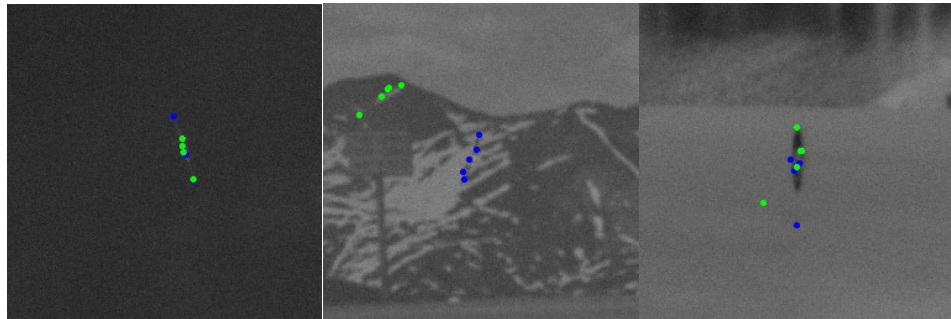


Figure 32. Poor Default Keypoint Predictions from AT Model.

The green points are the predicted keypoints and the blue points are the ground-truths in both figures. Figure 31 shows sufficient results from the AT model. The keypoints are aligned with the ground-truths and there is minimal difference. Figure 32 presents results of poor performance from the model. The second image in the figure is completely wrong for keypoint detection due to poor bounding box prediction. The model scored that bounding box for high accuracy, but it is detecting the mountain in the background instead



of the UAV. This further proves the limitations DL algorithms have when learning target pixels against obscure backgrounds. The keypoint predictions of the AT model were evaluated of the same performance metrics in section 2.B. Figure 33 is the confusion matrix for the default keypoint predictions.

True Class	center	28773	306	387	306	196	96.0%	4.0%
	left wing	1075	21116	1886	3777	2114	70.5%	29.5%
	nose	6054	4307	11429	4443	3735	38.1%	61.9%
	right wing	1092	3597	1846	21408	2025	71.4%	28.6%
	tail	1093	2494	2440	2707	21234	70.9%	29.1%
		75.5%	66.4%	63.5%	65.6%	72.5%		
		24.5%	33.6%	36.5%	34.4%	27.5%		
		center	left wing	nose	right wing	tail		
		Predicted Class						

Figure 33. AT Default Keypoint Confusion Matrix.

The confusion matrix shows low performance of the model predicting correct keypoints to ground-truths. The most accurate prediction was for the center keypoint. The bounding box prediction showed that the AT model compensated for the simulated background by cropping towards the center of the target. The center keypoint being the most accurate prediction supports the model’s behavior. The lowest performing prediction was the nose keypoint which would have often been cut out due to cropping. The AP from the confusion matrix is true to the AT model’s validation AP. While the expectation is for the model’s performance to improve during inferencing, this is an adequate result of accuracy based on the training results.

## (2) Heatmaps Keypoints

The heatmap keypoints has expected results for the AT model after considering the limitations already described. For heatmaps, there is a limitation due to bearing ambiguity. The AT dataset has several limitations applied to assess the effectiveness of DL implementation in a real-world setting. The challenges of image noise from simulated environments of lighting, weather, and landscape, and target blurring for simulated movement required the DL algorithm to learn and adjust on a significant scale. Figure 34 displays the visual results of the heatmap keypoints.

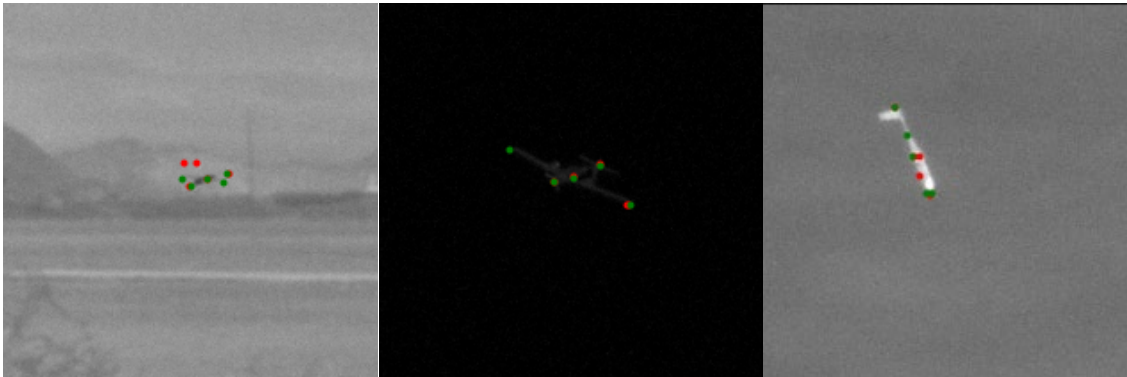


Figure 34. AT Model Heatmap Keypoint Predictions.

In Figure 34, the AT model's heatmaps resulted in a range of performance of keypoint detection. There were cases where it was extremely accurate as seen in the middle image. There were other cases where accuracy was lower due to factors mentioned above. The bearing ambiguity limitation was consistent between the two models. Figure 35 quantifies the heatmap keypoint prediction accuracy for the AT model in a confusion matrix.

