

A TRIDENT SCHOLAR PROJECT REPORT

NO. 503

Photorealistic Image Generation for Satellite Pose Estimation Using Generative Adversarial Networks

by

Midshipman 1/C Alec J. Engl, USN



UNITED STATES NAVAL ACADEMY
ANNAPOLIS, MARYLAND

This document has been approved for public
release and sale; its distribution is unlimited.

USNA-1531-2

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this 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 Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.					
1. REPORT DATE (DD-MM-YYYY) 7/12/21		2. REPORT TYPE		3. DATES COVERED (From - To)	
4. TITLE AND SUBTITLE Photorealistic Image Generation for Satellite Pose Estimation Using Generative Adversarial Networks				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) Engl, Alec J.				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Naval Academy Annapolis, MD 21402				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S) Trident Scholar Report no. 503 (2021)	
12. DISTRIBUTION / AVAILABILITY STATEMENT This document has been approved for public release; its distribution is UNLIMITED.					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT In autonomous satellite servicing operations, pose estimation is an integral process to guide the servicing satellite for rendezvous and capture of the satellite to be serviced. Convolutional Neural Network (CNN)-based methods show promise in satellite pose estimation. In order to train CNNs for pose estimation, sufficient quantity and quality of real training imagery that is labelled with detailed pose data are required. Such images are either unavailable or very costly to produce, often forcing augmentation using computer-generated or 'synthetic' image datasets. In order to enable CNN-based pose estimators to fulfill their robust and efficient potential, one may draw from the distribution-matching ability of the Generative Adversarial Network (GAN) to modify an existing training dataset of synthetic imagery based on the characteristics of markedly fewer real images. This research focuses on the Cycle-Consistent GAN (CycleGAN) architecture for its strength in such style transfer tasks. Both a geometrically simple 'proof-of-concept' object and the on-orbit images of a small satellite are employed for 'photorealistic' image generation using CycleGAN and training of a simple CNN pose estimator. Resulting improvement to real image pose estimation accuracy of this CNN when trained on such photorealistic imagery vice synthetic imagery provides valuable insight to future applications of the implementation of CycleGAN for such training data generation.					
15. SUBJECT TERMS Learning-based pose estimation, photorealistic image generation, image style transfer, Generative Adversarial Networks (GANs)					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES 131	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER (include area code)

U.S.N.A. --- Trident Scholar project report; no. 503 (2021)

**PHOTOREALISTIC IMAGE GENERATION FOR SATELLITE POSE ESTIMATION
USING GENERATIVE ADVERSARIAL NETWORKS**

by

Midshipman 1/C Alec J. Engl
United States Naval Academy
Annapolis, Maryland

Certification of Advisers Approval

Associate Professor Tae W. Lim
Aerospace Engineering Department

Associate Professor Gavin W. Taylor
Computer Science Department

Associate Professor Randy P. Broussard
Weapons, Robotics, and Control Engineering Department

Acceptance for the Trident Scholar Committee

Professor Maria J. Schroeder
Associate Director of Midshipman Research

USNA-1531-2

Abstract

In autonomous satellite servicing operations, pose estimation is an integral process to guide the servicing satellite for rendezvous and capture of the satellite to be serviced. Convolutional Neural Network (CNN)-based methods show promise in satellite pose estimation.

In order to train CNNs for pose estimation, sufficient quantity and quality of real training imagery that is labelled with detailed pose data are required. Such images are either unavailable or very costly to produce, often forcing augmentation using computer-generated or ‘synthetic’ image datasets. The consequences of using a synthetic training set has been shown to be detrimental to real-world application, even in cases where care has been taken to make the synthetic data more photorealistic; this is due to reliance on human perception and not the CNN’s perception when defining what is ‘photorealistic.’

In order to enable CNN-based pose estimators to fulfill their robust and efficient potential, one may draw from the distribution-matching ability of the Generative Adversarial Network (GAN) to modify an existing training dataset of synthetic imagery based on the characteristics of markedly fewer real images. This research focuses on the Cycle-Consistent GAN (CycleGAN) architecture for its strength in such style transfer tasks.

Both a geometrically simple ‘proof-of-concept’ object and the on-orbit images of a small satellite are employed for ‘photorealistic’ image generation using CycleGAN and training of a simple CNN pose estimator. Resulting improvement to real image pose estimation accuracy of this CNN when trained on such photorealistic imagery vice synthetic imagery provides valuable insight to future applications of the implementation of CycleGAN for such training data generation, and to fulfill the potential of CNN-based pose estimators, trained using monocular camera images, as initial pose acquisition devices for autonomous servicing satellites.

Keywords: *learning-based pose estimation, photorealistic image generation, image style transfer, Generative Adversarial Networks (GANs)*

Acknowledgements

My incredibly competent and patient advisors, for their patience and selflessness in pushing me to improve and for answering the questions I didn't know to ask. I don't really know how to properly thank them for all that you've done and allowed me to do that made this research possible.

Dr. David Gaylor, for introducing me to the state-of-the-art in autonomous satellite control, and for sponsoring my summer internship.

Charles Oestreich, USNA '19, for conducting research in CNN-based satellite pose estimation that would form the basis of this project.

Dakota Wenberg, USNA '19, for taking the time to engage me as a plebe, and to teach me skills that ultimately enabled me to conduct this project.

My fellow researchers (Joe Wiedemann, Logan Williams, Lillian Usadi, and Harrison Foley) and my teammates (Calvin McDonald, Devin Flagg, Eddie MacGuire, Jacob Rose, Bryce Biediger, Chris Meacham, Jacob Rose, Marcos Valdes, Laura Spratling, and the rest of the 4th Company's class of '21) for their wisdom, support, fellowship, and for knowing better than I when 'taking a break' was the best course of action.

Table of Contents

Abstract	1
Acknowledgements	2
List of Figures	5
List of Tables	11
1 Introduction and Basic Definitions	12
1.1 Pose and Pose Estimation	12
1.2 Demand for Pose Estimation in Space-Based Robotics	12
1.3 Research Introduction: Demand for Photorealistic Training Data for CNN-based Pose Estimators	13
2 Background on Machine Learning Style Transfer via GAN	15
2.1 Introduction to Deep Machine Learning via Neural Networks	15
2.1.1 Training of Neural Networks	17
2.1.2 Capabilities of Neural Networks	19
2.1.3 Convolutional Neural Networks (CNNs)	20
2.1.4 Transfer Learning	22
2.2 Generative Adversarial Networks (GANs)	23
2.2.1 Cycle-Consistent Adversarial Network (CycleGAN)	27
2.2.2 Weaknesses and Common Errors of GANs	32
3 Chosen Datasets and Preliminary Analysis	34
3.1 Proof of Concept Datasets	34
3.2 Spacecraft Pose Estimation Dataset (SPEED)	37
4 Research Method	39
4.1 CycleGAN Implementation	39
4.2 Analysis and Selection of CycleGAN Output for Photorealistic Dataset	39
4.3 Photorealistic Training of Pose Estimator CNN	40
4.3.1 Pose Labelling Method	40
4.3.2 Chosen CNN Pose Estimator, and Settings	42
4.4 Quantitative Validation	43
4.5 Application of Method to Chosen Datasets	45
4.5.1 POC and POC2	45
4.5.2 SPEED	46
4.5.3 Note on Downsampling	51
4.6 Computational Resources	52
5 CycleGAN Trials and Outputs	53
5.1 Using Unmodified (‘Vanilla’) CycleGAN	53
5.1.1 Foreword	53
5.1.2 Initial Trials with Un-Cropped, Un-Sorted SPEED	53
5.1.3 Training with Sort-Pare, Un-Cropped SPEED	55

5.1.4	Training More Rigorously with POC	69
5.1.5	Training with POC2	83
5.1.6	First Cropped SPEED Trials	90
5.1.7	Modified Approach to Cropped SPEED (Data Augmentation)	101
6	Analysis and Reduction of Results.....	113
6.1	CycleGAN Output	113
6.1.1	Hierarchy of Introduced Noise	113
6.1.2	Generalization Error.....	114
6.1.3	Regularity of Convergences	115
6.2	Training Pose Estimation CNNs	116
6.2.1	POC	117
6.2.2	POC2	117
6.2.3	SPEED.....	118
6.3	Non-Visual Methods of CycleGAN Evaluation and Implications	121
7	Future Work	124
7.1	Using Style Transfer as a Means of Augmenting Real Training Data	124
7.2	Structural Change Analysis Using Classical Feature Detection Comparison	124
7.3	Range Determination and More Sophisticated CNNs for Pose Estimation.....	124
7.4	Methods of Model-Agnostic GAN Style Transfer Evaluation.....	124
8	Final Recommendation	126
	Bibliography.....	128
	Appendix A: Training Validation Accuracies at Increasing Images per Label.....	131

List of Figures

Figure 1. Visualization of rigid body rotation via Euler Angles with the example of a CubeSat [1]..	12
Figure 2. Significant error difference in real and synthetic data trials in synthetically trained networks, demonstrated in the ESA/Stanford pose estimation challenge (network placing indicated)	14
Figure 3. Generalized depiction of a node with three inputs [6].....	15
Figure 4. Comparison of two commonly used activation functions: (a) Sigmoid; (b) ReLU [7]	16
Figure 5. Generic depiction feed-forward neural network [6]	16
Figure 6. Gradient descent illustration on mean-squared loss function (J) for one node weight (w).	17
Figure 7. Arbitrary two-layer neural network with labeled connection weights	18
Figure 8. Depiction of one step in the numerical gradient-based learning process on two dimensions of the loss function (where C is the loss function) [8]	19
Figure 9. A moderate-fidelity classification curve [6].....	20
Figure 10. Simplified CNN architecture [6]	20
Figure 11. Example of the first two convolutions of a 4×4 matrix with a kernel (2×2 matrix in center) [6]	21
Figure 12. Several examples of the convolution operation [6]	22
Figure 13. Depiction of the two main variations of pooling [6].....	22
Figure 14. Generalized GAN architecture with entirely random input into the Generator [15]	23
Figure 15. Illustrated GAN analogy [19]	25
Figure 16. Detailing an ideal case of the GAN's distribution modelling, where the dotted black line represents <i>p_{data}</i> , the green line represents <i>p_g</i> (the probability distribution of Generator output), and the blue line represents the governing loss function; note that the vectors drawn between the bottom two horizontal lines represent how the input noise (z) is mapped to match real data samples (x). [18]	25
Figure 17. The Conditional Generative Adversarial Network (cGAN) architecture [20]	26
Figure 18. Paired and unpaired image datasets [22]	26
Figure 19. Unpaired horse-to-zebra translation via CycleGAN [22]	27
Figure 20. Block diagram of CycleGAN architecture.....	28
Figure 21. CycleGAN Generator Architecture [23]	29
Figure 22. CycleGAN Discriminator Architecture [24]	30
Figure 23. Illustration of concept of cycle-consistency loss and how it uses both Generators' mapping [22]	32
Figure 24. Illustration of identity loss as it applies to each Generator's mapping [22]	32
Figure 25. Mode collapse for a GAN trained to produce hand-drawn eights [25]	33
Figure 26. Examples of images from synthetic and real sets of the 'Block Model' (note that the real images will have a black background)	34
Figure 27. Examples of images from synthetic and real sets of the POC Dataset	35
Figure 28. Examples of images from synthetic and real sets of the POC2 Dataset.....	36

Figure 29. Examples of differences in lighting between images from synthetic and real sets of the POC2 Dataset.....	37
Figure 30. Examples of images from the synthetic and real sets from SPEED [27]	38
Figure 31. Projecting onto orthogonal coordinates to visualize all possible 3-D orientations within a range	40
Figure 32. Example of the discretization scheme used to assign a range of poses (bounded in yellow) to a specific pose label (blue point).....	41
Figure 33. Visualization of how labels were selected on a specific axis over a range, where the green dot is the starting point label, the blue dots are labels, and the orange dots are the label bounds; note that lower bounds are exclusive (open circle) and upper bounds are inclusive (solid circle)	41
Figure 34. Discretization scheme used above, labelled symbolically.....	42
Figure 35. Example of a pose label configuration and its associated bounds over the range $[0^\circ, 360^\circ)$	42
Figure 36. Base AlexNet CNN Structure [30]	43
Figure 37. Validation via comparison of pose estimates from synthetically-trained CNN (left) and photorealistically-trained CNN (right)	44
Figure 38. POC/POC2 testing pipeline.....	45
Figure 39. SPEED testing pipeline	47
Figure 40. Lack of Earth background representation in real SPEED imagery [27].....	47
Figure 41. Examples of images from the synthetic and real sets from SPEED, cropped with grabCut() [27]	48
Figure 42. Cropped real SPEED imagery; note that, though there are slight differences, there are large groups of images that share very similar orientations (i.e. few modes of pose)	49
Figure 43. Illustration of geometry-preserving data augmentation [41]	49
Figure 44. Example of a 1200x1200 SPEED image bilinearly downsampled to 256x256.....	52
Figure 45. Five examples of S2R, visually poor impact of training with both space and Earth backgrounds, CycleGAN trained on SPEED for 37 epochs.....	54
Figure 46. Five examples of S2R, visually poor impact of training with unbalanced quantities of real and synthetic imagery, CycleGAN trained on Sorted SPEED for 38 epochs	55
Figure 47. Five examples of S2R, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 1 epoch	56
Figure 48. Five examples of R2S, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 1 epoch	56
Figure 49. Five examples of S2R, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 6 epochs.....	57
Figure 50. Five examples of R2S, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 6 epochs.....	57
Figure 51. Five examples of S2R, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 8 epochs.....	58
Figure 52. Five examples of R2S, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 8 epochs.....	58
Figure 53. Five examples of S2R, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 12 epochs	59

Figure 54. Five examples of R2S, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 12 epochs	59
Figure 55. Five examples of S2R, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 15 epochs	60
Figure 56. Five examples of R2S, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 15 epochs	60
Figure 57. Five examples of S2R, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 23 epochs	61
Figure 58. Five examples of R2S, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 23 epochs	61
Figure 59. Five examples of S2R, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 45 epochs	62
Figure 60. Five examples of R2S, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 45 epochs	62
Figure 61. Five examples of S2R, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 46 epochs	63
Figure 62. Five examples of R2S, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 46 epochs	63
Figure 63. Five examples of S2R, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 71 epochs	64
Figure 64. Five examples of R2S, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 71 epochs	64
Figure 65. Five examples of S2R, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 74 epochs	65
Figure 66. Five examples of R2S, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 74 epochs	65
Figure 67. Five examples of S2R, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 76 epochs	66
Figure 68. Five examples of R2S, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 76 epochs	66
Figure 69. Five examples of S2R, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 84 epochs	67
Figure 70. Five examples of R2S, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 84 epochs	67
Figure 71. Five examples of S2R, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 100 epochs	68
Figure 72. Five examples of R2S, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 100 epochs	69
Figure 73. Five examples of S2R, CycleGAN trained on POC for 1 epoch	70
Figure 74. Five examples of R2S, CycleGAN trained on POC for 1 epochs	70
Figure 75. Five examples of S2R, CycleGAN trained on POC for 2 epochs	71
Figure 76. Five examples of R2S, CycleGAN trained on POC for 2 epochs	71
Figure 77. Five examples of S2R, CycleGAN trained on POC for 4 epochs	72

Figure 78. Five examples of R2S, CycleGAN trained on POC for 4 epochs	73
Figure 79. Five examples of S2R, CycleGAN trained on POC for 7 epochs	73
Figure 80. Five examples of R2S, CycleGAN trained on POC for 7 epochs	74
Figure 81. Five examples of S2R, CycleGAN trained on POC for 9 epochs	74
Figure 82. Five examples of R2S, CycleGAN trained on POC for 9 epochs	75
Figure 83. Five examples of S2R, CycleGAN trained on POC for 10 epochs	75
Figure 84. Five examples of R2S, CycleGAN trained on POC for 10 epochs	76
Figure 85. Five examples of S2R, CycleGAN trained on POC for 12 epochs	77
Figure 86. Five examples of R2S, CycleGAN trained on POC for 12 epochs	77
Figure 87. Five examples of S2R, CycleGAN trained on POC for 14 epochs	78
Figure 88. Five examples of R2S, CycleGAN trained on POC for 14 epochs	78
Figure 89. Five examples of S2R, CycleGAN trained on POC for 24 epochs	79
Figure 90. Five examples of R2S, CycleGAN trained on POC for 24 epochs	79
Figure 91. Five examples of S2R, CycleGAN trained on POC for 41 epochs	80
Figure 92. Five examples of R2S, CycleGAN trained on POC for 41 epochs	80
Figure 93. Five examples of S2R, CycleGAN trained on POC for 45 epochs	81
Figure 94. Five examples of R2S, CycleGAN trained on POC for 45 epochs	81
Figure 95. Five examples of S2R, CycleGAN trained on POC for 52 epochs	82
Figure 96. Five examples of R2S, CycleGAN trained on POC for 52 epochs	82
Figure 97. Five examples of S2R, CycleGAN trained on POC for 67 epochs	83
Figure 98. Five examples of R2S, CycleGAN trained on POC for 67 epochs	83
Figure 99. Five examples of S2R, CycleGAN trained on POC2 for 1 epoch.....	84
Figure 100. Five examples of S2R, CycleGAN trained on POC2 for 8 epochs	84
Figure 101. Five examples of S2R, CycleGAN trained on POC2 for 11 epochs	85
Figure 102. Five examples of S2R, CycleGAN trained on POC2 for 13 epochs	85
Figure 103. Five examples of S2R, CycleGAN trained on POC2 for 28 epochs	86
Figure 104. Five examples of S2R, CycleGAN trained on POC2 for 29 epochs	86
Figure 105. Five examples of S2R, CycleGAN trained on POC2 for 49 epochs	87
Figure 106. Five examples of S2R, CycleGAN trained on POC2 for 57 epochs	87
Figure 107. Five examples of S2R, CycleGAN trained on POC2 for 78 epochs	88
Figure 108. Five examples of R2S, CycleGAN trained on POC2 for 1 epochs	88
Figure 109. Five examples of R2S, CycleGAN trained on POC2 for 6 epochs	89
Figure 110. Five examples of R2S, CycleGAN trained on POC2 for 9 epochs	89
Figure 111. Five examples of R2S, CycleGAN trained on POC2 for 13 epochs	90
Figure 112. Five examples of R2S, CycleGAN trained on POC2 for 19 epochs	90
Figure 113. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED for 1 epoch .	91
Figure 114. Five examples of R2S, CycleGAN trained on cropped sort-pare SPEED for 1 epoch .	91
Figure 115. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED for 6 epochs	92
Figure 116. Five examples of R2S, CycleGAN trained on cropped sort-pare SPEED for 6 epochs	92
Figure 117. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED for 12 epochs	93

Figure 118. Five examples of R2S, CycleGAN trained on cropped sort-pare SPEED for 12 epochs	93
Figure 119. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED for 15 epochs	94
Figure 120. Five examples of R2S, CycleGAN trained on cropped sort-pare SPEED for 15 epochs	94
Figure 121. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED for 22 epochs	95
Figure 122. Five examples of R2S, CycleGAN trained on cropped sort-pare SPEED for 22 epochs	95
Figure 123. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED for 24 epochs	96
Figure 124. Five examples of R2S, CycleGAN trained on cropped sort-pare SPEED for 24 epochs	96
Figure 125. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED for 26 epochs	97
Figure 126. Five examples of R2S, CycleGAN trained on cropped sort-pare SPEED for 26 epochs	97
Figure 127. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED for 33 epochs	98
Figure 128. Five examples of R2S, CycleGAN trained on cropped sort-pare SPEED for 33 epochs	98
Figure 129. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED for 50 epochs	99
Figure 130. Five examples of R2S, CycleGAN trained on cropped sort-pare SPEED for 50 epochs	99
Figure 131. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED for 57 epochs	100
Figure 132. Five examples of R2S, CycleGAN trained on cropped sort-pare SPEED for 57 epochs	100
Figure 133. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED for 58 epochs	101
Figure 134. Five examples of R2S, CycleGAN trained on cropped sort-pare SPEED for 58 epochs	101
Figure 135. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 1 epoch	103
Figure 136. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 2 epochs	103
Figure 137. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 5 epochs	103
Figure 138. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 8 epochs	104
Figure 139. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 14 epochs	104

Figure 140. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 18 epochs	105
Figure 141. Five examples of R2S, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 18 epochs	105
Figure 142. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 23 epochs	106
Figure 143. Five examples of R2S, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 23 epochs	106
Figure 144. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 34 epochs	107
Figure 145. Five examples of R2S, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 34 epochs	107
Figure 146. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 36 epochs	108
Figure 147. Five examples of R2S, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 36 epochs	108
Figure 148. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 47 epochs	109
Figure 149. Five examples of R2S, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 47 epochs	109
Figure 150. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 50 epochs	110
Figure 151. Five examples of R2S, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 50 epochs	110
Figure 152. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 52 epochs	111
Figure 153. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 76 epochs	111
Figure 154. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 82 epochs	112
Figure 155. Selected favorable results from CycleGAN synthetic-to-real Generator, trained on uncropped Sort-Pare SPEED for 100 epochs	114
Figure 156. Selected unfavorable results from CycleGAN synthetic-to-real Generator, trained on uncropped Sort-Pare SPEED for 100 epochs	115
Figure 157. Lack of distant-satellite image representation in real SPEED imagery	115
Figure 158. Pose classification accuracies for synthetically- and photorealistically- trained CNNs on POC dataset	117
Figure 159. Pose classification accuracies for synthetically- and photorealistically- trained CNNs on POC2 dataset	118
Figure 160. Failure to converge in CNN pose estimator training	119
Figure 161. Properly-trained, converged CNN pose estimator	119
Figure 162. Pose classification accuracies for synthetically- and photorealistically- trained CNNs on SPEED; pose label configuration 5 as per Table 3	120

Figure 163. Pose classification accuracies for synthetically- and photorealistically- trained CNNs on SPEED; pose label configuration 6 as per Table 3	120
Figure 164. Visually-selected photorealistic SPEED associated with the point in CycleGAN training at which they were generated, alongside histograms for reference	122
Figure 165. General evaluation pipeline for GAN output (inspired from [38])	125

List of Tables

Table 1. POC, POC2 Dataset labelling conventions	46
Table 2. Quantitative breakdown of datasets derived from SPEED	48
Table 3. Selected SPEED Label Configurations (those used for training are highlighted in gray)....	50
Table 4. Manual Sorting and Manipulation of Real SPEED Into Pose Configuration.....	51
Table 5. Apparent convergences in analyzed CycleGAN trials.....	116

1 Introduction and Basic Definitions

1.1 Pose and Pose Estimation

Pose is defined as the position *and* orientation of an object in space at any given time. In 3D Cartesian coordinates (x, y, z) , position is described by 3 spatial coordinates (distances along the x, y and z axes) and orientation is described by 3 rotations. One method of describing orientation is via Euler angles (φ, θ, ψ) , where φ is the rotation angle about the x-axis, θ is the rotation angle about the y-axis, and ψ is the rotation angle about the z-axis (as seen in Figure 1).