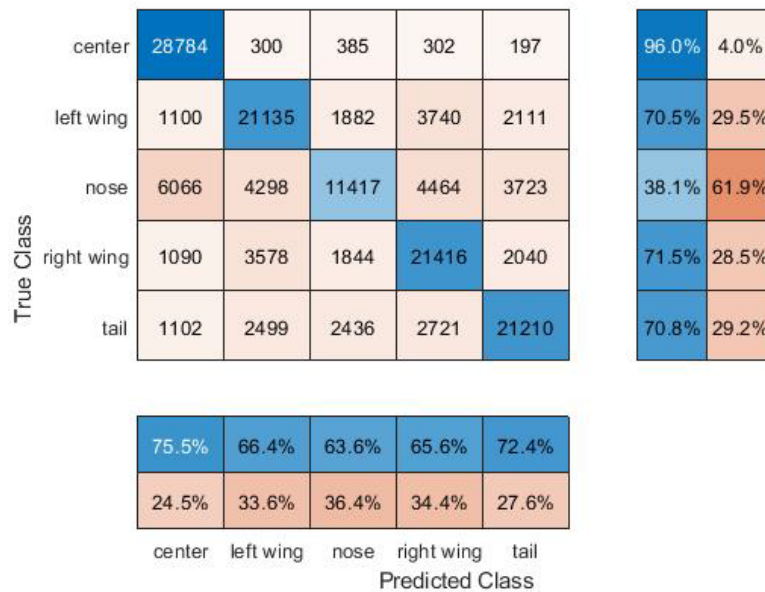


Figure 35. AT Heatmap Keypoint Confusion Matrix.

The heatmap confusion matrix showed consistent results with the confusion matrix for the default keypoint of this model. The center keypoint is the most accurately predicted keypoint and the nose keypoint is the less accurate. The significance this confusion matrix shows is that heatmaps are an acceptable form of determining keypoints for detection. The keypoints from the two methods performed identically to each other while still having different predictions. Heatmaps derived from DL algorithms provide a secondary approach for target detection which could be utilized in weapon guidance systems based on infrared technology.

## B. PHASE 2: POSE ESTIMATION

Pose estimation is one of the military applications this study wanted to evaluate for DL implementation. The process of determining the target pose using the predicted keypoints was explained in Chapter II. The primary question that Phase 2 and 3 answered was how the limitations found in the keypoint models' effected its usefulness in military applications. The evaluation metric utilized within the phase is RMSE, mentioned in Chapter II. Pose estimation results were based on the two methods of keypoint prediction.

## 1. Dataset 1

### a. *Default Keypoints*

As seen in section 1.B, the NOAT model had was extremely accurate in keypoint detection. The results from inferencing those predictions against the WPDLT function in MATLAB presented similar levels of accuracy. Figure 36 shows the pose estimation results for the default keypoints of the NOAT model.

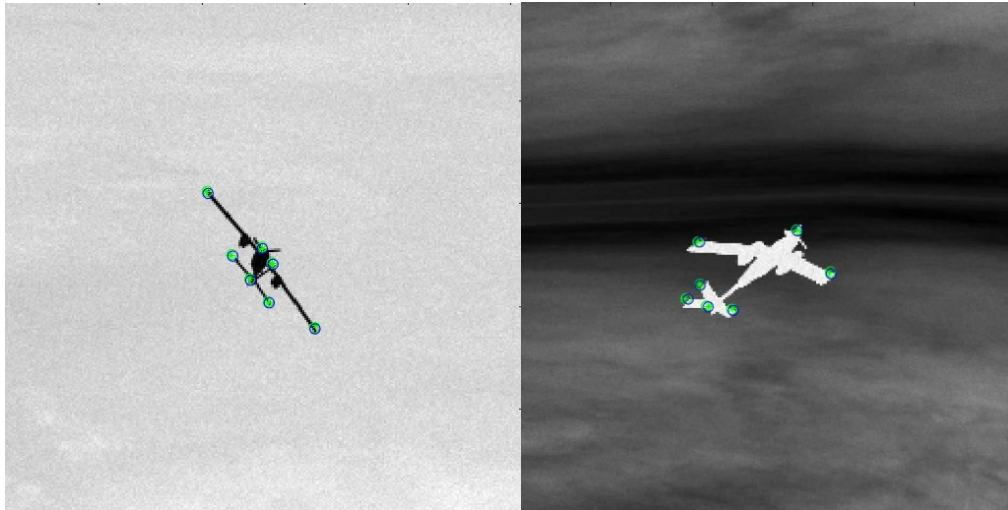


Figure 36. NOAT Pose Estimation from Default Keypoints.

The green asterisk markers in the images are the predicted keypoints of the model. The green open-circle marker is the 2D predicted pose and the blue open-circle marker is the 2D ground-truth pose. Figure 36 shows expected accuracy from the NOAT model. The pose circles are projected in the same location as the predicted keypoints. The limitations of these pose predictions can be seen in Figure 37.

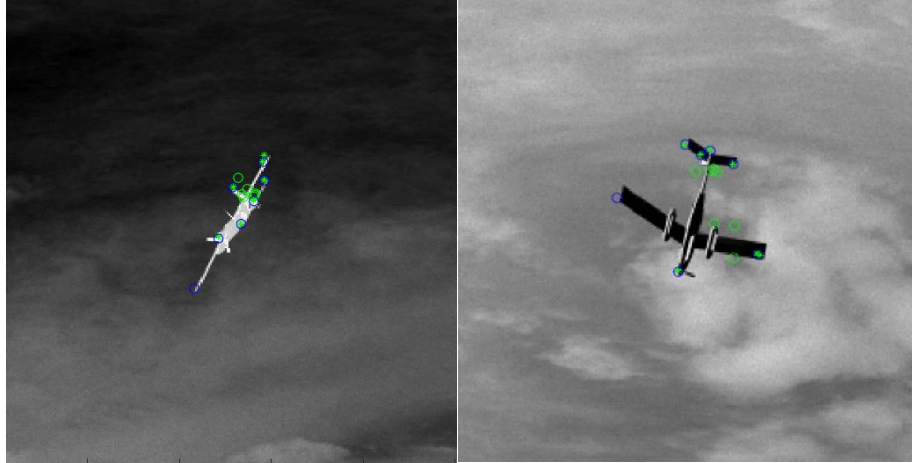


Figure 37. Poor Pose Estimation of Default Keypoints.

As seen in Figure 37, the predicted pose is extremely inaccurate in some cases. The cause is the limitation of bearing ambiguity seen within the NOAT model. The false positive predicted keypoints make pose estimation from the WPDLT nearly impossible. A limitation of the WPDLT function must be considered. The WPDLT presents advantageous for the method used to create the LM datasets. However, its accuracy is limited to that method. These datasets were created with the intention to simulate real-world scenarios. The various factors applied to make the datasets as realistic as possible potentially hinder the accuracy of WPDLT.

***b. Heatmap Keypoints***

The heatmap keypoints from the NOAT model were inferred in the same approach discussed above. Figures 38 and 39 display the various results from the heatmap predictions.



Figure 38. Optimal Pose Estimation of Heatmap Keypoints.

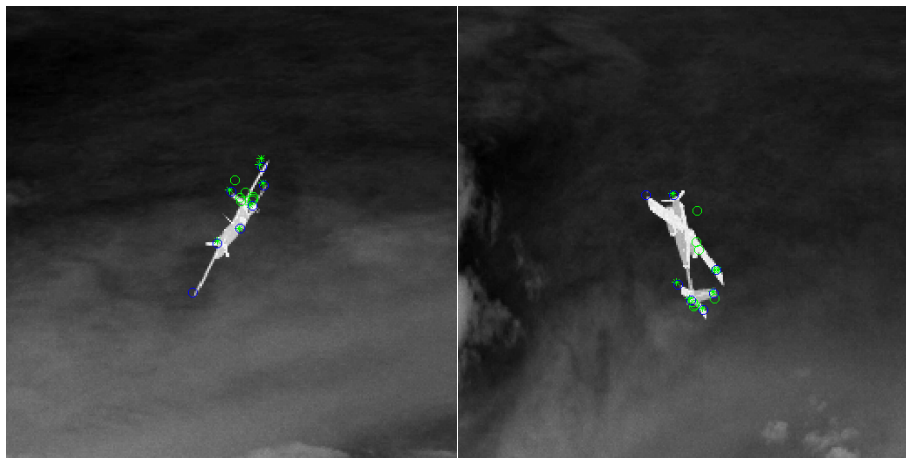


Figure 39. Poor Pose Estimation of Heatmap Keypoints.

The pose estimation results of the heatmap keypoints are of expected performance. There are instances where the pose estimation is completely accurate, Figure 38, and instances where it is severely off, Figure 39. The primary concern of the heatmap keypoints were their limitation to certain pose and bearing ambiguity. Discussed within section 1.B, the heatmap predictions had higher false positives of other features due to this limitation. It is expected that the heatmaps would have a higher penalty for pose performance as a result.

*c. Evaluation of NOAT Model*

The pose estimation of the NOAT keypoints resulted in expected behavior. The accuracy of the predicted pose was determined from calculating the RMSE. As discussed in Chapter II, RMSE is the error percentage between a ground-truth and prediction [31]. For pose estimation, the RMSE was calculated by first finding the error between the predicted 3D pose values and their ground-truth values. The pose values are Euler angles and in the units of radians. RMSE is presented as a percentage of error. Figure 40 shows the plotted RMSE for the pose estimation of both keypoint methods. Table 5 shows the summary average RMSE for each case.

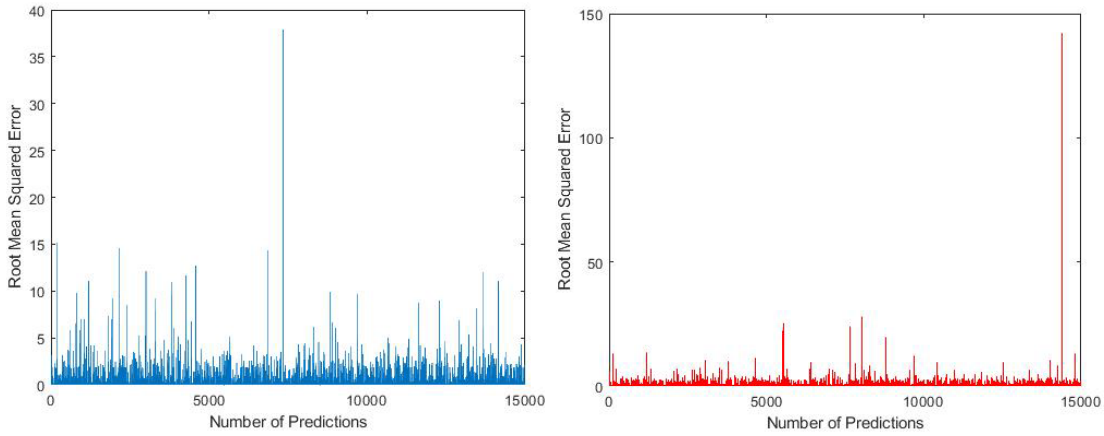


Figure 40. NOAT Pose RMSE Results. (Left: Default Keypoint. Right: Heatmap Keypoints).