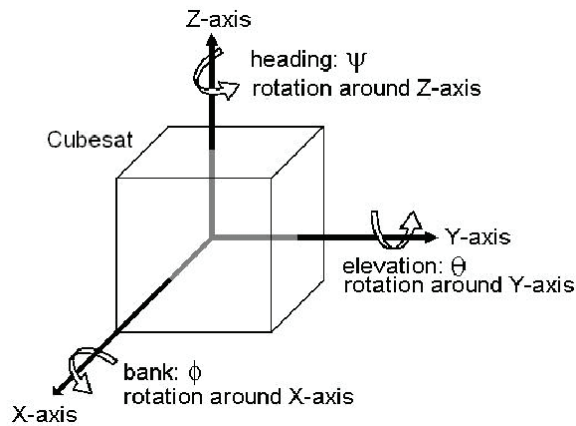


Figure 1. Visualization of rigid body rotation via Euler Angles with the example of a CubeSat [1]

For any given 3D object, pose is the information required to interact with it in space- for example, reaching for a book on your shelf, one first sees it, understand how far away that book is, and how it is oriented in order to extend your hand to pick it up. Thus, for visually-controlled robotics, an algorithm that utilizes input from a visual sensor and can perform *pose estimation* in real time is important.

1.2 Demand for Pose Estimation in Space-Based Robotics

There is currently a major sustenance issue embedded deeply in our space mission: satellites, which make up some of the most cost-dense and vital commercial and government assets, can and do fail because of a number of probable failures, especially at the beginning of their operational use. Also a consideration are those satellites that are planned or commanded into their end-of-life, i.e. running out of fuel, or being abandoned due to the conclusion of research. Either way, these are functional satellites on-orbit that are repairable or refuellable, and could provide many more years of operations. Finding a reliable ‘on-orbit servicing’ solution will not be trivial. Manned servicing missions are far too costly to stand-alone, and human-controlled robotic modules do not meet the required precision for proximity operations.

On-orbit servicing has only recently moved into the realm of Autonomous Rendezvous, Proximity Operations, and Docking (ARPOD) from human-controlled servicing missions. The Naval Research

Laboratory (NRL) has developed the Robotic Servicing of Geosynchronous Satellites (RSGS) program. [2] Likewise, NASA is currently undergoing preparations for its OSAM-1 (On-orbit Servicing, Assembly, and Manufacturing) mission which intends to rendezvous with Landsat 7 satellite in sun-synchronous orbit. [3] These designs utilize two robotic arms and a ‘golf bag’ of interchangeable tools in order to rendezvous with and provide diagnostics, anomaly corrections, and upgrade installation with full autonomy. ‘Full autonomy’ entails they would be launched into orbit and navigate to, rendezvous with, and execute a servicing mission without operator input. SpaceLogistics LLC now provides cooperative space logistics and in-orbit satellite servicing to geosynchronous satellite operators. Their proof-of-concept mission, MEV-1 (Mission Extension Vehicle) successfully docked with the geostationary Intelsat IS-901 in early 2020, augmenting the craft with its own thrusters to extend its operational life. [4]

However, despite their relative sophistication, the technology demonstrated by these missions has not reached technical maturity yet, gauged by the amount of separate research and development that goes into the customization of the controls for every ‘one-off’ satellite repair mission. In other words, the industry hasn’t settled on one strategy to apply to the entire range of satellite repair missions.

To contribute to this maturation, a potential area for improvement lies in enhancing/introducing the pose initialization stage in the navigation control algorithm using machine learning-based (Convolutional Neural Network, or CNN-based) pose estimation. This approach would process a 2D image feed and provide a real-time pose estimate (relative 3D position and orientation) of the target satellite for subsequent refinement by the main algorithm. This enables more computational efficiency and more resilience to noise due to the characteristics of the machine learning process (see 2.1.3). In addition, this could serve as a standalone pose estimation methodology for smaller robotic platforms with limited computation capability.

1.3 Research Introduction: Demand for Photorealistic Training Data for CNN-based Pose Estimators

As established above, the main weakness to the use of CNNs for pose estimation is the lack of availability of any sufficient amount of real (or real-appearing) training data that is labelled with detailed pose data. The consequences of using a synthetic training set is assuredly detrimental to real-world application, even in cases where care has been taken to make the synthetic data more photorealistic. The results of the ESA/Stanford Pose Estimation Challenge [5], held in July of 2019, demonstrate this; even amongst the top five scoring networks, the error indices obtained using real data trials were 3 to 40 orders of magnitude higher than those using the synthetic data. This can be seen in Figure 2 below.

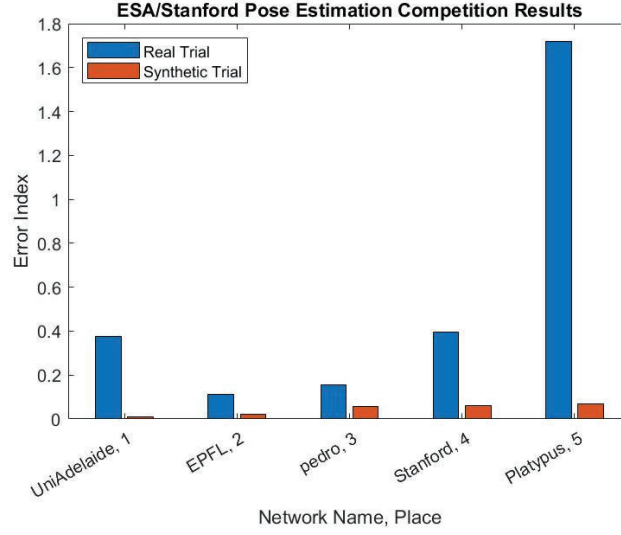


Figure 2. Significant error difference in real and synthetic data trials in synthetically trained networks, demonstrated in the ESA/Stanford pose estimation challenge (network placing indicated)

This research was intended to enable CNN-based pose estimators to fulfill their potential as more robust and efficient alternatives to existing methods (e.g. ICP). This was done by drawing from the distribution-matching ability of the GAN to modify a set of computer-generated (referred to in this paper as ‘synthetic’) imagery based on the characteristics of a set of real-world (or simply ‘real’) data. The intended result was a trained Generator network that can create a training set of sufficient size and diversity from a sparse set of real data in order to train CNNs for real-world pose estimation; this is a *model-based* approach (i.e. requiring *a priori* model of target satellite), and will not be generalized to enable pose estimation of any and all satellite bodies (this would imply a *model-free* approach). It was thus required that, for any target satellite/object, that there was a large-enough synthetic dataset for CNN training (or a means of digitally producing it) alongside a real dataset (which could be much smaller in quantity). The image datasets utilized in the photorealistic image generation process using GAN are detailed in section 3.

2 Background on Machine Learning Style Transfer via GAN

2.1 Introduction to Deep Machine Learning via Neural Networks

The architecture of neural networks is fundamentally inspired by the brain. They, like their organic counterparts, are constructed from a hierarchically interconnected lattice structure of synthetic neurons (or nodes). For illustration, a depiction of a node with three inputs is included in Figure 3.

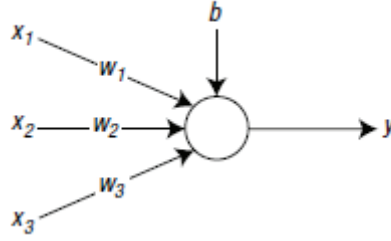


Figure 3. Generalized depiction of a node with three inputs [6]

These nodes take input signals from and provide a digested output to other nodes. Each node is an independent process, and weighs uniquely each of the inputs it receives when it generates an output. The equations for the single node in Figure 3 are included below.

$$v = [w_1 \ w_2 \ w_3] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + b = (w_1x_1 + w_2x_2 + w_3x_3) + b \quad [1]$$

$$y = \phi(v) \quad [2]$$

where b is node bias, x_i is the i^{th} input, w_i is the weight at the i^{th} input, v is the weighted sum, $\phi(\cdot)$ is the activation function, and y is the node output. [6] In other words, the node ‘weighs’ each of its inputs by multiplying each input signal x_i by the numerical weight w_i associated with it. The larger in absolute value that w_i is relative to other weights the more potent x_i becomes in the output with respect to other inputs. It then sums each of the weighted inputs and adds an internal bias, which is a value that is constant to the node. From there, the node runs the weighted total of these inputs through an internal ‘activation function,’ which outputs a single scalar. This scalar is an interpretation of how much or how little the node was ‘activated’, and is passed as the node’s final output. An activation function can be any mathematical function that ‘*shapes*’ the output of the node, but nonlinearity in the activation functions of a neural network’s nodes allows the network to interpret more complex, nonlinear phenomenon. Two common examples are the sigmoid and Rectified Linear Unit (ReLU) activation functions, defined respectively by

$$\phi(v) := \sigma(v) = \frac{1}{1 + e^{-v}} \quad [3]$$

$$\phi(v) := R(v) = \max(0, v) = \begin{cases} v, & v \geq 0 \\ 0, & v < 0 \end{cases} \quad [4]$$

where v is the weighted sum of the inputs, $\phi(\cdot)$ is the activation function, $\sigma(\cdot)$ is the sigmoid function, and $R(\cdot)$ is the ReLU function. These functions are displayed below in Figure 4 (note that z is used in place of v per the source's convention).

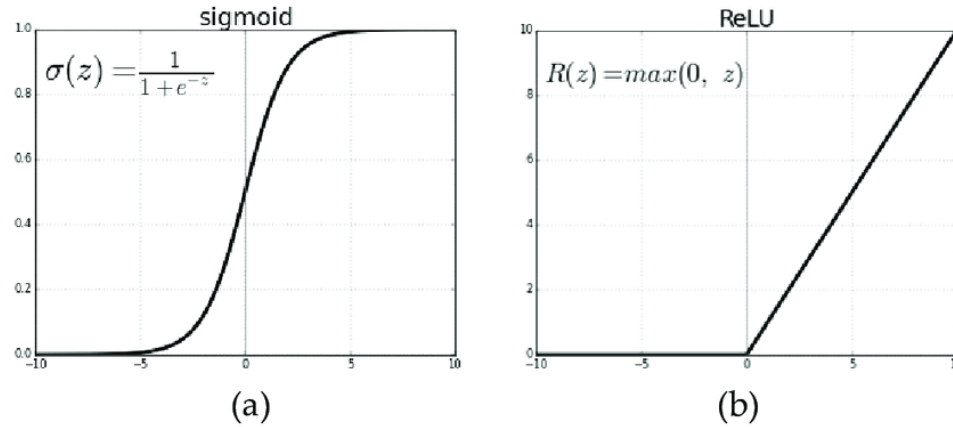


Figure 4. Comparison of two commonly used activation functions: (a) Sigmoid; (b) ReLU [7]

A basic ‘feed-forward’ neural network is formed from an arbitrary number of nodes organized into layers, where the outputs of a previous layer become the inputs to the next layer. Network layers are divided into input, output, and hidden. *Input layer* nodes simply output the input data. If an image classification network was designed to process images of resolution $n \times n$ pixels, it would have an input layer of n^2 nodes, each outputting one pixel's value. The *output layer* outputs the final results provided by the network. *Hidden layers* are those in connecting input to output, and are not accessible from the network's exterior. A simple feed-forward network architecture can be seen in Figure 5.

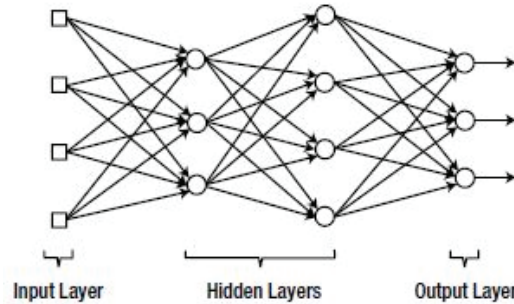


Figure 5. Generic depiction feed-forward neural network [6]

2.1.1 Training of Neural Networks

A neural network, like the brain, ‘learns’ (or improves in its functions) from experience. If one desires the network to become an accurate model, one must provide it many samples of what it would be expected to observe with corresponding correct output. For example, if one wanted a neural network that could identify pictures of cats and dogs, one would provide it with many diverse examples of both (this is known as the “training set”), all the while allowing it to make internal adjustments of weights to reinforce the correct answer (or the data’s ‘label’). Then, its classification ability will be tested with another, distinct “test set” of pictures of cats and dogs. Based on the outcome of this validation, one may determine how well-trained the network is.

The training process for typical feed-forward neural network involves supervised learning. This entails providing the network with a sufficient amount of training data, each of which has been ‘labelled’ with the correct output. A crucial part of this process is that the training dataset will be distinct from, but representative of the real input that the network will later encounter. There are many methods (or ‘learning rules’) on how to optimize training of the network with the finite amount of training data one is allotted. Most representative are Gradient Descent (GD) methods. GD is a numerical method to change the weighting of the network and approximate the optimum operating condition based on each item within the training set. A loss function is chosen which quantifies the difference between the correct labels and the output of the network. There are many forms of loss functions, but essentially they follow the same basic concept: the more difference there is between the output and the correct answer, the higher the loss function value is. Perhaps the most basic example is the ‘mean-squared error’ function, written as

$$J = \frac{1}{2M} \sum_{i=1}^M \|d_i - y_i\|^2 \quad [5]$$

where J is the loss function value, d_i is the correct value at the i^{th} output node, y_i is the actual label at the i^{th} output node, and M is the number of output nodes. A visualization of a simple GD method with the above loss function can be seen in Figure 6. [8]

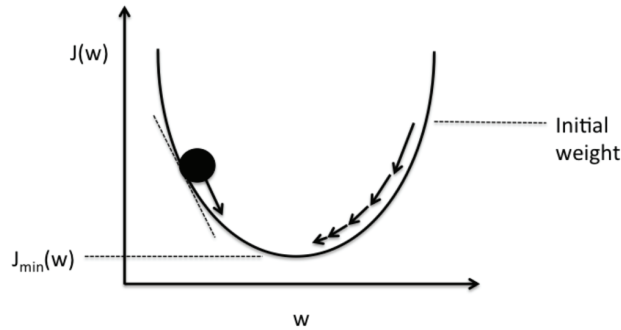


Figure 6. Gradient descent illustration on mean-squared loss function (J) for one node weight (w)

The loss can be changed by altering the weights w_i within each node of the neural network. From the rate of change of this loss function with respect to these weights, the ‘back-propagation’ algorithm will

change the weights and bias of each node in the network (starting from the output and ending at the input layer, hence ‘back’-propagating). [9] The governing equations of the back-propagation algorithm follow:

$$w := w - \alpha \frac{\partial J}{\partial w} \quad [6]$$

$$b := b - \alpha \frac{\partial J}{\partial b} \quad [7]$$

where w is a weight, α is learning rate, J is loss function, and b is bias. Consider the following network diagram in Figure 7. [10]

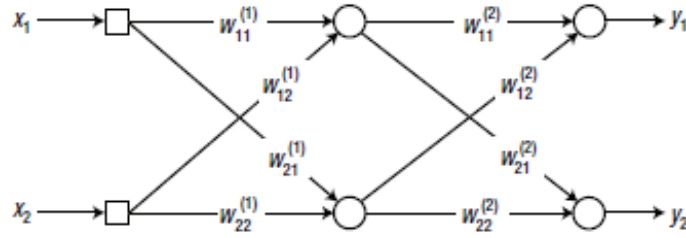


Figure 7. Arbitrary two-layer neural network with labeled connection weights

Simplifying the partial derivatives of J , and adding subscripts to indicate network locations, in accordance with Figure 7, these equations simplify to the following.

$$w_{ij}^k := w_{ij}^k - \alpha \frac{\partial J}{\partial y_i^k} \phi'(v_i^k) y_i^k \quad [8]$$

$$b_i^k := b_i^k - \alpha \frac{\partial J}{\partial y_i^k} \phi'(v_i^k) \quad [9]$$

where w_{ij}^k is the weight from the j^{th} node of the $(k-1)^{\text{th}}$ layer to the i^{th} node of the k^{th} layer, b_i^k is the bias in the i^{th} node of the k^{th} layer, y_i^k is the output from the i^{th} node of the k^{th} layer, v_i^k is the weighted sum of all inputs to the i^{th} node of the k^{th} layer, and $\phi'(\cdot)$ is the partial derivative of the activation function with respect to v_i^k . [10]

The cyclic process of propagating signals forward (Equations [4], [5]) and backward (Equations [6], [7]) through the network will generally be iterated over a large quantity of training data. The network is trained sufficiently to the data when the loss function is minimized with respect to the biases and the weights. This is pictured over a two-dimensional domain (two arbitrary outputs, v_1 and v_2), in Figure 8.

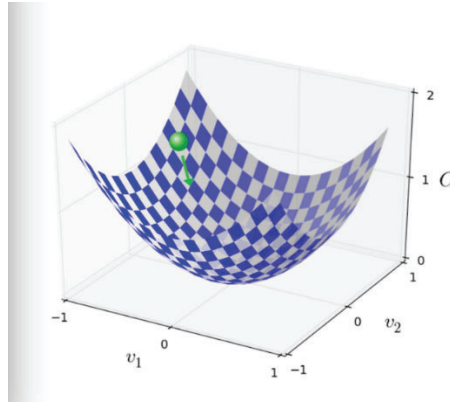


Figure 8. Depiction of one step in the numerical gradient-based learning process on two dimensions of the loss function (where C is the loss function) [8]

Part of what makes a training run stable is if loss function generally moves towards the network's goal of minimization, and eventually reaches a result of optimal training on its dataset. Once a network has functionally settled at or around a minimum of the loss function using GD (or another method of optimization), it is referred to as 'converged,' doing so predictably over the course of training is another sign that the training run was stable. Training run instability may cause heightened or less predictable variations in trained model output, and may lead to fewer, lower quality convergence results or none at all (divergence).

Monitoring training is done via preset network parameters (duration, images per training iteration, etc.) as well as during the training itself via metrics (training and validation accuracies, etc.). Duration of a training phase is typically controlled by providing the number of times in total the network will process the total dataset, or number of epochs. Training and validation accuracies are those classification accuracies (i.e. percentage of correctly sorted data into the learned class labels) computed using the trained model throughout the duration of training (usually calculated each epoch), where training accuracy considers the classification of the training dataset, and validation accuracy considers the classification of a 'test' dataset, which is typically part of the dataset not utilized for model training; note that additional controls on duration can be set based on a condition, such as maximum/minimum validation accuracy. Validation accuracy provides a check to simulate how effectively the model can generalize to data that it hasn't directly observed.

Additionally, in training one may specify the number of datapoints processed by the network before a weight update is issued (see Equations [6], [7]), or 'batch size.' Each of such updates is known as a 'training iteration.'

2.1.2 Capabilities of Neural Networks

A network's sensitivity is generally a function of its complexity. This takes shape in quantity of nodes. The Universal Approximation Theorem, outlined by Hornik et al. [11], states that a network may predict any function to any degree of accuracy provided that it have the right number of hidden nodes with appropriate activation functions. [8] This is part of the reason why 'deep learning' networks, or networks endowed with two or more hidden layers, are better in analysis functions.

Neural networks can be used as powerful classifiers of information based on this property. Classification may be thought about as a matter of drawing a line in a plot to separate discretized groupings of triangles and circles (i.e. approximating the function that separates them). A somewhat trained model may produce the following approximation (see Figure 9).

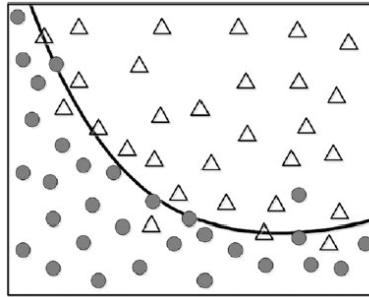


Figure 9. A moderate-fidelity classification curve [6]

Consider now that the two different shapes correspond to two different classes of image (e.g. dogs and cats). This is, in a nutshell, how image classification works; granted, things become much more complicated with more classes of images, but the application of the Universal Approximation Theorem remains unchanged.

2.1.3 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are specialized deep learning networks which have become popular for the purposes of image classification. This is because of their ability to draw intrinsic qualities of objects from an image, and ultimately differentiate low-level from high-level features to help discriminate what is principally important in an image. According to Gatys, “[CNNs] develop a representation of the image that makes object information increasingly explicit along the processing hierarchy.” [12] The architecture of CNNs can be separated into feature extraction and classification network layers, as seen below in Figure 10.

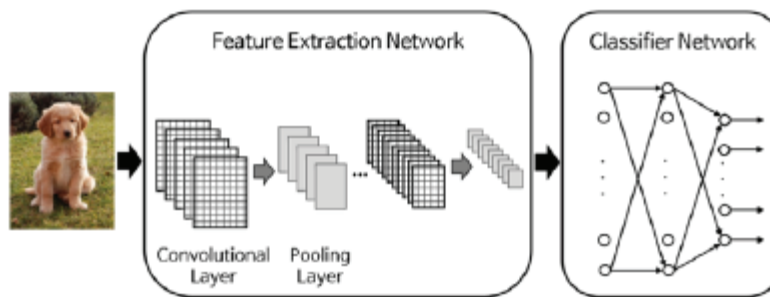


Figure 10. Simplified CNN architecture [6]

The convolution and pooling layers within the feature extraction network function differently from other layers discussed thus far, in that they do *not* operate based on weighted connections between nodes. The convolution layers filter out elementary features from an image by performing the namesake ‘convolution’ operation with a given ‘kernel’ matrix. This process is expanded below in Figure 11.

The figure illustrates two examples of 2D convolution. Each example shows a 4x4 input matrix, a 2x2 kernel, and a 3x3 output feature map. The kernel is represented by a 2x2 matrix with a circled asterisk (*).

Example 1: The 4x4 input matrix is $\begin{bmatrix} 1 & 1 & 1 & 3 \\ 4 & 6 & 4 & 8 \\ 30 & 0 & 1 & 5 \\ 0 & 2 & 2 & 4 \end{bmatrix}$. The 2x2 kernel is $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. The resulting 3x3 output feature map is $\begin{bmatrix} 7 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$. The top-left element of the output (7) is highlighted.

Example 2: The 4x4 input matrix is the same. The 2x2 kernel is the same. The resulting 3x3 output feature map is $\begin{bmatrix} 7 & 5 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$. The top-middle element of the output (5) is highlighted.

Figure 11. Example of the first two convolutions of a 4×4 matrix with a kernel (2×2 matrix in center) [6]

This convolution operation is depicted generally as

$$\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} \circledast \begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} Aa + Bb + Ce + Df & Ab + Bc + Cf + Dg & Ac + Bd + Cg + Dh \\ Ae + Bf + Ci + Dj & Af + Bg + Cj + Dk & Ag + Bh + Ck + Dl \\ Ai + Bj + Cm + Dn & Aj + Bk + Cn + Do & Ak + Bl + Co + Dp \end{bmatrix} \quad [10]$$

where lowercase letters represent entries in the 4×4 matrix and the uppercase letters the entries in the 2×2 kernel. The 4×4 matrix in Figure 11 above is representative of a grayscale image, where each number is the pixel intensity (0-255). Each kernel, or ‘convolution filter’, excels at bringing out a particular fundamental feature in an image (e.g. sharp corners, or vertical lines, etc.). Once the entire image has been convolved, the smaller resultant matrix is produced, known as a ‘feature map’. [6] Several examples are featured with their kernels in Figure 12. While kernels may seem too abstract to be able to interpret such intrinsic meaning, they are determined empirically in the training of the CNN and not manually. [6]

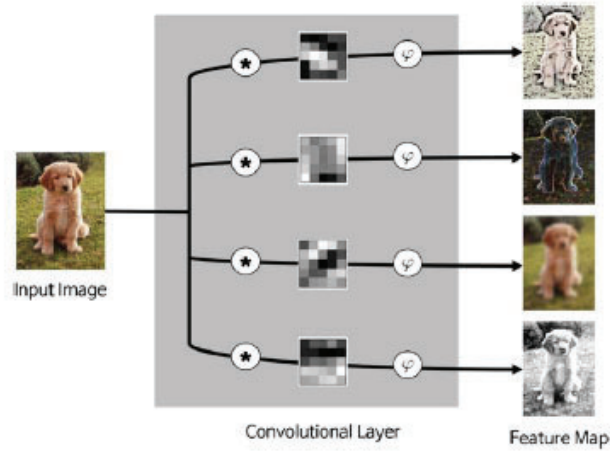


Figure 12. Several examples of the convolution operation [6]

The pooling layers serve to further condense the features brought out in the convolution operations. There are two variants of pooling: mean and max. Consider a 4×4 image undergoing 2×2 mean and max pooling (see Figure 13).

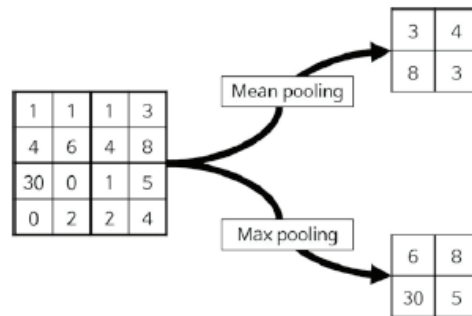


Figure 13. Depiction of the two main variations of pooling [6]

Each flavor of pooling is true to its name. 2×2 mean and max pooling replace each 2×2 sector in the image with the mean and maximum of its values, respectively. This operation shrinks the number of values to be fed in to the next layer. Overall, the various combination of convolution and pooling layers within the feature extraction network exploit the spatial and temporal dependencies within an image to extract features from an image. [13]

2.1.4 Transfer Learning

As a brief note, a common practice in developing deep learning models is to train them in advance with a separate well-defined (presumably large labelled appropriately) dataset in order to then hasten and/or enhance the learning process on the target dataset. Much like the human brain, the ‘pretrained’ network uses

that which it learned via the initial period of training to better make generalizations over the target dataset, relatively so compared to if the network was trained purely on the data available in said dataset. [14]

2.2 Generative Adversarial Networks (GANs)

Imagine that one desired to design a machine learning network that, when prompted with a quantity of data, was able to produce fake data with the same properties. In order for its output to be regulated proactively to avoid significant variations in the image content, such a network would need a comprehensive loss function that adapts to the distribution of data that is being generated.

A Generative Adversarial Network (GAN) is a single compound network consisting of two interacting deep learning networks. This is a flavor of machine learning ‘generative model’, so-called because it uses two networks (Generator and Discriminator) to produce counterfeit data based off a set of real data. A flowchart outlining a general GAN architecture is included in Figure 14 below.

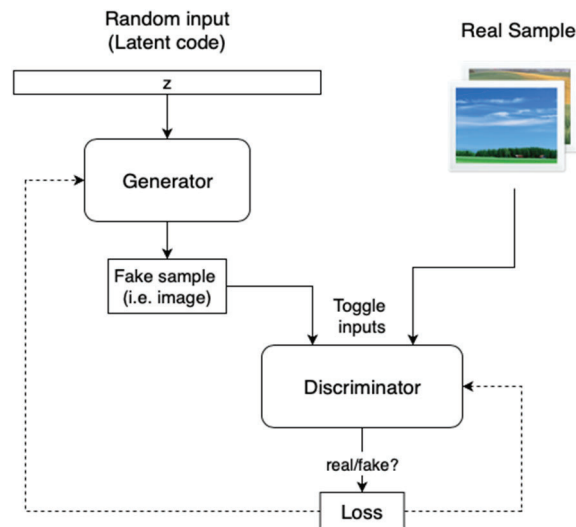


Figure 14. Generalized GAN architecture with entirely random input into the Generator [15]

The Generator and Discriminator are two neural networks. In the above example, the Generator network is input with random noise and tasked with producing a fake image; the Discriminator is alternately input with samples from a real dataset and the images produced by the Generator and tasked with determining whether the image belongs to the real dataset. [15,16]

The change to the random noise made by the Generator may be at first arbitrary due to initial weighting. Over time, the Generator will get better at ‘fooling’ the Discriminator into accepting its output as real, and the Discriminator will become better at seeking out the differences. The two networks are deemed ‘adversarial’ as the Generator aims to minimize a value V and the other the Discriminator aims to maximize it. This can be summed into an algorithm that converges to a Nash Equilibrium [17] between the two networks, which in game theory describes where neither competitor can make any gain by changing their strategy; this is synonymous with driving the Generator’s output to be indistinguishable from the population of training data via the Discriminator. Mathematically, the adversarial objective or loss function $\mathcal{L}_{GAN}$ is

$$\min_G \max_D \mathcal{L}_{GAN}(D, G) = \mathbb{E}_{x \sim p_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))] \quad [11]$$

where G is the Generator output, D is the Discriminator output, p_{data} is the probability distribution of real data, p_z is the probability distribution of input noise, and x and z are samples of these distributions (respectively). D ranges from $[0,1]$, where ‘0’ or ‘1’ are the Discriminator believing an input image to be fake or real, respectively. Note that a lower value of $\mathcal{L}_{GAN}$ corresponds with better Generator performance (and thus better fake images), hence why it is a ‘loss’ function. [18]

In practice, each network is programmed to minimize or maximize its own value function in keeping with the concept of Equation [11]. The Generator’s minimization term ($\mathcal{L}_G$) is

$$\mathcal{L}_G = \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))] \quad [12]$$

where minimization of $\log(1 - D(G(z)))$ is conceptually equivalent to maximizing $\log(D(G(z)))$, or the log-probability that the Discriminator will output ‘1’ (real) for the Generator’s output. The Discriminator’s maximization term ($\mathcal{L}_D$) is

$$\mathcal{L}_D = \mathbb{E}_{x \sim p_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))] \quad [13]$$

where $\log(D(x))$ is the log-probability that the Discriminator will output ‘1’ (real) for a given real sample, and $\log(1 - D(G(z)))$ is $\mathcal{L}_G$; maximizing the sum of these will enforce that the Discriminator classify both real and fake images appropriately. [18]

During training, a unique number of updates can be specified to the Generator and the Discriminator. For the purposes of this paper, this is referred to as ‘training ratio’ (r), and is expressed mathematically as

$$r = \frac{n_G}{n_D} \quad [14]$$

where n_G and n_D are the number of training updates provided to the Generator and the Discriminator, respectively. As a rule of thumb, the training ratio is generally greater than one, as the Generator generally takes more effort to train.

Each network (G, D) is programmed to minimize its own loss and maximize the other’s by learning from the Discriminator’s output. It is the counterfeiter’s (the Generator’s) objective to pass as many of its products off as legitimate to the inspector (the Discriminator). The more the counterfeiter’s products are denied, the more improvements may be made to form. Likewise, as the counterfeiter’s technique improves, the inspector too becomes wise to the tricks specific to the counterfeiter. Figure 15 describes the adversarial process in analogy.

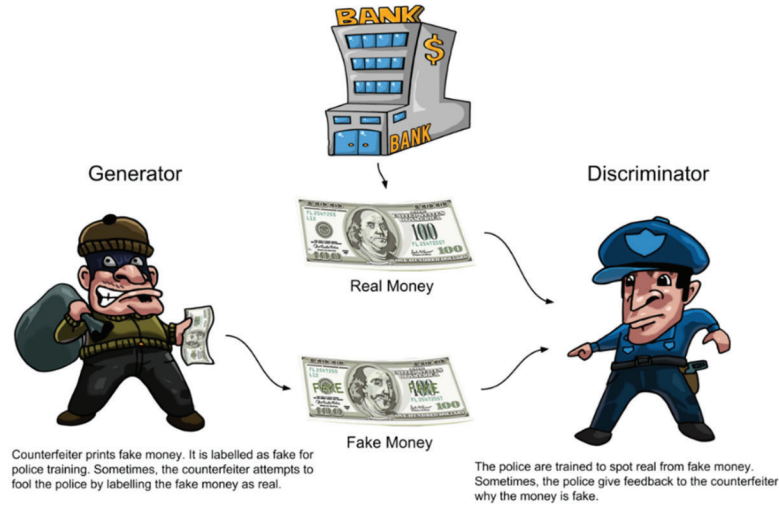


Figure 15. Illustrated GAN analogy [19]

Figure 16 below illustrates the manner in which the learned distribution of the GAN's generated imagery approaches the discrete statistical distribution of the data over the course of an ideal training sequence.

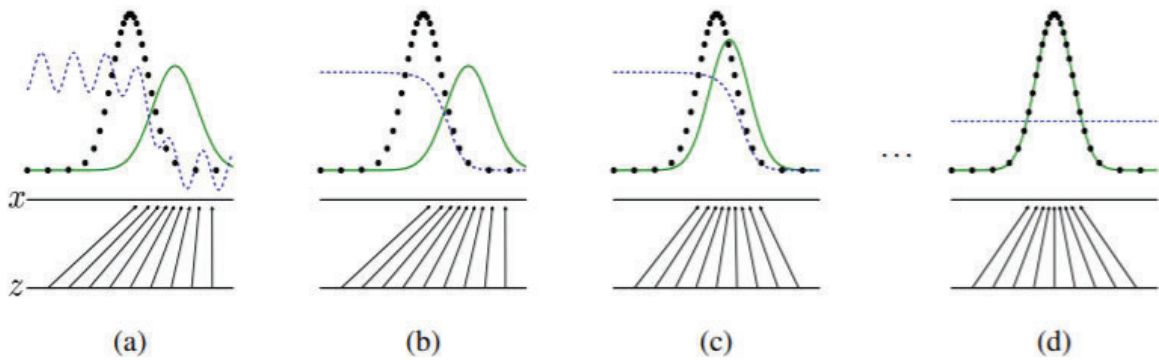


Figure 16. Detailing an ideal case of the GAN's distribution modelling, where the dotted black line represents p_{data} , the green line represents p_g (the probability distribution of Generator output), and the blue line represents the governing loss function; note that the vectors drawn between the bottom two horizontal lines represent how the input noise (z) is mapped to match real data samples (x). [18]

GANs may be adapted for the input of an image into the Generator (in addition to noise) that serves as a template for the output. The architecture that implements this is known as a 'Conditional GAN' (cGAN); see Figure 17 below.

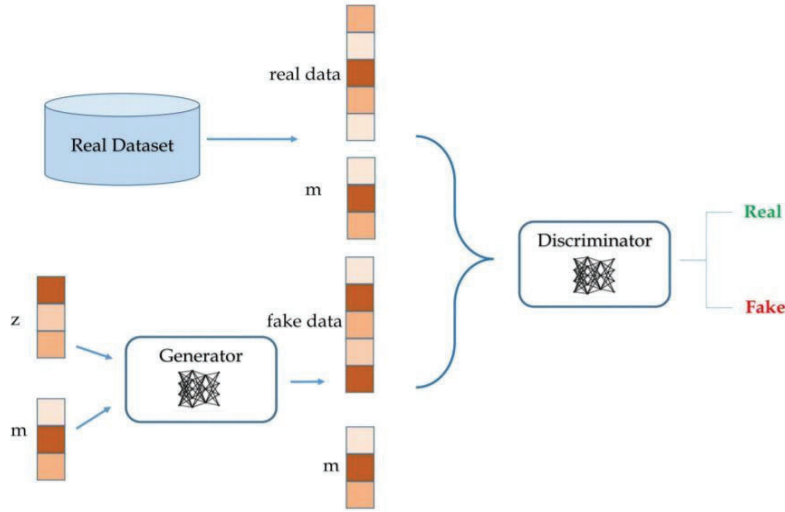


Figure 17. The Conditional Generative Adversarial Network (cGAN) architecture [20]

The Generator utilizes ‘condition’ data (m), random noise (z), and feedback from the Discriminator (Real/Fake) in order to better constrain the fake data it produces. A subset of cGAN architectures excels at image translation, which is a large problem field consisting of ‘mapping’ one image domain to another. In this case, the provided condition data is an entire image to be modified by the Generator. Examples include tasks such as image reconstruction from edge plots [17,21] or transforming horses into zebras, [22] and involve two image datasets that represent each domain. These two datasets may either be paired or unpaired. Paired image translation learns the mapping based on datasets connected by one-to-one pairing (each image in dataset X has a direct counterpart in dataset Y). Of primary note to this paper is unpaired image translation, or the more general translation between two domains of images without such one-to-one pairing. Examples of both are provided in Figure 18.

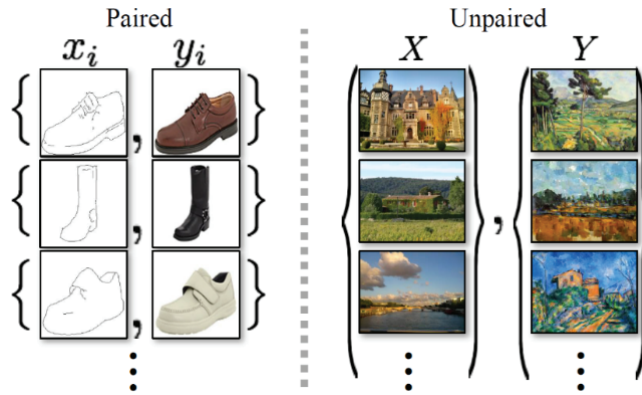


Figure 18. Paired and unpaired image datasets [22]

2.2.1 Cycle-Consistent Adversarial Network (CycleGAN)

Using the human brain, one may picture what a particular landscape would look like if it were painted by Van Gogh using a little imagination, but asking a neural network to do the same offers a challenging task. CycleGAN is a specialized GAN architecture that represents the cutting edge of unpaired image translation. [22]

For example, a trained CycleGAN may be input with a dataset of horses and tasked with changing them to appear like zebras, or vice versa. See Figure 19 for examples of a trained such translation.

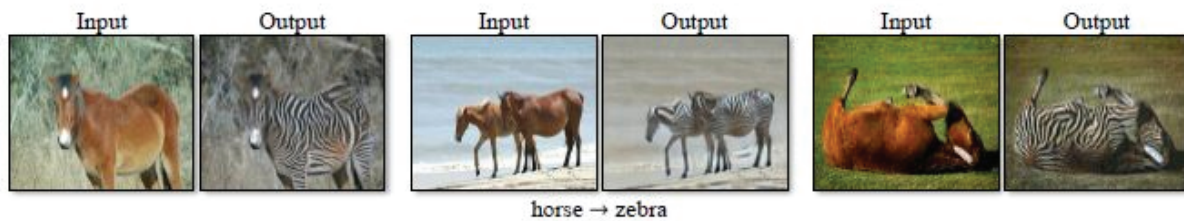


Figure 19. Unpaired horse-to-zebra translation via CycleGAN [22]

It does this by using two complete cGANs, one for the ‘forward’ translation (e.g. horse-to-zebra) and one for the ‘inverse’ translation (e.g. zebra-to-horse) for a given unpaired dataset. In other words, one Generator is input with horses and learns to translate them to fake zebras, and the other is input with zebras and learns to translate them to fake horses. The simplified architecture is provided below in Figure 20.