Table 5. Mean RMSE of NOAT Pose Results

<b>NOAT Model Case</b>	<b>Mean RMSE</b>
Default Keypoints	0.2133
Heatmap Keypoints	0.2005

The blue plot on the left of Figure 40 is the calculated RMSE for the default keypoint pose. The red plot of the right is the calculated RMSE for the heatmap keypoint pose. The RMSE is plotted as a line graph to illustrate error noise of the pose. The RMSE is on the y-axis of both plots and ranges from the smallest error value to the largest. As seen from Table 5, the heatmap keypoints performed better than the default keypoints. The noise of the heatmap RMSE seen in Figure 40 is focused in a smaller range of error. The default keypoints had an average error of 0.2133%. The heatmap keypoints has an average error of 0.2005%. There is also a decreased in error spiking for the heatmap RMSE. There is only one spike that results in significant inaccuracy. This is an unexpected result due to the heatmap keypoints challenges with certain pose perspectives. It can be inferred that heatmaps have the potential to be more accurate and efficient for specific military applications. DL implementation of heatmaps for target detection and pose estimation could significantly improve the accuracy of military weapon systems. The NOAT model had excellent accuracy in detecting the UAV model and determining keypoints. With the utilization of a pose estimation method fit for the environment of the intended system, this model proves that there are advantageous of DL implementation for pose determination.

## **2. Dataset 2**

### ***a. Default Keypoints***

The main limitations seen from the AT model are due to the simulated environment applied to its dataset. The model struggled to learn to its full potential with several scenarios applied to it. Figure 41 shows pose predictions for its default keypoints.



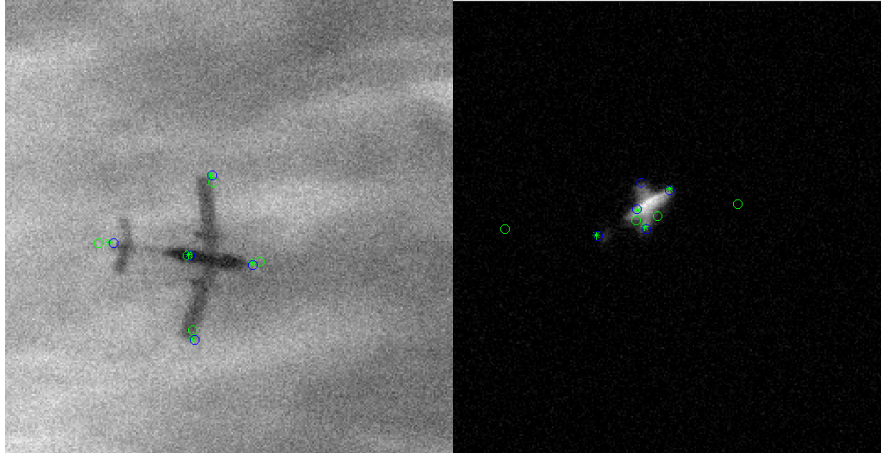


Figure 41. AT Pose Estimation from Default Keypoints.

Figure 41 displays expected results. The green open-circle predicted pose is accurate to the blue ground-truths where it is easier for the model to distinguish the target against the background. This is seen on the left image. There are other predictions where the predicted pose is off due to inaccurate keypoints, as seen on the right image. In the right image, the target has blurring around the edge of its silhouette. The AT model has the tendency of cropping its bounding boxes toward the center of the UAV targets where its pixels are more prominent. This cropping limits the accuracy this model can achieve for pose estimation.

***b. Heatmap Keypoints***

The heatmap keypoints results of the AT model did not show significant improvement from the default keypoints. Figure 42 shows the pose estimation results from the heatmap keypoints.

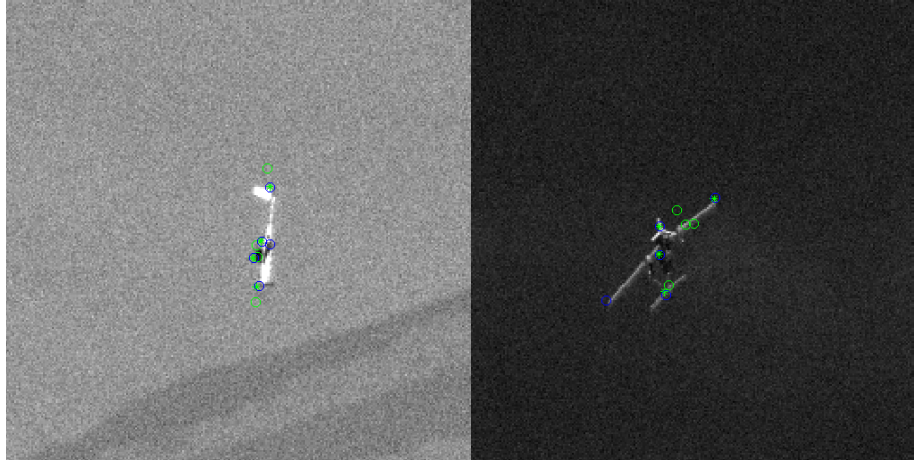


Figure 42. AT Pose Estimation from Heatmap Keypoints.