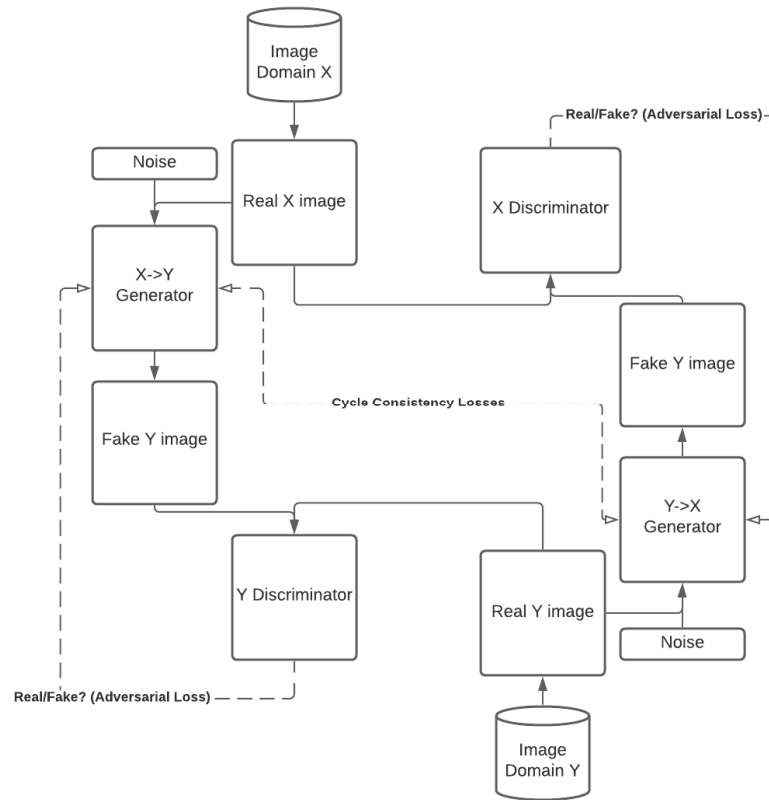


Figure 20. Block diagram of CycleGAN architecture

The structures of each Generator and Discriminator are shown respectively in Figure 21 and Figure 22. Note the input sizes, as they will be relevant in later sections.

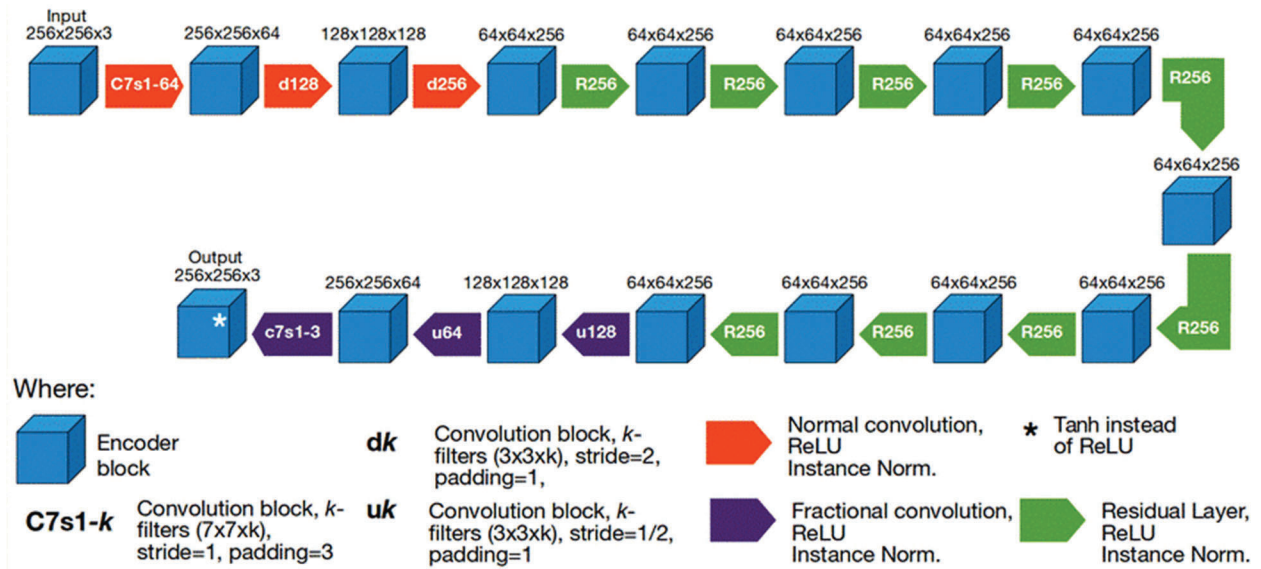


Figure 21. CycleGAN Generator Architecture [23]

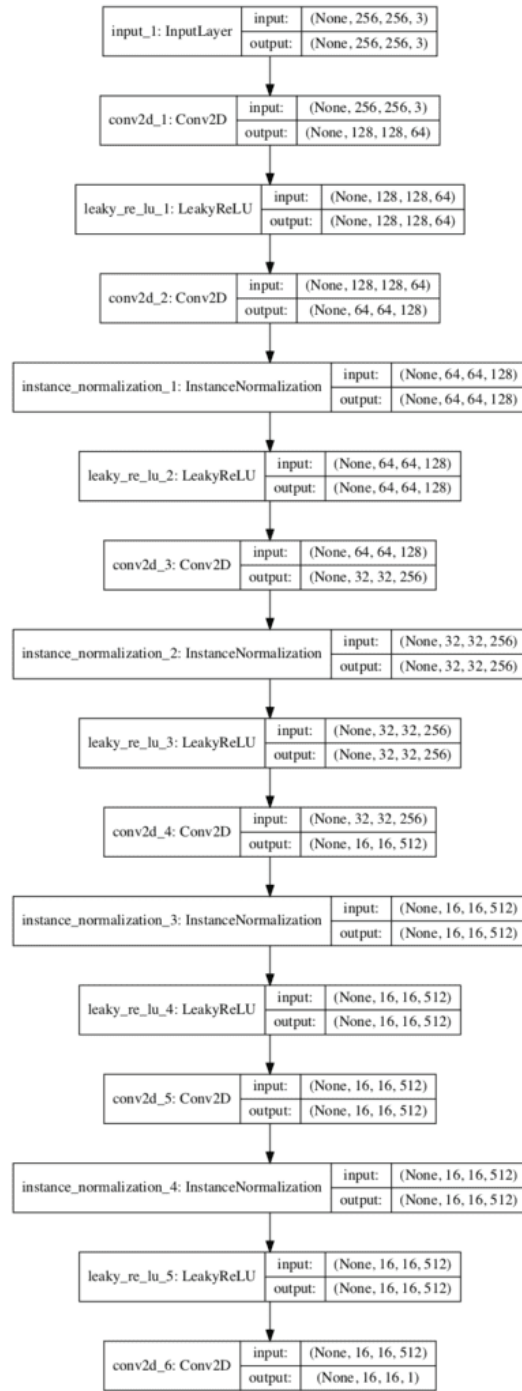


Figure 22. CycleGAN Discriminator Architecture [24]

The CycleGAN architecture capitalizes the relationship between its two ‘yin-yang’ cGANs to produce the high visual quality seen in Figure 19. The Generator’s minimization term, or loss function ($\mathcal{L}_G$) involves three loss calculations: adversarial loss ($\mathcal{L}_{GAN}$, calculated once for each cGAN), cycle-consistency loss ($\mathcal{L}_{cyc}$), and identity loss ($\mathcal{L}_{ID}$, not pictured in Figure 20). [22]

$$\mathcal{L}_G(G, F, D_X, D_Y) = \mathcal{L}_{GAN}(G, D_Y) + \mathcal{L}_{GAN}(F, D_X) + \lambda \mathcal{L}_{cyc}(G, F) + \mu \mathcal{L}_{ID}(G, F) \quad [15]$$

where G is the forward Generator, F is the inverse Generator, D_X is the X dataset Discriminator, D_Y is the Y dataset Discriminator, and λ and μ are constants (hyperparameters of the system that control the relative weighting or importance of the different loss terms). Note that adversarial loss is analogous to that described by Equation [11].

One may conceptualize the two Generators translating between two datasets X and Y as a ‘translation function’ $G(\cdot)$ and an ‘inverse translation function’ $F(\cdot)$, or

$$G : X \rightarrow Y; G(x) = y' \quad [16]$$

$$F : Y \rightarrow X; F(y) = x' \quad [17]$$

where x and y are samples from sets X and Y , which are respectively translated to y' and x' into sets $\hat{Y}$ and $\hat{X}$ (image sets that ‘resemble’ their real counterparts, but are results of translation). These functions should ideally be cyclically permutable, or mathematical inverses of one another. Cycle-consistency loss enforces this to ensure the quality of each translation by converging $F(G(x)) \approx x$ and $G(F(y)) \approx y$. Mathematically, this is done by minimizing the loss term

$$\mathcal{L}_{cyc}(G, F) = \mathbb{E}_{x \sim p_{data}(x)} [\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{data}(y)} [\|G(F(y)) - y\|_1] \quad [18]$$

where $\|\cdot\|_1$ is the 1-norm (or the sum of all pixel-wise differences across an image*), and x and y are samples from sets X and Y (i.e. distributions $p_{data}(x)$ and $p_{data}(y)$). This is depicted visually in Figure 23 (b) and (c), respectively. Once CycleGAN is trained, this term ensures that when a horse is translated into a zebra [$G(horse) = fake\ zebra$], then inverse-translated back to a horse, it appears roughly the same as when it started [$F(G(horse)) = F(fake\ zebra) \approx original\ horse$]. [22]

* This means that the images are essentially treated as matrices, subtracted, and the resulting entries are summed. Note that this strong connection to the images’ ‘pixel space’ in the loss function is partially why CycleGAN interprets data visually very well (as humans see imagery in pixel space!).

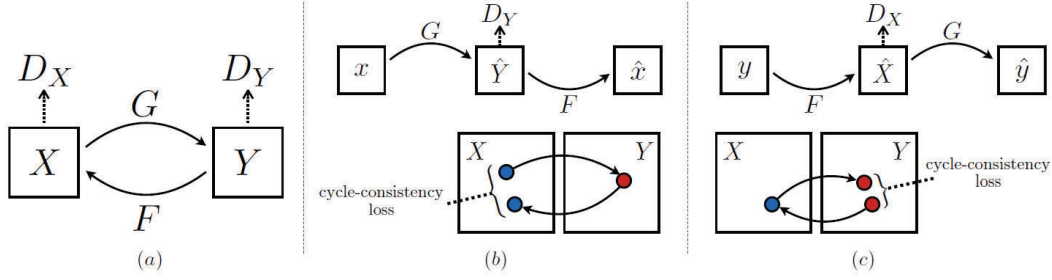


Figure 23. Illustration of concept of cycle-consistency loss and how it uses both Generators' mapping [22]

Identity loss is an additional loss function term applied independently to each Generator that helps ensure that no extreme mapping cases are being promoted (e.g. no unnecessary color changes). This loss is derived from the thought that, if an image from the Y domain is input into the X to Y Generator, the output should be unchanged since the goal of that Generator is to make images look like Y. This goal is attained by minimizing the loss term

$$\mathcal{L}_{ID}(G, F) = \mathbb{E}_{x \sim p_{data}(x)} [\|F(x) - x\|_1] + \mathbb{E}_{y \sim p_{data}(y)} [\|G(y) - y\|_1] \quad [19]$$

where Figure 24 describes this visually.

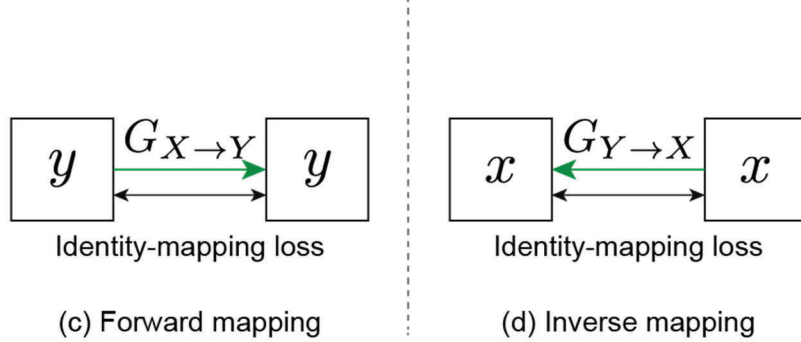


Figure 24. Illustration of identity loss as it applies to each Generator's mapping [22]

CycleGAN is limited in translating extreme geometric/structural changes between image domains; for example, it fails when attempting to translate images of dogs to cats. However, it excels with style translation, e.g. horses to zebras (Figure 19); in this scenario, CycleGAN maintained the salient visual features of the horse while projecting well the style of the zebra images (this is indicated by striping transfer, and change in background grass tint). [22]

2.2.2 Weaknesses and Common Errors of GANs

Generally, when training neural network models, one must be sure to train the model sufficiently over a given dataset. For GANs, lack of training can lead to improper representation of a target dataset's

characteristic, and for image GANs this can commonly result in superficial disparities such as discoloration or false texture. For CycleGAN, this can manifest itself in both random superficial changes as well as a small or trivial change made to the input. The latter is especially pertinent for this analysis, as to human judgement output imagery may appear realistic, when in reality the change made to the input is not a meaningful one to the perspective of the CNN that would be training on it.

One must also be wary of ‘overtraining,’ or conditioning the model excessively on a particular set of data. For GANs, this can lead to ‘mode collapse,’ or the GAN converging to a trivial solution and outputting the same or similar image for many different inputs. An example is included below. [25]

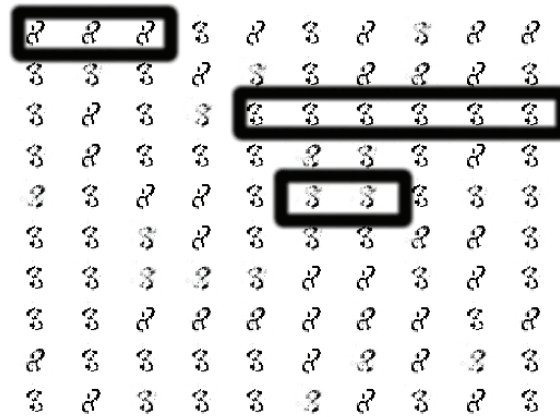


Figure 25. Mode collapse for a GAN trained to produce hand-drawn eights [25]

Moreover, for CycleGAN specifically, one tends to see more and deeper, structural changes made to input imagery as it progresses further into training. This will be expressed more in section 5.

3 Chosen Datasets and Preliminary Analysis

Style transfer for pose estimation was researched over two overarching translation datasets, each containing datasets of real and synthetic imagery of a specific geometric [satellite] model.

3.1 Proof of Concept Datasets

Based on the low quantity of associated (unpaired) satellite imagery that is publicly available, the idea for a ‘toy model’ that will permit maximum control over the researched method was born. Originally, the concept was that the real dataset would be created by taking pictures of a geometrically primitive model (i.e. the ‘Block Model’, see Figure 26), and the synthetic dataset will be made by rendering an identical model in the *Blender* digital studio. [26]

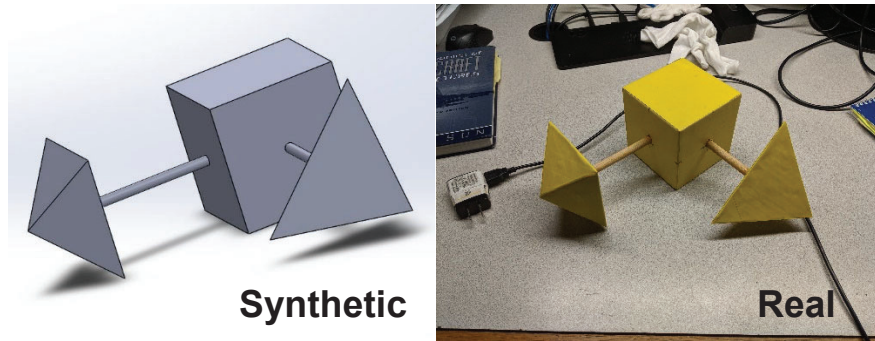


Figure 26. Examples of images from synthetic and real sets of the ‘Block Model’ (note that the real images will have a black background)

However, COVID restrictions during the 2020 Spring semester prevented access to the environment and equipment necessary to make this happen. Instead, an alternative ‘Proof of Concept’ (POC) dataset was generated entirely within *Blender*. The two datasets were generated from the same poses at a uniform range, but the rendering engine utilized was changed for each case. The ‘synthetic’ set was generated using the less-sophisticated ‘Workbench’ engine that produces more lighting exposure and no texture/noise, while the ‘pseudo-real’ set was generated using the ‘Eevee’ engine that allows for more control over these aspects and better lighting. In general, these two sets were, in general, intended to be different to mimic the respective difference between true real images and synthetic ones. In addition, due to its engine the synthetic set has a dark gray background, which became a fortunate carry-over to the real SPEED dataset’s brightness of background.

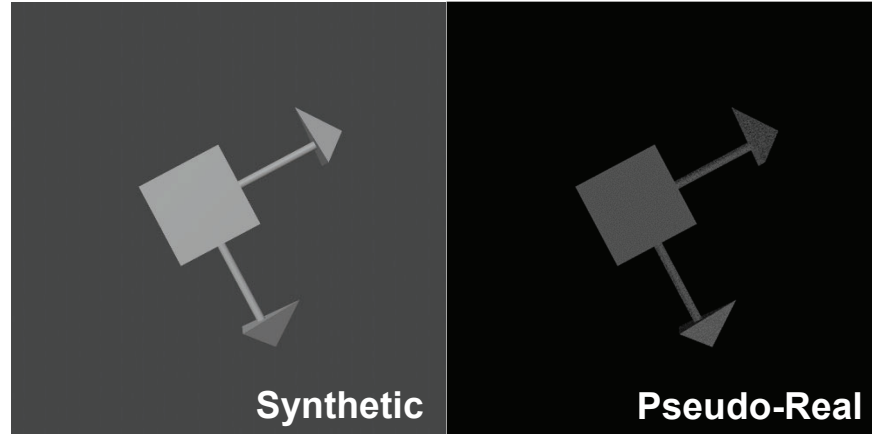


Figure 27. Examples of images from synthetic and real sets of the POC Dataset

However, in early analysis of results on the POC dataset, it was noted that it might emphasize too much change in the background's style versus the foreground's, which would hamper the research's attempt to consider the full range of stylistic differences between real-world and synthetic imagery. To evaluate the more subtle aspects of style in POC and to draw CycleGAN's attention to the foreground, a second POC dataset (POC2) was created in a similar way to the first. The 'synthetic' set was generated using 'Eevee' engine with the same settings as the previous, while the 'pseudo-real' set was generated using the even more sophisticated 'Cycles' engine that has much more realistic lighting. The two datasets were still generated from the same poses at a uniform range, but the lighting was randomized for each image independently, and background color was made uniformly black in both. Figure 28 below contains several examples of each set of POC2.

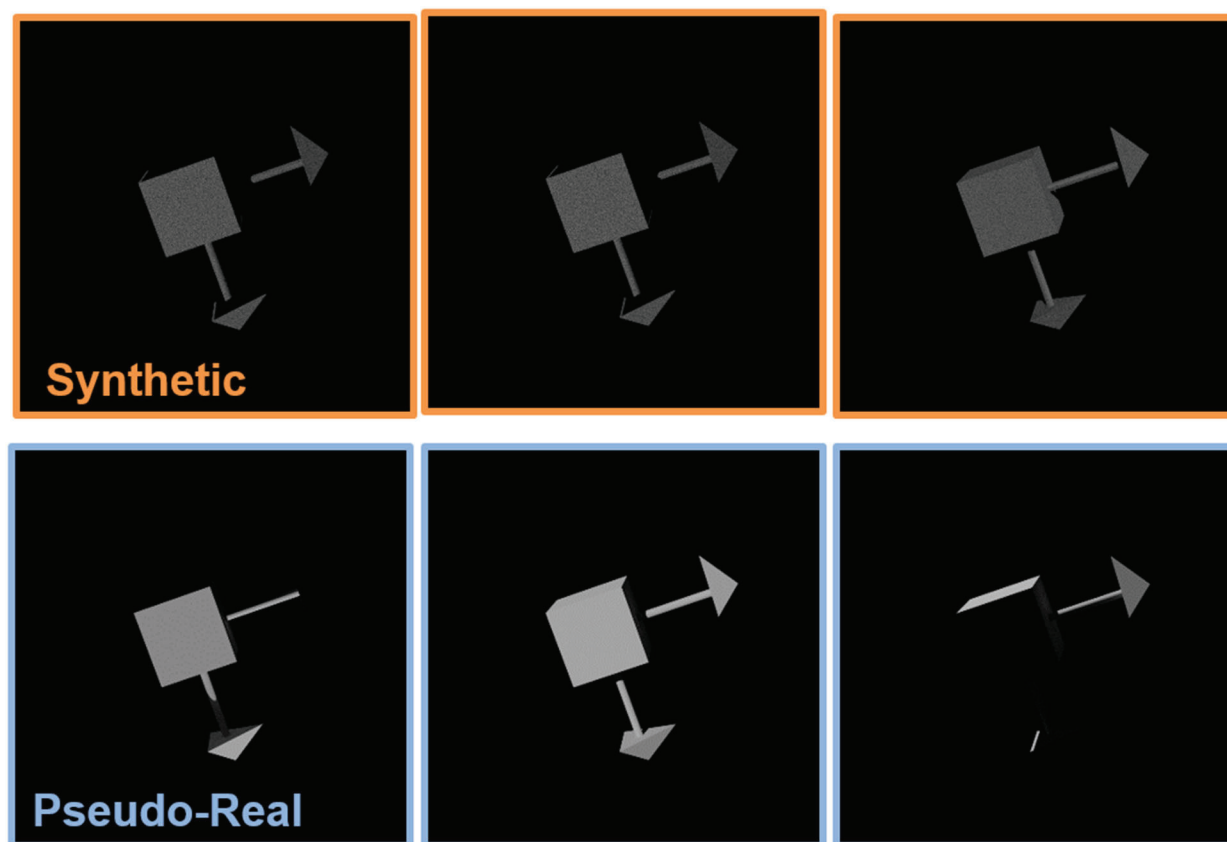


Figure 28. Examples of images from synthetic and real sets of the POC2 Dataset

Note the implications of independently randomized lighting; below are some examples of where lighting significantly masked one, but not the other.

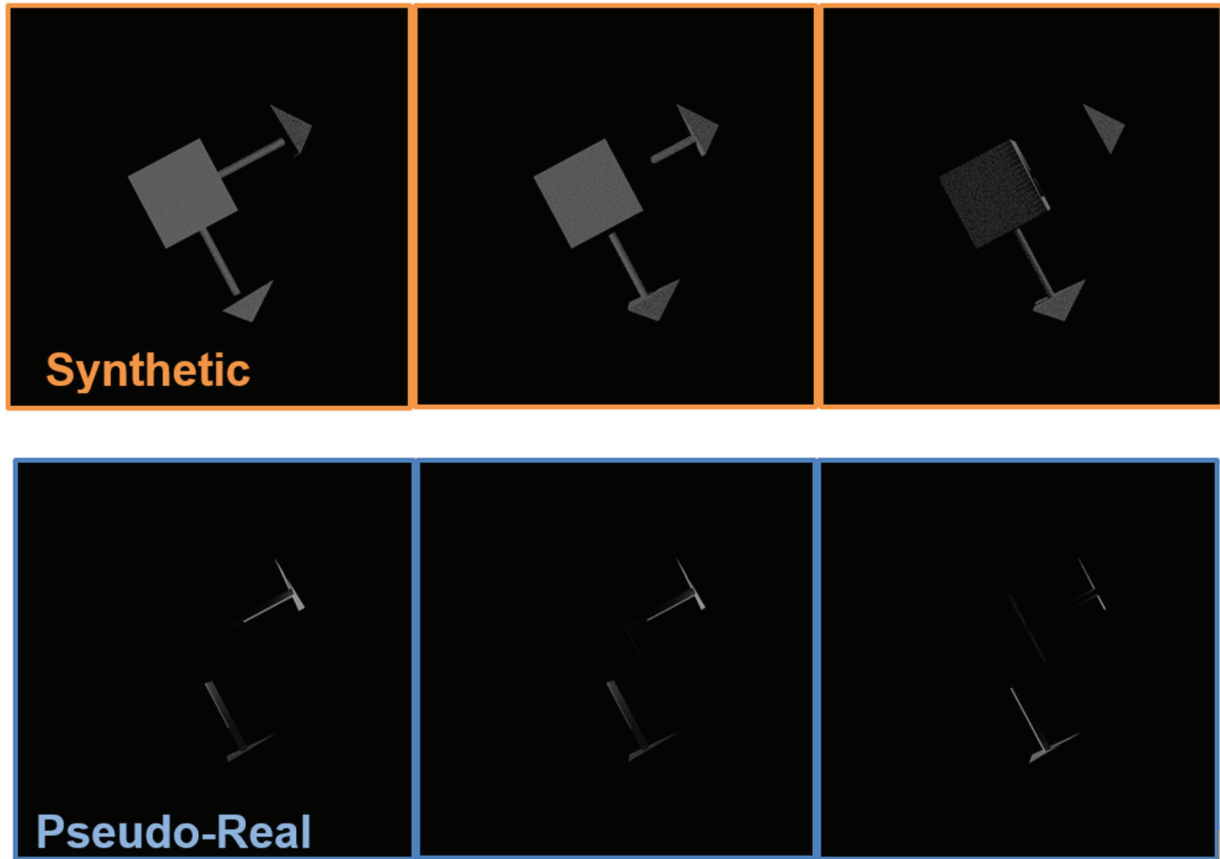


Figure 29. Examples of differences in lighting between images from synthetic and real sets of the POC2 Dataset

Two sets of 27,000 images (pseudo-real and synthetic) were generated for both POC and POC2.

The simplicity of the geometry of this model was deliberate, in order to allow isolated analysis of the GAN's ability to transfer style alone, and not structure. With the control available over the generation of both synthetic and pseudo-real data for this model and it was used to develop the research process and 'smooth out' details.

3.2 Spacecraft Pose Estimation Dataset (SPEED)

The second model to be tested will be that used in the ESA/Stanford Pose Estimation Challenge, the Tango small satellite. Its training dataset (labeled) consists of 12,000 synthetic images and 5 real images, and its test dataset (unlabeled) consists of 2,998 synthetic images and 300 real images. [27] If an image is labelled, its complete 3D pose with respect to the reference frame of the camera is provided. See below for examples of each image class.

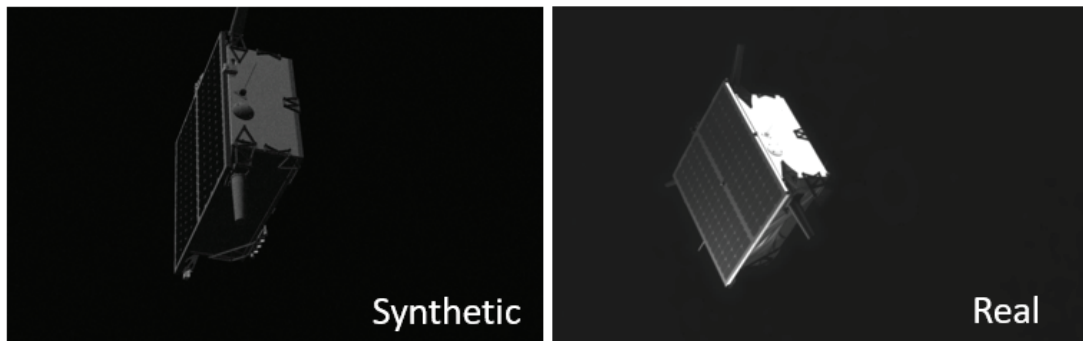


Figure 30. Examples of images from the synthetic and real sets from SPEED [27]

The quantity of real images is relatively low compared to the quantity of synthetic images, and will offer an opportunity to analyze the effects this will have on the output photorealistic imagery. Such discrepancy in real imagery is fully expected for such real-world satellite pose estimation tasks, and thus this will be a pertinent point of analysis.

4 Research Method

4.1 CycleGAN Implementation

The particular implementation of CycleGAN that was used for this research was largely borrowed from Jason Brownlee’s literature, using the Keras library in the Python 3.7 coding language; the full documentation of this code can be found in [24].

4.2 Analysis and Selection of CycleGAN Output for Photorealistic Dataset

A fundamental challenge with using such high-level deep learning techniques, such as CycleGAN, is the tendency to view them as ‘black boxes,’ or exclusively analyzing the efficacy of a model to take input and produce desired output. This approach obviously leaves a lot to be desired in terms of the critical analysis of the internals of such models, and when taken to an extreme, can be a blind spot to their potential vulnerabilities. [28]

However, due to the peculiar dynamic nature of GANs and the high dimensionality of input and output data, it is difficult to evaluate and compare them from a standpoint that is not subjective; Equations [11]-[13], as discussed, enforce a Nash Equilibrium, which is completely relative to the particular Generator/Discriminator pair at their particular stage of training and not transferrable to the training of other GANs, or even different training runs of the same GAN architecture. Furthermore, the ability of CycleGAN to behave and converge predictably, as will be demonstrated in later sections, is hampered by the low quantities of data encountered in the principle dataset analyzed in this research (SPEED), making comparison between different training runs all the more difficult.

The CycleGAN Generator prioritized in the course of this research is the one trained to transform synthetic satellite imagery to real, or S2R. Its complement Generator, that which is trained to transform real satellite imagery to synthetic, will be known as R2S. With this research, care was taken to balance qualitative, black box input/output analysis of CycleGAN with a deeper, quantitative validation of the stated hypothesis. For a given training run and dataset, CycleGAN’s S2R input and output was visually compared throughout the training in order to determine the points where the output appeared visually similar (the criteria used are defined below) to the dataset’s real images. R2S output will also be observed for completeness.

Once these points are selected, they were used to index the S2R Generator at that iteration of training, which were then input with the full training dataset of synthetic images in order to output ‘photorealistic’ datasets. These datasets were used to train the selected pose estimation CNN, the results of which were used to provide quantitative validation to back up qualitative analysis (as will be discussed further in 4.4).

It is important to note that the current state-of-the-art research in CycleGAN leans on this type of ‘opaque box’ research. Granted the insufficiency of using human perception as the sole metric (discussed in 1.3), it was nonetheless an effective way to potentially determine several converged Generators (output imagery appears similar to imagery from the real dataset) by ruling out the un-converged Generators (e.g. output is riddled with noise or has undergone mode collapse).

4.3 Photorealistic Training of Pose Estimator CNN

As a major goal of this research is to evaluate the impact on pose estimator performance that such photorealistic training data has, the principle ‘metric.’ Once a photorealistic dataset was selected amongst the refined datasets produced from a trained synthetic-to-real (S2R) Generator from CycleGAN, it was used to train a selected CNN-based pose estimator architecture. This specific per dataset and is outlined in the following sections.

To limit the scope of the problem, the CNN ‘pose estimator’ utilized in this research was only trained to estimate orientation angles of a target object (and not position). See further discussion in 7.3.

4.3.1 Pose Labelling Method

Since position was neglected for this simple pose estimator, for the following analysis the term ‘pose’ is used to describe 3-D orientation, but is easily generalizable to the full 6 degrees of freedom. In order to understand the way that a range of pose is divided up into discrete labels (or how an image would be assigned from its ‘ground truth,’ or exact/true pose to a pose label), the following diagrams are provided. Imagine that each possible pose, which consists of the ZYX Euler Angle triplet $(\varphi, \theta, \psi)^\dagger$, is mapped to cartesian coordinates as follows in Figure 31.

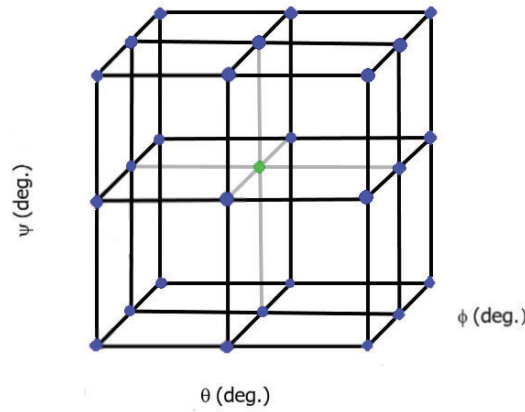


Figure 31. Projecting onto orthogonal coordinates to visualize all possible 3-D orientations within a range

Since this is a continuous range that needs to be converted into a discrete one, one may then consider cubic ‘buckets’ around a few selected points. Thus, for the example provided in Figure 32, all poses that fall within the yellow cube would be assigned a label corresponding to the blue pose point.

[†] The ZYX Euler Angle sequence (φ, θ, ψ) is analogous to the XYZ Roll-Pitch-Yaw convention. [29]

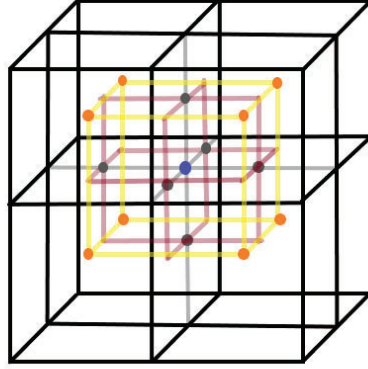


Figure 32. Example of the discretization scheme used to assign a range of poses (bounded in yellow) to a specific pose label (blue point)

The method selected to define these buckets over a selected range involves defining the first label point (or starting point) for each axis and the desired cube width (or label width). For a single axis, this is indicated in Figure 33. Note that, for each label, lower bounds are exclusive and upper bounds are inclusive.

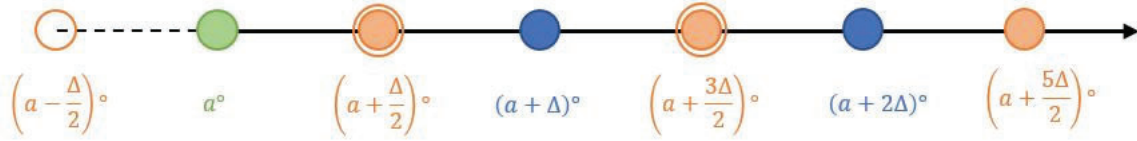


Figure 33. Visualization of how labels were selected on a specific axis over a range, where the green dot is the starting point label, the blue dots are labels, and the orange dots are the label bounds; note that lower bounds are exclusive (open circle) and upper bounds are inclusive (solid circle)

One may see this logic generalized to all constructed three dimensions below in Figure 34. Note that a and Δ are kept the same in this research for all three orientation axes.

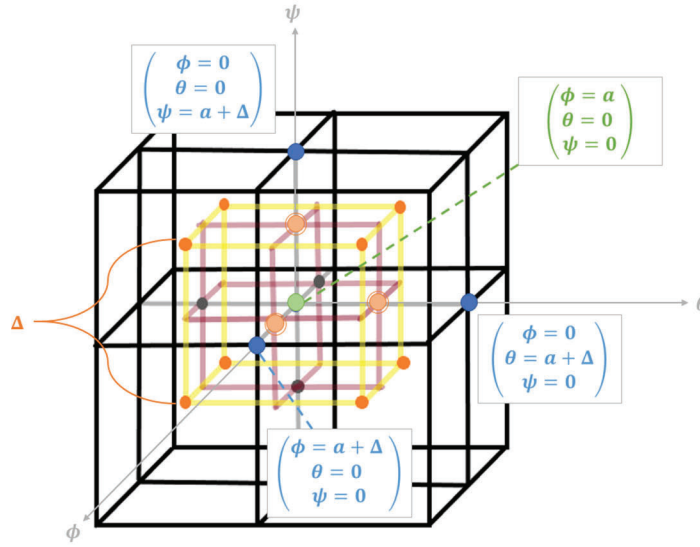


Figure 34. Discretization scheme used above, labelled symbolically

Note that this is repeated for the entire range of desired poses. For $\alpha = 5^\circ$ and $\Delta = 45^\circ$, over the entire domain of $[0^\circ, 360^\circ]$, this is illustrated in Figure 35 (representing any one particular orientation axis).

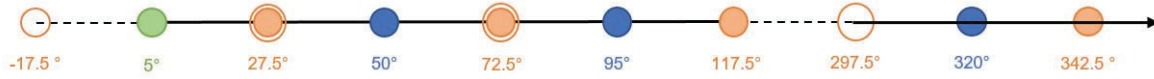


Figure 35. Example of a pose label configuration and its associated bounds over the range $[0^\circ, 360^\circ]$

Note that, since $\frac{\max - \min}{\Delta} = \frac{360^\circ - 0^\circ}{45^\circ} \in \mathbb{Z}$, the bottom-most and the top-most labels mate ideally ($342.5^\circ = -17.5^\circ$). For others, based on the coding discipline utilized, there is a segment of possible poses that is left uncovered by any label for the specified range. This approach is generalized for use on all three pose axes, and starting point and label width were held constant across all three for any specified configuration.

4.3.2 Chosen CNN Pose Estimator, and Settings

As the scope of this research does not involve the design or comparison of different CNN-based pose estimation algorithms, a simple one was chosen. Training was performed via transfer learning (see 2.1.4) to a pretrained ‘AlexNet’ classifier CNN; this classifier architecture was used for the later pose estimation tasks for POC and SPEED datasets. The AlexNet architecture is included below in Figure 36 for reference.

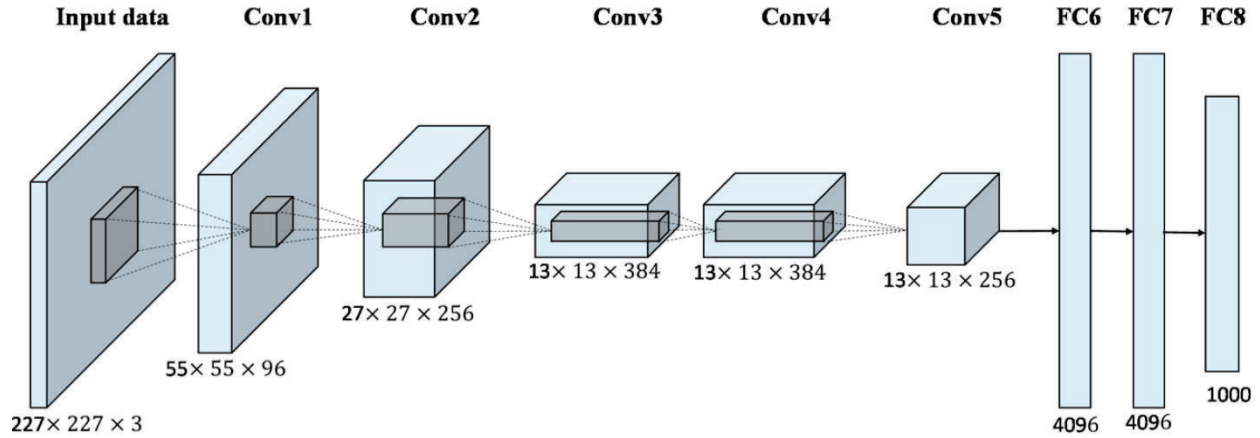


Figure 36. Base AlexNet CNN Structure [30]

This architecture was adapted for the Note that the pretrained weights for AlexNet were those resulting from training on the >1 million images of the ImageNet dataset, broken up into 1,000 image object classes (provided in the respective MATLAB add-on, [31]). In order to apply this network to the number of desired pose labels (classes) involved with the pose estimation (classification) of any one of the selected datasets, the final (output) layer (labelled as FC8 above) was excluded and replaced directly with output nodes corresponding to the number of labels. Note that the activation function utilized in this output layer still utilizes the Softmax function (see Figure 4).

This transfer learning was conducted for 8 epochs, with a constant learning rate of .001 and batch sizes of 25 images per iteration for SPEED and 64 for POC/POC2.

Note that, in a separate analysis[‡], transfer learning training using this architecture saw roughly 90% validation accuracy when training on anywhere from 50 to 100 images per label. However, this falls to roughly 84% at 40-45 images per label.

4.4 Quantitative Validation

The results from this trained model will be compared to those of an identical CNN that was trained on the original synthetic dataset (pre-Generator refinement). This process is described below in Figure 37.