The heatmap keypoints pose performance is similar to the AT default keypoints. There does seem to be a slight improvement how inaccurate the pose can be predicted, seen in the right image. The pose is closer towards the target than being spread across the image. The NOAT model showed that heatmaps performed better than the keypoints in terms of minimizing pose error. The heatmaps of the AT model may have the same benefit.

### *c. Evaluation of AT Model*

The RMSE calculated from the AT predictions is presented in Figure 42. The blue plot on the left is the RMSE of the pose from the default keypoints. The red plot on the right is the RMSE of the pose from the heatmap keypoints. Table 6 presents the average RMSE for both keypoint cases of this model.

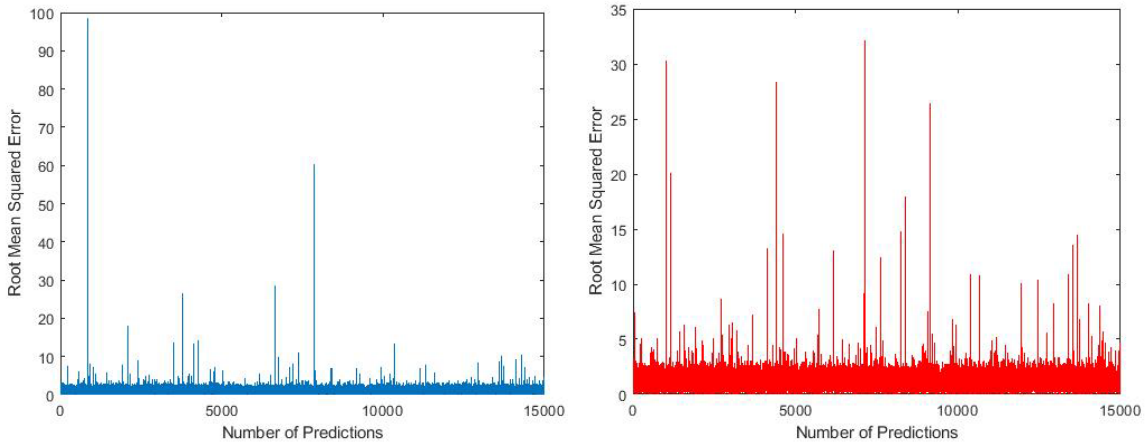


Figure 43. AT Pose RMSE Results.

Table 6. AT Pose Mean RMSE

AT Model Case	Mean RMSE
Default Keypoints	1.3901
Heatmap Keypoints	1.2805

The RMSE results showed that the heatmap predictions performed better for pose than the default keypoints for the AT model as well. The heatmap keypoints RMSE have an average error of 1.2805%. There is some spiking in the 30% error range which can be seen in the image of the right, in Figure 43. However, for the default keypoints, the average RMSE is 1.3901%. There is spiking that ranges from 30–60% error. The benefit of utilizing heatmap keypoints over the default keypoints is consistent over the two models. This further supports the usefulness of DL heatmap implementation for pose estimation in military tracking and guidance systems. The AT model was able to achieve an average error of 3% regardless of the severity of the simulated effects. This DL model shows its capability of being implemented in a real system to have similar results.

### C. PHASE 3 AIMPOINT SELECTION

The second military application that evaluated the potential of DL implementation was aimpoint selection. Aimpoint selection was achieved through the pose projection results of the predicted keypoints. The intended aimpoint was selected prior to evaluation, seen in Table 3. RMSE was the evaluation metric for aimpoint accuracy. This section reviews the results and limitations of DL implementation for this military application.

#### 1. Dataset 1

##### a. *Default Keypoints*

Since pose estimation was dependent on the pose estimation predictions, the results were expected to be consistent with the behavior seen in section B. Figure 44 shows the aimpoint results of the NOAT default keypoints.



Figure 44. NOAT Aimpoint Selection from Default Keypoints.

In Figure 44, the blue asterisk marker is the ground-truth aimpoint and the green asterisk marker is the predicted keypoint. These results show that there is secondary adverse effects to poor performance of pose. The image of the right shows that the NOAT model completely missed the target. If the pose predicted is inaccurate then there is the expectation the aimpoint will be predicted incorrectly as well.

***b. Heatmap Keypoints***

Figure 45 presents the aimpoint results from the NOAT heatmap keypoints predictions. There is similar behavior exhibited from the heatmap keypoints from what is shown above in Figure 44.



Figure 45. NOAT Aimpoint Selection from Heatmap Keypoints.

The heatmap aimpoint had varying results of accuracy to the ground-truth. In the image on the left, the aimpoint is projected behind the desired aimpoint location. As seen within this discussion, the heatmap keypoints have a limitation due to certain bearing ambiguity. In the right image, the tail of the UAV is not visible and the aimpoint prediction is incorrect as a result.

***c. Evaluation of NOAT Model***

Aimpoint selection was evaluated by the same metric used for pose estimation. The error was calculated between the predicted 2D pixel coordinates of the aimpoint and its ground-truth pixel coordinates. Figure 46 presents the RMSE results for the aimpoint determined by the NOAT model. Table 7 summaries the average error for both keypoint cases. The blue plot is the RMSE for the aimpoints predicted from default keypoints. The red plot is the RMSE for the heatmap keypoints.

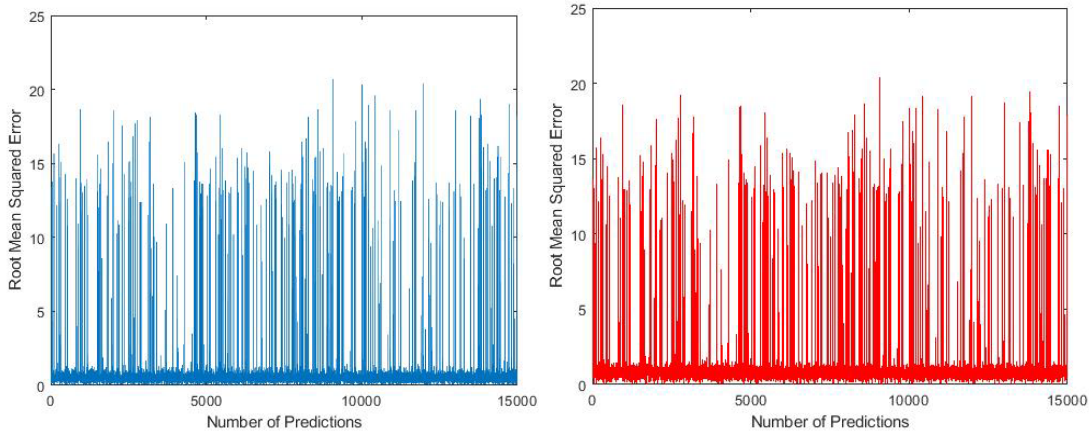


Figure 46. NOAT Aimpoint RMSE Results.

Table 7. NOAT Aimpoint Mean RMSE

<b>NOAT Model Case</b>	<b>Mean RMSE</b>
Default Keypoints	0.7401
Heatmap Keypoints	0.9894

As seen in Figure 46, there is not a significant difference between the RMSE of the two keypoint determination methods. From Table 7, the heatmap RMSE average is slightly higher than the default keypoint’s average. However, the scale of the plots and the spread of spiking is consistent between the two cases. The RMSE of the aimpoint does not show a certain advantage for the use of either keypoint method. This does show that the NOAT model is consistent within this application. However, the aimpoint RMSE is much larger than the pose estimation RMSE. The volume of spiking has increased and thus increased the range of error a system would see if this model were to be implemented. For aimpoint selection, a small error margin is heavily desirable. As seen in figures 44 and 45, poor pose can lead to a weapon system missing the target when using aimpoint selection and targeting. The results from the NOAT model provided reasonable results for aimpoint selection. The average error was below 1% for 15,000 samples. The large volume of spiking is the limitation of this application. The smallest deviation from the ground-truth

in pose can cause large error in aimpoint selection which would be detrimental for a weapon system in this application.

## 2. Dataset 2

### a. *Default Keypoints*

The limitations of the AT Model reflected in higher error for its use in pose estimation. Aimpoint selection is a direct product of pose. Figure 47 shows the aimpoint selection from the AT default keypoints. The varying results are consistent in the model's performance throughout the discussion.



Figure 47. AT Aimpoint Selection from Default Keypoints.

The images in Figure 47 show the limitations of the AT model in its use for military applications. Depending on the accuracy of the default keypoint prediction, the model has shown promising or poor performance. The green predicted marker in the first image is ahead of the desired aimpoint. The body of the UAV is the most distinguishable feature in the image. Bounding box cropping and overcompensating seen from the model can produce aimpoints that are significant off from the intended location.



***b. Heatmap Keypoints***

The heatmap keypoints of the AT model showed consistent performance to its default keypoints. There is the expectation that this consistency should be seen within its utilization for aimpoint prediction. Figure 48 shows the aimpoint results for the heatmap keypoint.

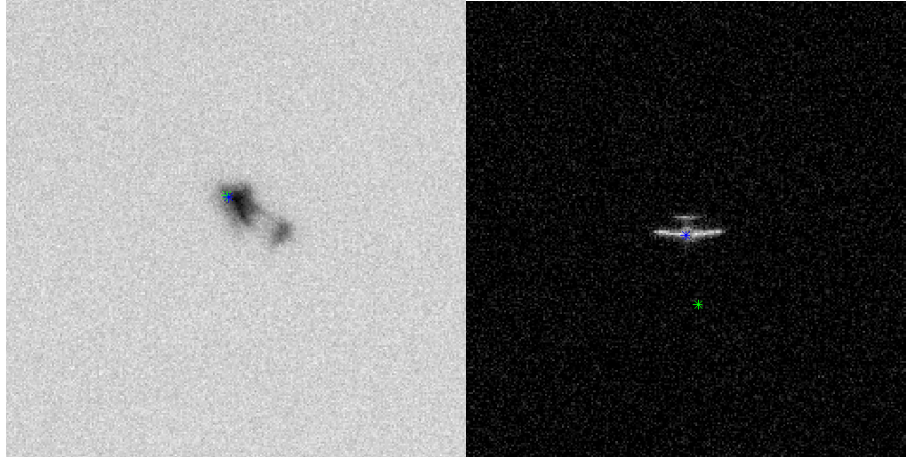


Figure 48. AT Aimpoint Selection from Heatmap Keypoints.