[‡]See Appendix A: Training Validation Accuracies at Increasing Images per Label

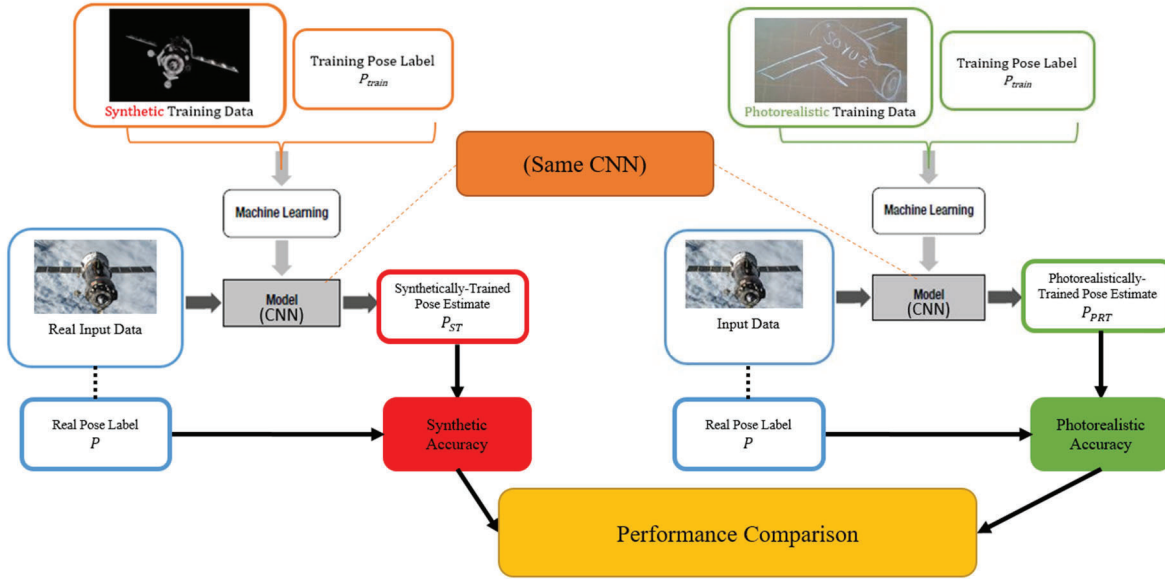


Figure 37. Validation via comparison of pose estimates from synthetically-trained CNN (left) and photorealistically-trained CNN (right)

For a given image dataset, this approach utilizes two instances of the same CNN architecture, one of which is trained on the dataset's synthetic images while the other is trained on the corresponding photorealistic images (synthetic images processed by trained CycleGAN S2R Generator). As a result, these parallel CNNs are trained on the same number of images over the same pose labels, with the only difference between them being the functional addition of the S2R Generator's pre-processing of the training data. Note that the synthetic and photorealistic datasets will also be using the same pose labelling configuration (see discussion in 4.3.1).

Pose estimation accuracies were calculated for each CNN over the same test/validation dataset of real images. This was more precisely termed a class accuracy, and was based entirely on the percentage of images in the validation set for which the CNN classified correctly into pose labels. Mathematically, this is

$$acc = \frac{\text{no. correct pose classifications}}{\text{no. images in validation set}} \quad [20]$$

where *acc* is class accuracy. Not taking into consideration the error in any incorrect pose label classifications does depart from the physical reality of the classifier as a pose estimator. In real world systems, an error is not simply an error, and a faulty pose estimation is measured in the magnitude of difference between it and the correct pose. However, this research is not meant to evaluate the CNN as anything but a pose classifier, and thus this accuracy formalism is useful.

This method above others was selected to evaluate CycleGAN's output out of abundant caution to avoid the problem with synthetic training data; i.e., because we cannot trust humans to judge photorealism in the sense that matters to the CNN. It generalizes little about the internal nature of CycleGAN, and grounds the research in its priority to improve CNN performance on real data.

While this approach evaluates the two networks based explicitly on input-output, it also provides subtle insight to the carry-overs between CycleGAN’s learned convolutional features to those of the CNN in general, something that, if shown, has repercussions for other applications of style transfer enlargement of training datasets.

4.5 Application of Method to Chosen Datasets

4.5.1 POC and POC2

As POC and POC2 were designed for use in this research, the methodology was relatively simple (as seen below in).

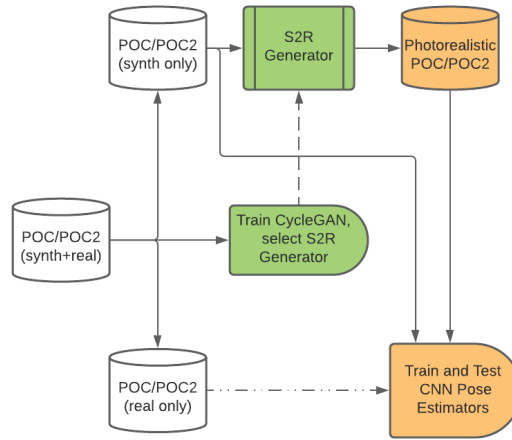


Figure 38. POC/POC2 testing pipeline

Note that the datasets were already sorted into labels (discussed in the next few paragraphs), allowing for immediate input for training/processing via CycleGAN (see green highlights) and for CNN training (see orange highlights).

Both POC and POC2 were distributed over a limited range of poses. Images were generated in Blender by orienting the object with respect to the camera with the following set of ground truth poses:

$$\phi \in \{152^\circ, 154^\circ, 156^\circ, \dots, 208^\circ, 210^\circ\} \text{ (30 total)}$$

$$\theta, \psi \in \{-28^\circ, -26^\circ, -24^\circ, \dots, 28^\circ, 30^\circ\} \text{ (30 total)}$$

One image per ground truth was generated, amounting to $n_{images} = 30^3 = 27,000$. These images were then placed into respective pose labels based on the convention in Table 1.

Table 1. POC, POC2 Dataset labelling conventions

Dataset	Starting Point (a)	Label Width (Δ)	Minimum Number of Images Per Label	Number of Labels	Total Number of Images
POC	5°	10°	100	216	27000
POC2	5°	10°	100	216	27000

This configuration was selected for the generous image number per label (100 total) and for its fineness in terms of label width. Due to the limited range of poses the images occupied, the valid labels were as follows:

$$\Phi \in \{155^\circ, 165^\circ, \dots, 205^\circ\} \text{ (6 total)}$$

$$\Theta, \Psi \in \{-25^\circ, -15^\circ, \dots, 25^\circ\} \text{ (6 total)}$$

The total number of valid labels, as indicated in Table 1, was then $n_{labels} = 6^3 = 216$.

Target object was already centered in each square image frame of 1080x1080 pixels. This was a higher image resolution than what is default input to CycleGAN, and thus downsampling and not upsampling was required. In addition, since the image was already square, there were no distortion issues when downsampling to the square input for CycleGAN (see Figure 21 and Figure 22). Further discussion is provided on downsampling in 4.5.3.

4.5.2 SPEED

As SPEED was obtained from a competition in which pose estimation algorithms were only expected to utilize synthetic training data and labels to predict the full pose of a range of satellite imagery, this progression was not so simple (as seen below in Figure 39). Note that many of these components were heuristically developed in lieu of result, and were chosen for their ultimate benefit to training stability.

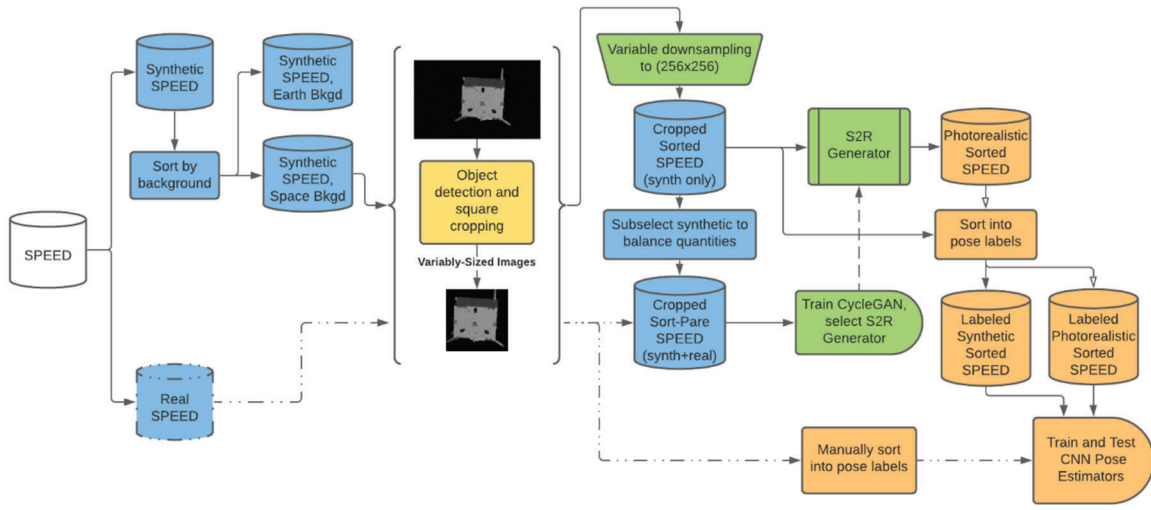


Figure 39. SPEED testing pipeline

4.5.2.1 SPEED Sorting and Paring Down (BLUE)

Synthetic SPEED images have a background of either starless space (black) or Earth in various levels of shading. Real SPEED images only have black backgrounds. The synthetic dataset was sorted based on background for simplification of image processing operations (hence the ‘Sorted’ dataset). The synthetic Earth-background images were omitted from this dataset, as there is no equivalent in the real dataset (see Figure 40).

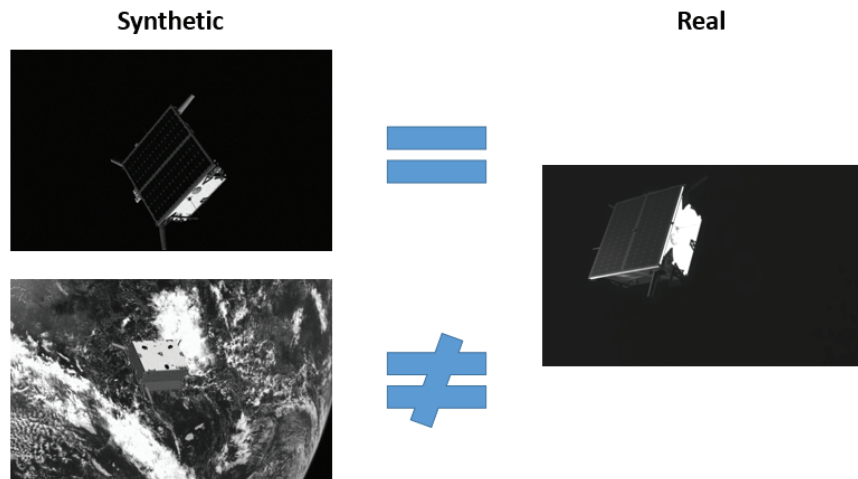


Figure 40. Lack of Earth background representation in real SPEED imagery [27]

In addition, for CycleGAN style transfer it is shown that equal-numbered datasets are optimal. Thus, ‘sorted SPEED’ was further reduced by paring the synthetic dataset down to an equal quantity with the real dataset (300 images); the resultant dataset was deemed ‘sort-pare SPEED’. These 300 synthetic images were hand-picked to be diverse and most representative of the variety across SPEED.

The breakdown of each of these SPEED versions by quantity and background type is seen in Table 2. More detail is included in 5.1.2.

Table 2. Quantitative breakdown of datasets derived from SPEED

SPEED Variant	Background	Training		Testing	
		Real	Synthetic	Real	Synthetic
Sorted	Space	300	7048	5	1873
	Earth	0	4952	0	1125
Sort-Pare	Space	300	300	5	5

4.5.2.2 Object Detection and Cropping (YELLOW)

In addition, to focus and center the satellite body from the original variable-range rectangular aspect ratio image to a square frame, SPEED was cropped using OpenCV’s `grabCut()` function. `grabCut()` utilizes a classical, iterative segmentation algorithm, which is designed to distinguish the image’s foreground (the satellite) from its background. It was allowed 10 iterations to refine its detection criteria. [32] As a follow-on from Figure 30, below are the cropped versions of those images.

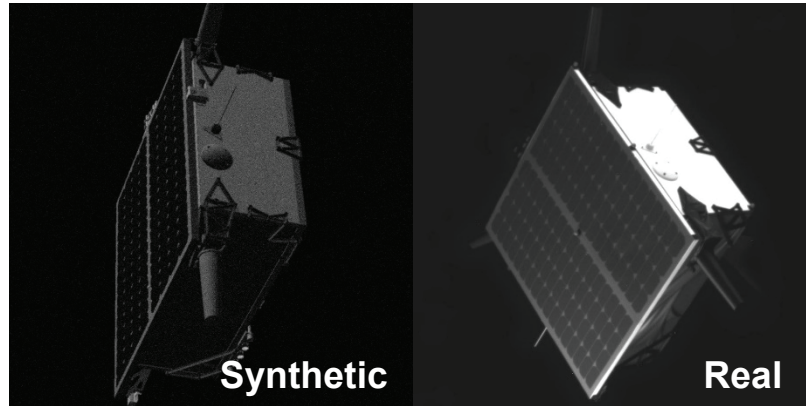


Figure 41. Examples of images from the synthetic and real sets from SPEED, cropped with `grabCut()` [27]

4.5.2.3 CycleGAN Training and Processing (GREEN)

As indicated in the green section of Figure 39, the cropped Sort-Pare SPEED dataset was used to train the CycleGAN.

The cropping operation removes some variety from its subject dataset, as it centers and zooms uniformly to the image object. For SPEED imagery, this removed all variation of position and, ideally, only orientation differences distinguish images (see Figure 42).

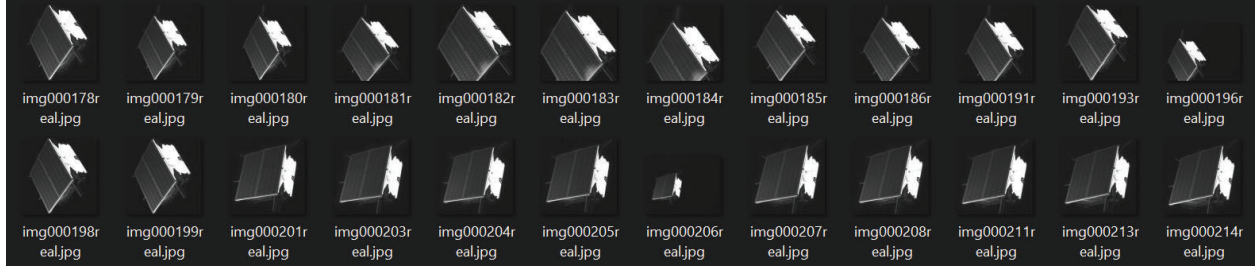


Figure 42. Cropped real SPEED imagery; note that, though there are slight differences, there are large groups of images that share very similar orientations (i.e. few modes of pose)

A potential consequence of this reduction of variation in the training data is subsequent overtraining in CycleGAN. A fairly standard practice during training known as ‘data augmentation’ was introduced during CycleGAN training, where images are randomly replaced by rotated and reflected versions of themselves. Care was taken to only introduce those augmentation transformations that did not warp the geometry of the image so that the CycleGAN only learns stylistic and not structural differences between synthetic and real images. An example of such geometry-preserving training data augmentation can be seen below.

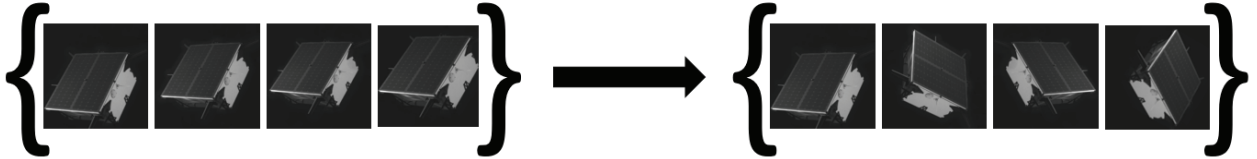


Figure 43. Illustration of geometry-preserving data augmentation [41]

This introduces diversity ‘artificially’ to the dataset, i.e. without adding any new data. As this was only applied to CycleGAN training where no pose labels were required, there was no risk from inadvertently changing pose label through this process.

Several trained S2R Generators were selected from various points of CycleGAN training based on the quality of their test output on the Sort-Pare SPEED. Each of these trained Generator models were then used to process the entire synthetic training dataset of Sorted SPEED, creating a photorealistic dataset.

This might seem like a lot for a given trained Generator model to generalize (processing 7,048 synthetic images based off of training with 300 real and 300 synthetic images). However, due to the relatively low diversity between the Sorted (space background) and Sort-Pare synthetic datasets (see summary discussion in section 8), this is expected to result in a generally uniform quality of image across the refined dataset.

4.5.2.4 Labelling (ORANGE)

The fact that Sorted SPEED’s synthetic dataset was significantly lower in quantity than that of the POC dataset made it challenging to train the CNN to the same degree of accuracy. This is in addition to the fact that its images ground truth poses are distributed over the full range of pose ($\phi, \theta, \psi \in [0^\circ, 360^\circ)$), which made it challenging to find appropriately-sized pose labels.

Additionally, the labels chosen for the real SPEED validation imagery would have to be a subset of those chosen for synthetic SPEED, as the CNN will only be able to accurately classify to the pose labels with which it has been trained. Test data must therefore also fall within these pose labels.

4.5.2.4.1 Synthetic SPEED

The three factors that were taken into consideration when choosing the label scheme were

- i. Maximizing the number of images per label
- ii. Maximizing number of resulting valid labels
- iii. Minimizing the coarseness of the label

Ultimately, by comparing different plausible label configurations, the details of those used for training are included in Table 3. Refer to 4.3.1 for definition of terms.

Table 3. Selected SPEED Label Configurations (those used for training are highlighted in gray)

Configuration	Starting Point (a)	Label Width (Δ)	Minimum Number of Images Per Label	Number of Valid Labels	Total Number of Images
1	5°	30°	45	3	147
2	5°	35°	45	6	306
3	5°	40°	45	31	1699
4	5°	40°	50	20	1187
5	5°	45°	45	46	2770
6	5°	45°	50	36	2300

Configurations 5 and 6 were selected for use in latter analysis for their moderate image number per label and overall total, for their moderate number of valid labels, and their moderate granularity in terms of label width. Note discussion in 4.3.2 regarding number of images advised for chosen CNN.

4.5.2.4.2 Real SPEED

The 300 images of real SPEED were not provided with ground truth poses. However, with the coarseness of the labeling configuration selected, and the relatively few modes of pose revealed by cropping in real SPEED (see Figure 42), manual sorting became a plausible course of action. For each pose mode in the real imagery, an approximate pose was recorded and matched with the valid pose labels of Configurations 5 and 6 of Table 3. In some cases, the real images of a pose mode had to be rotated in increments of 90° in order to properly fit into a label. This information is presented in Table 4.

Table 4. Manual Sorting and Manipulation of Real SPEED Into Pose Configuration

Real Image Pose Mode	Image Index Range	Approximate Pose			Rotation	Selected Label for Configuration 5			Selected Label for Configuration 6		
		φ	θ	ψ		Φ	Θ	Ψ	Φ	Θ	Ψ
1	7→100	~135°	<45°	~0°	--	140°	5°	5°	--	--	--
2	105→199	<180°	>270°	~45°	CCW 90°	185°	50°	230°	185°	50°	230°
3	201→299	<180°	>270°	~0°	CW 90°	185°	50°	95°	185°	50°	95°
4	302→399	<180°	~0°	~0°	--	140°	5°	5°	--	--	--
5	404→497	<180°	~-5°	~5°	--	140°	5°	5°	--	--	--
6	503→600	<180°	~-5°	~75°	CW 90°	140°	5°	50°	--	--	--
Number of Included Images						300			105		

4.5.3 Note on Downsampling

It was important that scaling was minimized in usage, or at least used carefully with the imagery in each dataset's pipeline. This is in order to preserve the intrinsic information of imagery as it passes from one stage to another in the pipeline, or in other words, only changing it when intended. If the imagery is reduced in resolution (downsampled), information provided by the higher-resolution image is partially lost by increasing pixel coarseness.

In addition, increasing an image in resolution (upsampling) means that the contents of many pixels must be guessed from few, i.e. false information is being unintentionally created and added to the image. This practice is thus completely avoided in this pipeline, and downsampling is preferred. [33] This is why the resolution-preserving `grabCut()` was utilized, as the images can safely be downsampled once cropped. When image resizing is necessary for deep learning, downsampling is shown to be favorable to upsampling. [34]

An example of the bilinear downsampling technique, utilized in 4.5.1 and 4.5.2, is included for an example synthetic SPEED image in Figure 44. Very minor differences are visible (mainly visible in smoother texture upon downsampling), and no noise appears to be introduced despite the nearly four-fold decrease in width (1200x1200 to 256x256).

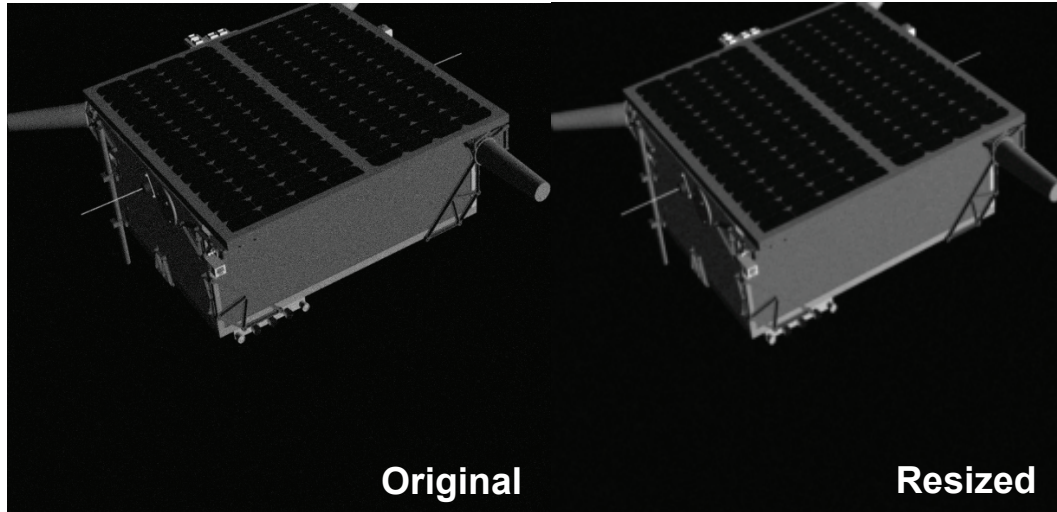


Figure 44. Example of a 1200x1200 SPEED image bilinearly downsampled to 256x256

4.6 Computational Resources

Computation resources sufficient for extensive neural network training were available at USNA through the Advanced Research Computing Support (ARCS) group, as well as the Computer Science Department located in Michaelson Hall. These are ‘the Cluster’ and Tribble/Trouble. These are GPU (Graphics Processing Unit)-enabled workstations, which surpasses those which are CPU-enabled in terms of parallel processing and thus computation speed for parallel-heavy machine learning tasks. For ARCS’ Cluster, tasking is passed via specialized queueing scripts through the Secure Shell (SSH) Protocol over the USNA Intranet. [35,36] For Tribble/Trouble, tasking is assigned directly to one or more of the four computation nodes via SSH over the USNA Intranet.

5 CycleGAN Trials and Outputs

5.1 Using Unmodified ('Vanilla') CycleGAN

All below trials with CycleGAN used the default hyperparameter settings and network architecture prescribed by its originating literature. [22,24]

5.1.1 Foreword

5.1.1.1 Presentation and Style of Analysis

In order to record the visual analysis that was conducted on each set of CycleGAN results, input/output pairings are shown from selected points in the training (epochs) from both R2S and S2R Generators. Many such points are shown, and the analysis points out trends in the visual data as training progressed. Both of the Generators' results are shown to take into consideration the cycle-consistency training loss (Equation [18]), and the coupling effect it has between the Generators (i.e. to observe if convergence in one necessarily results in convergence for the other).

5.1.1.2 Figure Etiquette

All figures used to denote training stages for CycleGAN are organized in two rows as follows.

- > TOP: five random input images from synthetic or real components of a given dataset
- > BOTTOM: the images resulting from inputting the top row into the trained 'synthetic-to-real' (S2R) or 'real-to-synthetic' (R2S) CycleGAN Generator

5.1.2 Initial Trials with Un-Cropped, Un-Sorted SPEED

The synthetic imagery in both the training and test subsets of SPEED included imagery with both space (dark) and Earth backgrounds. Initially, the selected GAN architecture was trained on the conglomerate set, but the GAN was attempting to generalize the background too much. The results can be seen in Figure 45.

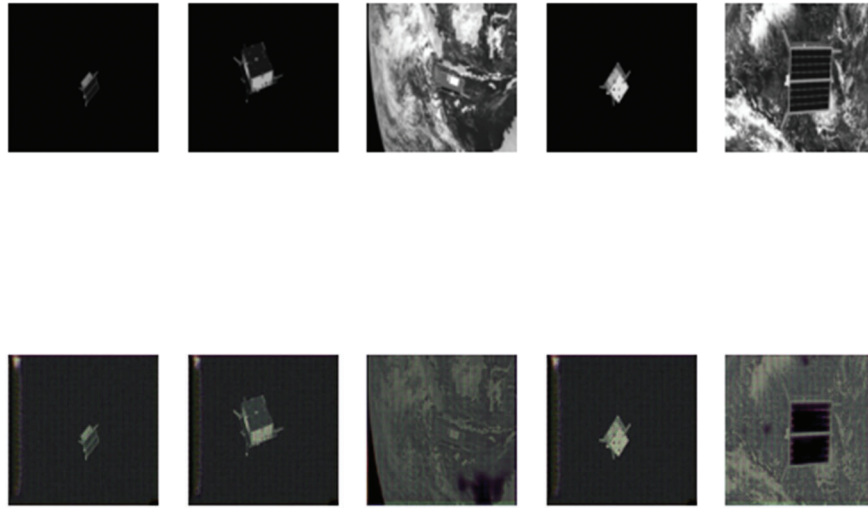


Figure 45. Five examples of S2R, visually poor impact of training with both space and Earth backgrounds, CycleGAN trained on SPEED for 37 epochs

Due to the nature of these results, the synthetic imagery of SPEED was sorted based on background. In addition, it was discovered that the real imagery in SPEED did not include any Earth backgrounds. This resulted in 'Sorted SPEED'. The rationale behind this is discussed around Figure 40; this is elaborated in 3.2.

Using Sorted SPEED, quantity of synthetic images (7,048) still far exceeded that of real images (300); thus, the results of CycleGAN training strongly reflected this over-familiarity with the same few real images (overtraining), as mode collapse occurred. [25] This can be seen in Figure 46.

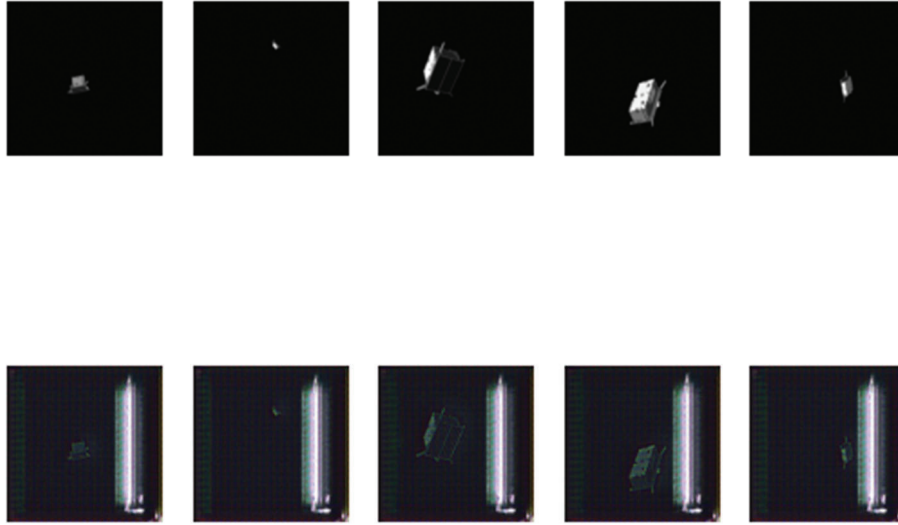


Figure 46. Five examples of S2R, visually poor impact of training with unbalanced quantities of real and synthetic imagery, CycleGAN trained on Sorted SPEED for 38 epochs

5.1.3 Training with Sort-Pare, Un-Cropped SPEED

The network initialized random noise and artifacts, with very noticeable ‘structural’ aberrations (or geometric disturbances in the image object) in epoch 1 (Figure 47 and Figure 48). This noise stayed further and lead to output that intermittently became more or less distinct (in smaller local satellite details) with signs of mode collapse occurring during the points at which output was notably ‘less distinct’ (e.g. epochs 8, 12, or 15; seen in Figure 51 and Figure 52, Figure 53 and Figure 54, and Figure 55 and Figure 56 respectively). CycleGAN converges to and does not deviate greatly from the appropriate color scheme from about epoch 6 (Figure 49 and Figure 50) onwards.

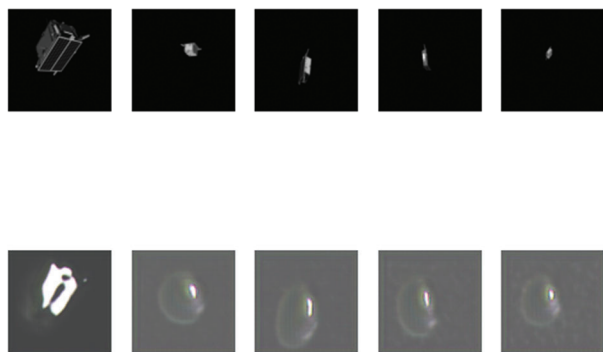


Figure 47. Five examples of S2R, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 1 epoch

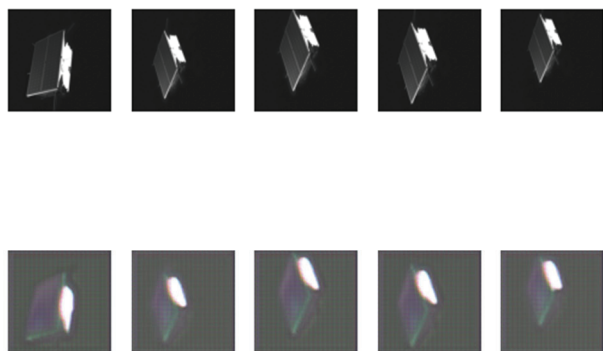


Figure 48. Five examples of R2S, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 1 epoch

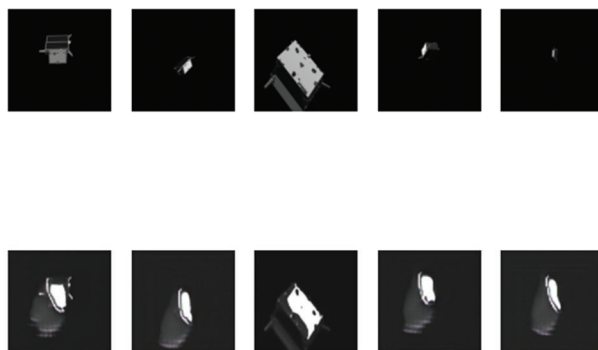


Figure 49. Five examples of S2R, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 6 epochs

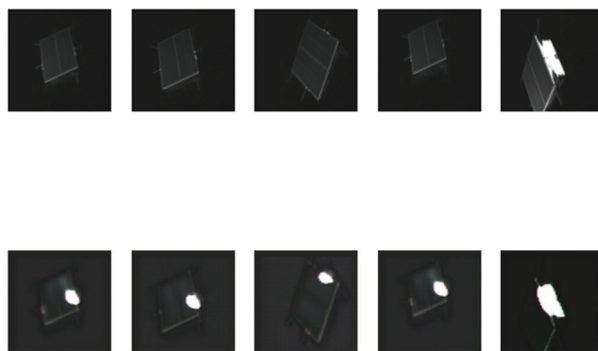


Figure 50. Five examples of R2S, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 6 epochs

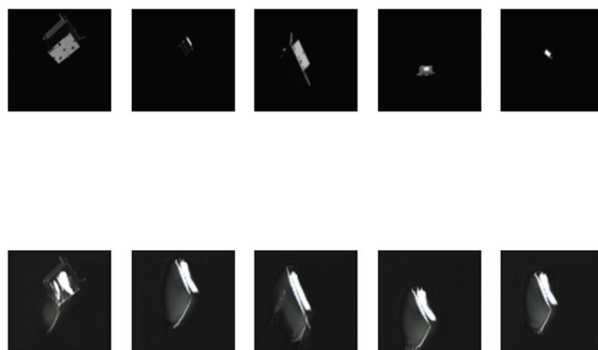


Figure 51. Five examples of S2R, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 8 epochs

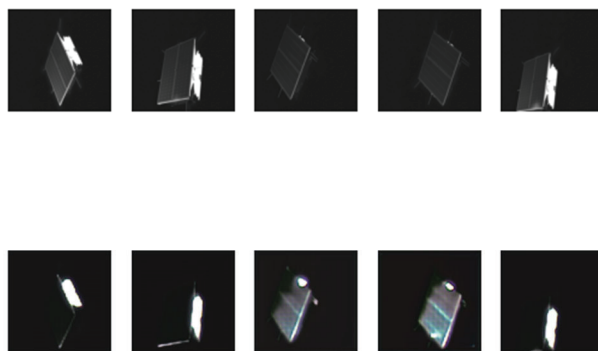


Figure 52. Five examples of R2S, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 8 epochs

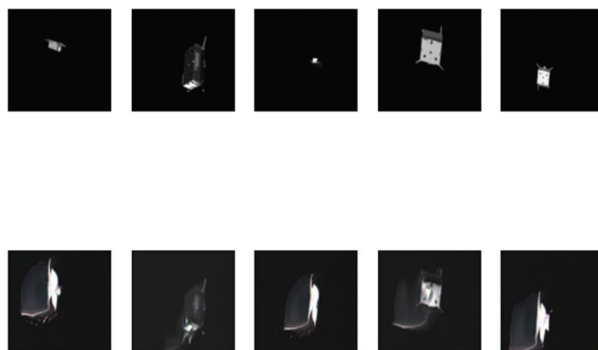


Figure 53. Five examples of S2R, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 12 epochs

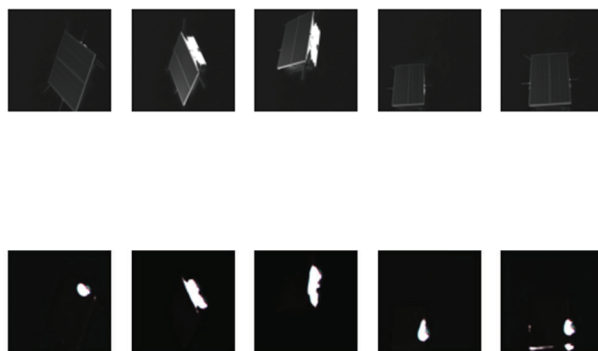


Figure 54. Five examples of R2S, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 12 epochs

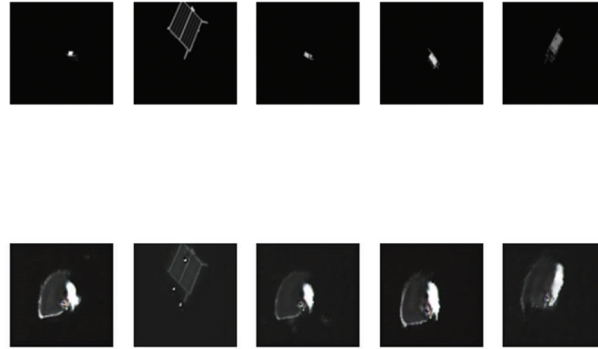


Figure 55. Five examples of S2R, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 15 epochs

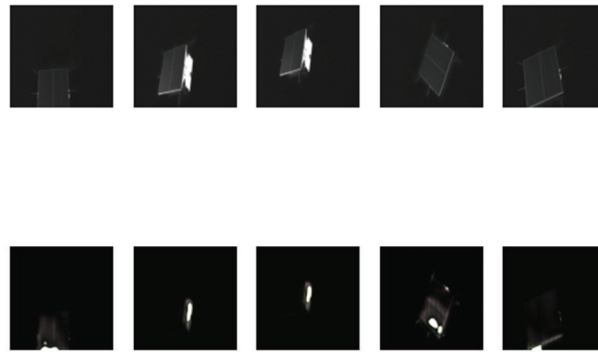


Figure 56. Five examples of R2S, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 15 epochs

In between the intermittent noise discussed above, the network converges briefly to visually favorable output imagery at epochs 23, 45, 46, 71, and 100 (Figure 57, Figure 59, Figure 60, Figure 63, and Figure 71). Such convergence is notably regular.

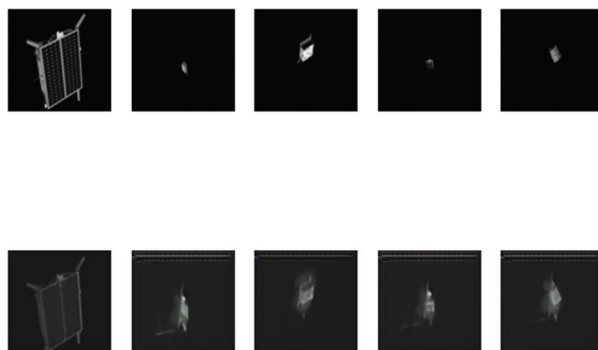


Figure 57. Five examples of S2R, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 23 epochs

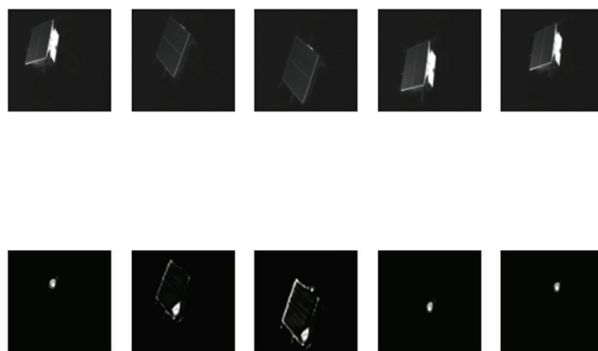


Figure 58. Five examples of R2S, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 23 epochs

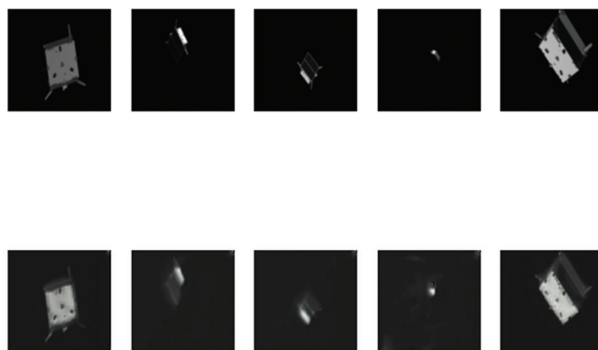


Figure 59. Five examples of S2R, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 45 epochs

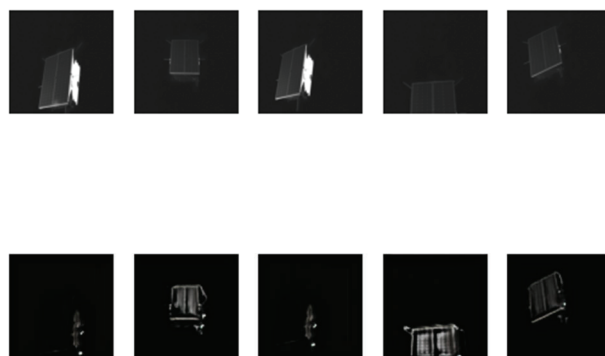


Figure 60. Five examples of R2S, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 45 epochs

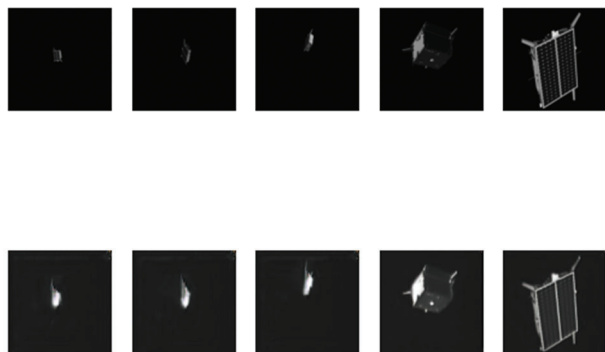


Figure 61. Five examples of S2R, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 46 epochs

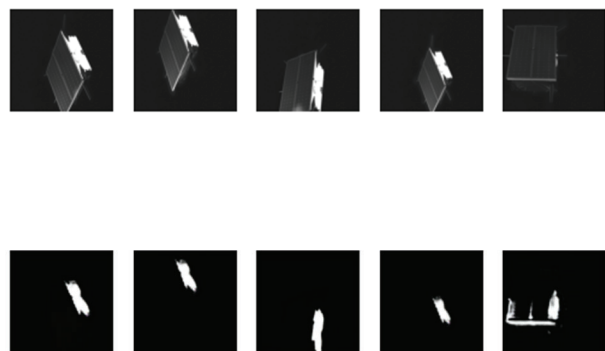


Figure 62. Five examples of R2S, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 46 epochs

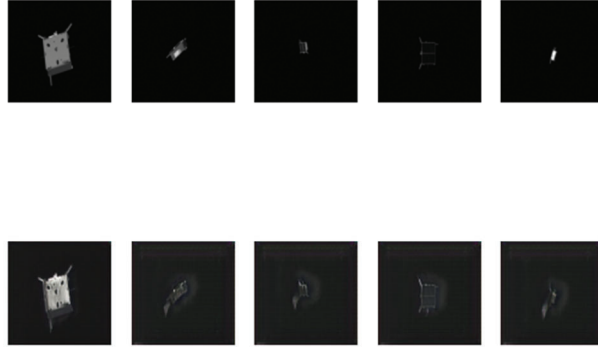


Figure 63. Five examples of S2R, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 71 epochs

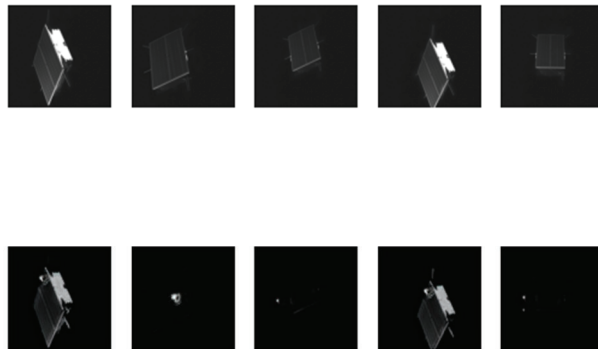


Figure 64. Five examples of R2S, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 71 epochs

Note that significant structural aberrations (and RGB alteration to the grayscale color scheme for R2S only) begin at epoch 74 (Figure 65 and Figure 66); as results approach their best at epoch 100 (view S2R progression from Figure 67 to Figure 71), note that structural aberrations and color alterations disappear gradually.