The aimpoint determined from the heatmap keypoints show expected results. There are cases where the accuracy is high due to previous phases performing well. Cases of low accuracy show the effects of the simulated scenarios on the AT Model. The image on the right has a large disparity between the predicted and ground-truth aimpoint. The UAV in that model is small, has blurring and is within a dark environment. The performance of the AT model has shown that its performance is dependent on the combination of these affects and their strength levels.

***c. Evaluation AT Model***

The pose estimation of the AT model showed an improvement in accurate for the heatmap keypoints. The expectation is that is improvement should be seen within aimpoint selection. The RMSE of the aimpoint predictions of the AT model can be seen in Figure 49. Table 8 presents the average RMSE values for both cases of the model.



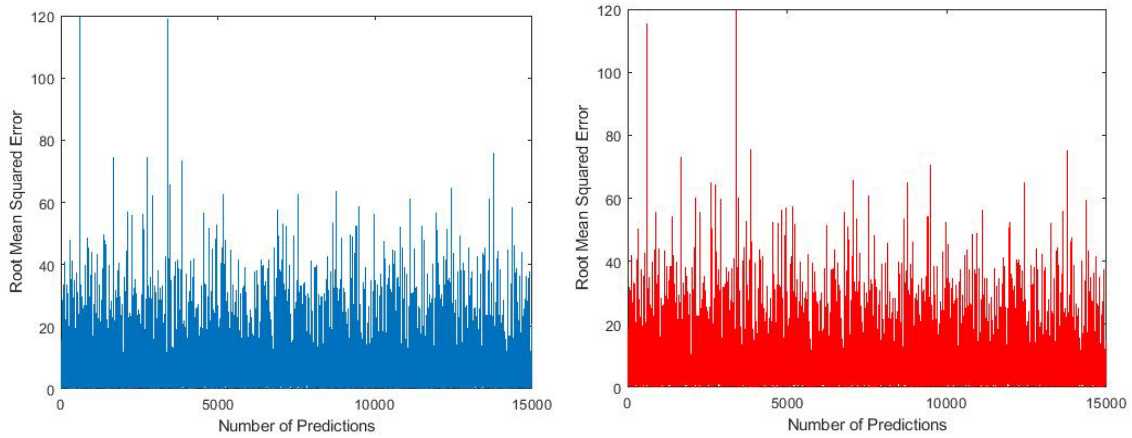


Figure 49. AT Aimpoint RMSE Results.

Table 8. AT Aimpoint Mean RMSE

AT Model Case	Mean RMSE
Default Keypoints	6.6785
Heatmap Keypoints	6.6412

The aimpoint RMSE of the AT Model has a significantly higher margin of error than the NOAT model. The average error for both its keypoint method is approximately 6.6%. Between the two methods, the aimpoint error is consistent across the model. The limitations seen from the model are heavily shown within this military application. Aimpoint selection does not allow for a large area of estimation. Poorly predicted pose has severe effects on the aimpoint performance. The DL implementation of this model would not currently benefit systems in a real-world environment. While the dataset implements scenarios seen in real data, the error presented is concerning. Another factor that may add to this limitation is the WPDLT. If a pose determination method more advantageous for real data was used for the NOAT model, the results of its pose and aimpoint determination may improve. There are limitations of the scope in the WPDLT function that are only beneficial for data like the NOAT dataset.

## V. CONCLUSION

### A. SUMMARY

This research studied the implementation of DL as a proof of concept for improving the accuracy of weapon guidance systems. This was accomplished by training a DL algorithm on two datasets that represented real-world scenarios of aerial targets. The models that were a result of that training were then evaluated on their performance and accuracy for target keypoint detection, pose estimation and aimpoint selection. Results from the NOAT model had an overall accuracy of 97% for target detection. Pose estimation and aimpoint selection had a 97–99% accuracy. Results from the AT model had lower accuracy in all areas of evaluation. The target and keypoint detection for the AT model was approximately 65% which decreased accuracy for pose estimation and aimpoint selection as well. The decrease in performance was due to the several effects the data induced to simulate a real-world scenario. Weather, lighting, and target blurring have adverse effects of the efficiency of a DL algorithm made for keypoint detection. However, experimentation also showed that heatmaps provide an alternative approach of deriving keypoints to improve performance. Heatmaps showed the most benefit for pose estimation. Poor pose estimations results were limited due to only one mathematical approach being utilized. The results of the experiment demonstrated that DL implementation has potential and can be used to improve the accuracy of weapon guidance systems. Implementation should focus on a pose estimation function fit for the applied system.

### B. FUTURE RESEARCH

Future opportunities will focus on expanding on the limitations seen in pose estimation and aimpoint selection. Evaluating other pose estimation methods for the AT model could show an ideal approach to use in conjunction with keypoint detection. Expansion of DL training could be conducted by focusing on different combinations of simulated environments for the datasets. This would provide information on which effect: weather, lighting, blurring, has the most influence of the accuracy of the DL model.

THIS PAGE INTENTIONALLY LEFT BLANK

## LIST OF REFERENCES

- [1] Dey, A., 2016, “Machine Learning Algorithms: A Review,” *International Journal of Computer Science and Information Technologies*, 3, pp. 1174–1179.
- [2] Fradkov, A., 2020, ‘Early History of Machine Learning,” *IFAC-PapersOnLine*, 53, pp.1385-1390. <https://doi.org/10.1016/j.ifacol.2020.12.1888>
- [3] Udousoro, I., 2020, “Machine Learning: A Review,” *Semiconductor Science and Information Devices*, 2. <http://dx.doi.org/10.30564/ssid.v2i2.1931>
- [4] Kim, P., 2017, *MATLAB Deep Learning: With Machine Learning, Neural Networks and Artificial Intelligence*, Apress, DE.
- [5] Aloysius, N., and Geetha, M., 2017, “A Review on Deep Convolutional Neural Networks,” *IEEE International Conference on Communication and Signal Processing*, pp. 588–592.
- [6] Li, Z., Yang, W., Peng, S. and Liu, F., 2021, “A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects,” *IEEE Transactions on Neural Networks and Learning Systems*. [doi: 10.1109/TNNLS.2021.3084827](https://doi.org/10.1109/TNNLS.2021.3084827)
- [7] Girshick, R., Donahue, J., Darrell, T., and Malik, J., 2014, “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation,” *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587. [doi: 10.1109/CVPR.2014.81](https://doi.org/10.1109/CVPR.2014.81)
- [8] He, K., Gkioxari, G., Dollár, P., and Girshick, R., 2017, “Mask R-CNN,” *IEEE International Conference on Computer Vision*, pp. 2980–2988. [doi: 10.1109/ICCV.2017.322](https://doi.org/10.1109/ICCV.2017.322)
- [9] Patil, C., and Vikas, G., 2021, “Human Pose Estimation Using Keypoint RCNN in Pytorch,” *The OpenCV Library, Dr. Dobb’s Journal of Software Tools*. [Human Pose Estimation using Keypoint RCNN in PyTorch \(learnopencv.com\)](https://learnopencv.com/human-pose-estimation-using-keypoint-rcnn-in-pytorch/)
- [10] Lin, TY. et al., 2014, “Microsoft COCO: Common Objects in Context,” *European Conference on Computer Vision*, 8693, pp. 740–755. [https://doi.org/10.1007/978-3-319-10602-1\\_48](https://doi.org/10.1007/978-3-319-10602-1_48)
- [11] Odemakinde, E., 2021, *Human Pose Estimation with Deep Learning*, VISO AI, Switzerland.
- [12] Pavlakos, G., Zhou, X., Chan, A., Derpanis, K., and Daniilidis. K., 2017, “6-DOF Object Pose from Semantic Keypoints,” *2017 IEEE International Conference on Robotics and Automation*, pp. 2011–2018. [doi: 10.1109/ICRA.2017.7989233](https://doi.org/10.1109/ICRA.2017.7989233)



- [25] Claesen, M., and Moor, B., 2015, “Hyperparameters Search in Machine Learning,” *The XI Metaheuristics International Conference*. <https://doi.org/10.48550/arXiv.1502.02127>
- [26] Pazke, A. et al., 2019, Torch.Optim. [torch.optim — PyTorch 1.11.0 documentation](https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html).
- [27] Smith, S., and Le, Q., 2018, “A Bayesian Perspective on Generalization and Stochastic Gradient Descent,” *ICLR*.
- [28] Shi, B., 2021, “On the Hyperparameters in Stochastic Gradient Descent with Momentum,” White Paper, Cornell University.
- [29] Juba, B., Le, H. , 2019, “Precision-Recall versus Accuracy and the Role of Large Data Sets,” *Proceedings of the AAAI Conference on Artificial Intelligence*, 33, pp. 4039–4048. <https://doi.org/10.1609/aaai.v33i01.33014039>
- [30] Botchkarev, A., 2019, “A New Typology Design of Performance Metrics to Measure Errors in Machine Learning Regression Algorithms,” *Interdisciplinary Journal of Information, Knowledge, and Management*, 14, 45–76. [doi:https://doi.org/10.28945/4184](https://doi.org/10.28945/4184)
- [31] Chai, T., and Draxler, R., 2014, “Root Mean Square Error (RSME) or Mean Absolute Error (MAE),” *Geoscientific Model Development*, 7, pp. 1247–1250.
- [32] Jean, N., Burke, M., Xie, M., Davis, W. M., Lobell, D. B., Ermon, S., 2016, “Combining Satellite Imagery and Machine Learning To Predict Poverty,” *Science*, 353(6301): 790–794.
- [33] Pazke, A. et al., 2019. “Torchvision Object Detection Finetuning Tutorial.” [https://pytorch.org/tutorials/intermediate/torchvision\\_tutorial.html](https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html)
- [34] Lu, P., 2020, *Self-Supervised Keypoint Learning: A Review, Towards Data Science*. Medium, CA.

THIS PAGE INTENTIONALLY LEFT BLANK

## INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center  
Ft. Belvoir, Virginia
2. Dudley Knox Library  
Naval Postgraduate School  
Monterey, California