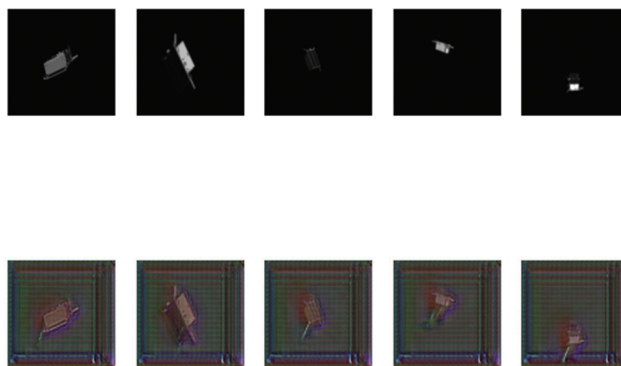


Figure 65. Five examples of S2R, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 74 epochs

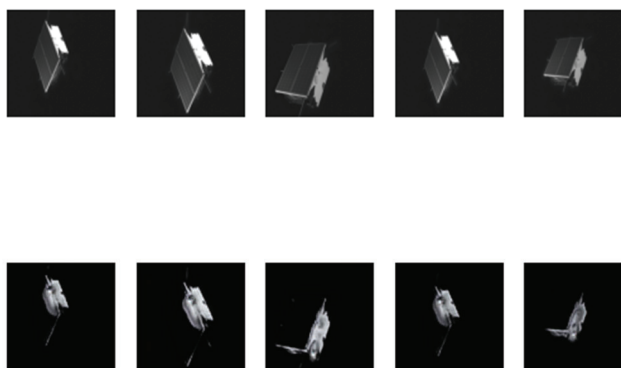


Figure 66. Five examples of R2S, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 74 epochs

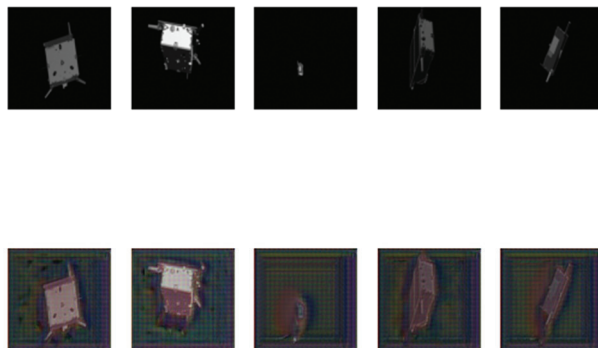


Figure 67. Five examples of S2R, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 76 epochs

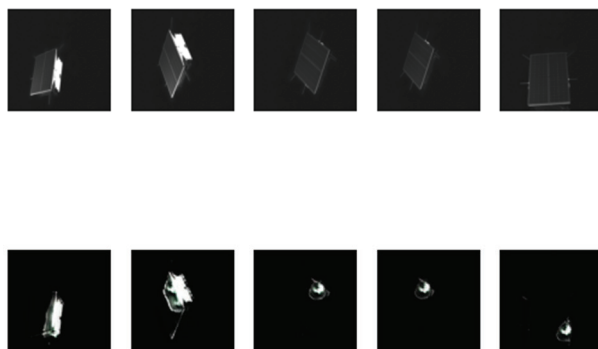


Figure 68. Five examples of R2S, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 76 epochs

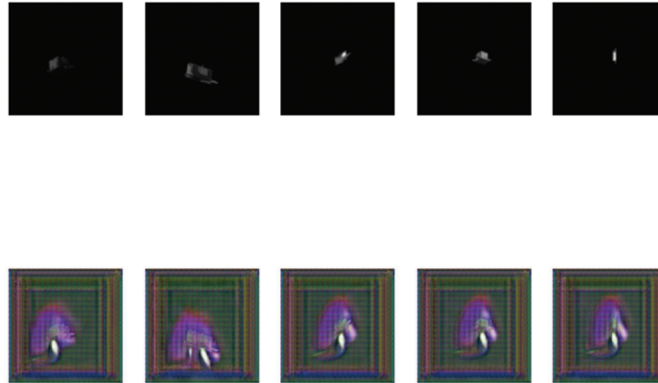


Figure 69. Five examples of S2R, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 84 epochs

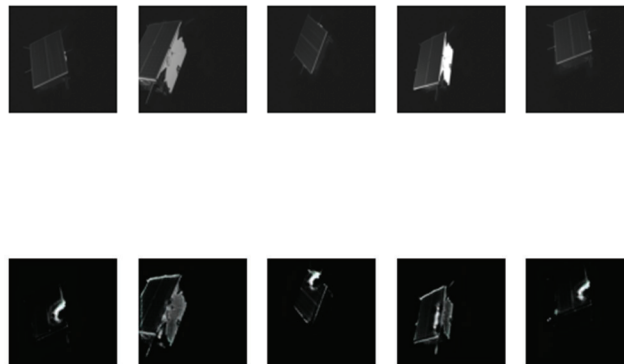


Figure 70. Five examples of R2S, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 84 epochs

The lack of apparent widespread background noise and adherence to the visual characteristics of real imagery (higher contrast, brighter background; see Figure 30) in epoch 100 make them the visually most favorable results in the S2R Generator throughout this training run (Figure 71). Despite this visual evidence

of success, the apparent repeated artifacts indicative of minor mode collapse in the corresponding R2S Generator serve as a warning flag (Figure 72). Ultimately, while not expected to be completely in-sync, the two Generators are ultimately coupled and visually poor results in one should be expected to inspire rapid change in the other via the cycle-consistency loss term (Equation [18]). At the same time, note that mode collapse is a form of convergence, and perhaps this particular instance in R2S is visually minor enough to promote stability in S2R.

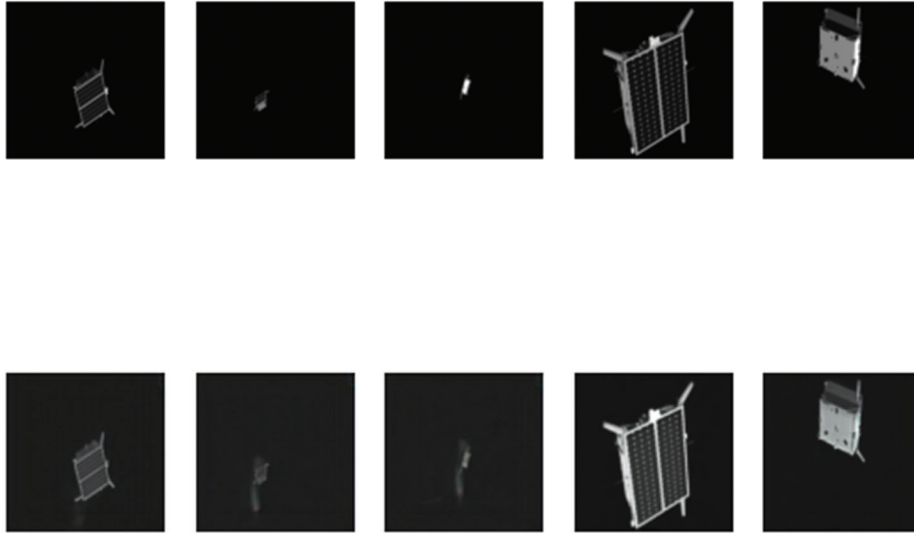


Figure 71. Five examples of S2R, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 100 epochs

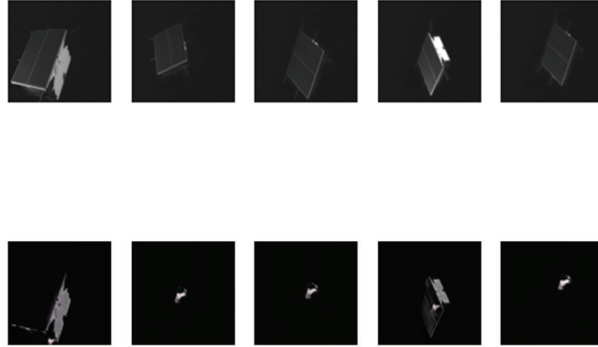


Figure 72. Five examples of R2S, training with both space and Earth backgrounds, CycleGAN trained on SPEED for 100 epochs

5.1.4 Training More Rigorously with POC

Note that, in order to save time and check output more frequently, as well as to provide a better comparison epoch-by-epoch to the SPEED training runs, an ‘epoch’ was declared to be 1200 (600 images per dataset) images despite the total of 54,000 images in POC.

CycleGAN adjusted quickly to the appropriate color scheme from its random initial noise. In this particular run, it did so between epoch 1 and epoch 2 (compare Figure 73 and Figure 74 with Figure 75 and Figure 76).

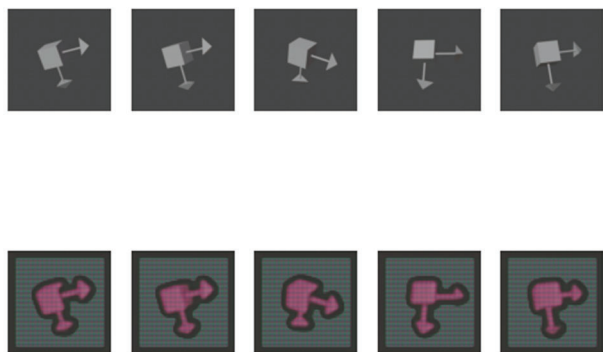


Figure 73. Five examples of S2R, CycleGAN trained on POC for 1 epoch

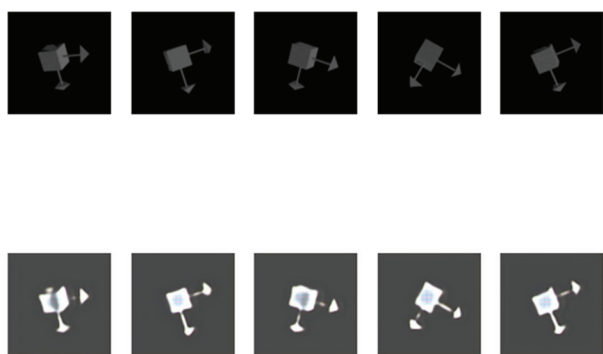


Figure 74. Five examples of R2S, CycleGAN trained on POC for 1 epochs

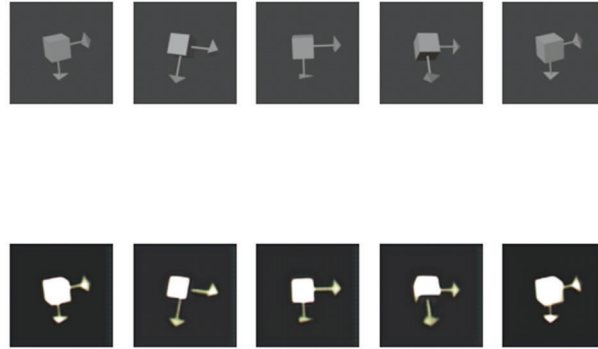


Figure 75. Five examples of S2R, CycleGAN trained on POC for 2 epochs

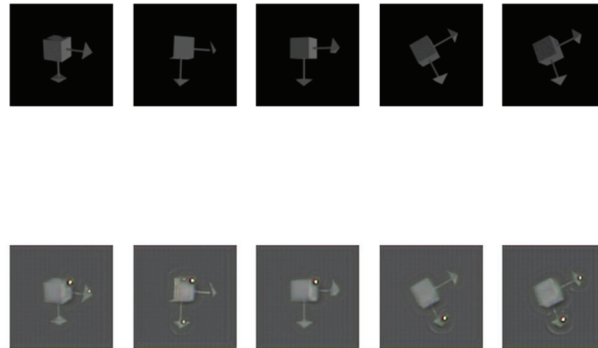


Figure 76. Five examples of R2S, CycleGAN trained on POC for 2 epochs

In general across its training runs, CycleGAN was observed to progress from the large initial changes of the first few epochs to making finer adjustments and subtler stylistic changes. This is evident in this particular training run by the change in pace and scale from the fast, 1-epoch adjustments made by CycleGAN from its initially random output, to the stylistic changes of contrast and shading on the satellite

object that are observable progressing from epoch 2 to epoch 7 (Figure 75 and Figure 76 to Figure 79 and Figure 80).

Particularly evident in this run, the CycleGAN tends to hold onto swaths of high-intensity bright pixels, and over the course of training, slowly shrink and dissipate them from the main satellite structure to the background of the image. This can be seen in the progression from epochs 4-10 in both S2R and R2S outputs (Figure 77 and Figure 78 to Figure 83 and Figure 84). This was presumed to be a result of the normalization scheme that is used within CycleGAN, and is well-known in literature (see the discussion on normalization in [37]).



Figure 77. Five examples of S2R, CycleGAN trained on POC for 4 epochs

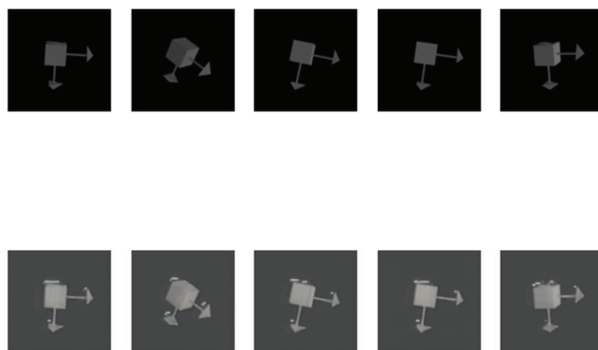


Figure 78. Five examples of R2S, CycleGAN trained on POC for 4 epochs

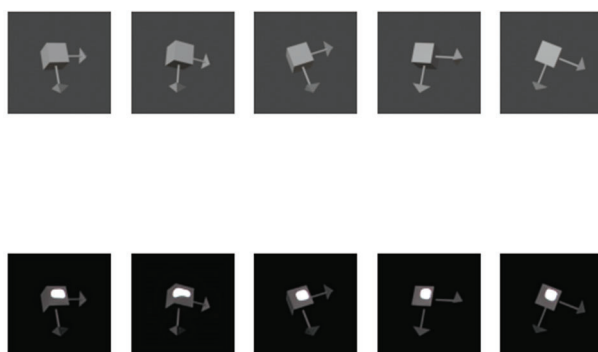


Figure 79. Five examples of S2R, CycleGAN trained on POC for 7 epochs

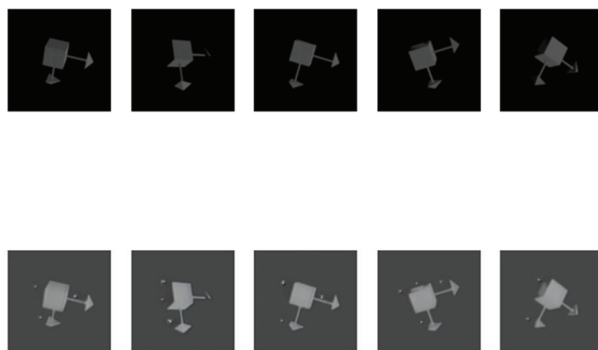


Figure 80. Five examples of R2S, CycleGAN trained on POC for 7 epochs



Figure 81. Five examples of S2R, CycleGAN trained on POC for 9 epochs

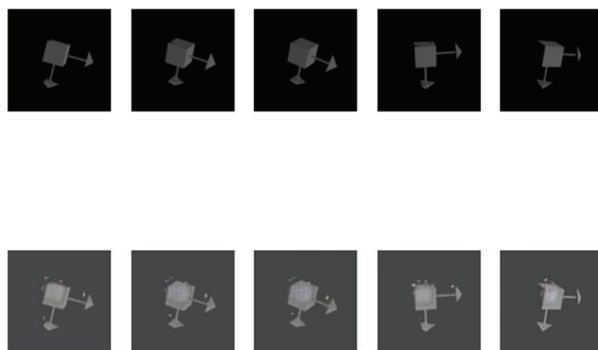


Figure 82. Five examples of R2S, CycleGAN trained on POC for 9 epochs

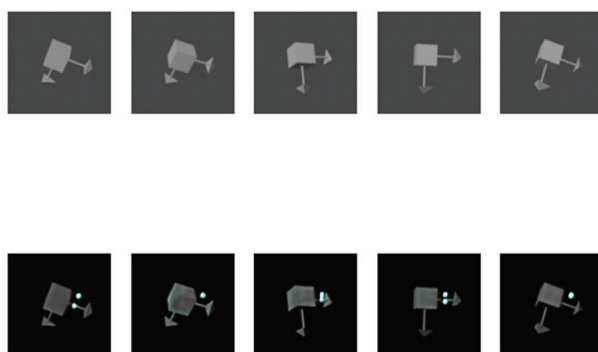


Figure 83. Five examples of S2R, CycleGAN trained on POC for 10 epochs

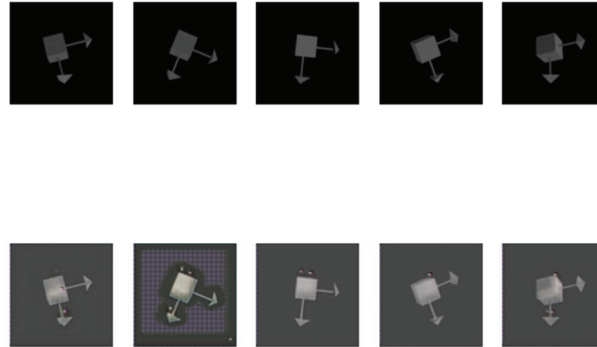


Figure 84. Five examples of R2S, CycleGAN trained on POC for 10 epochs

At the point where the focused points of bright pixels dissipate into pixels of noise in the image background (most notable in epoch 10 of R2S, in the second output image from the left; see Figure 84), CycleGAN converged to visually good results in both S2R and R2S in epoch 12 (see Figure 85 and Figure 86, respectively). Note that CycleGAN maintained both the structure and majority of the shading in the input images in their respective outputs. Perhaps this is due to advanced learning done by CycleGAN, but seeing as this occurred at epoch 12, perhaps this is simply trivial change in background color resulting from the small amount of training exposure gained by CycleGAN so far. Note also the grid-like noise that appears in the real-to-synthetic transformation, suggesting also a lack of training.

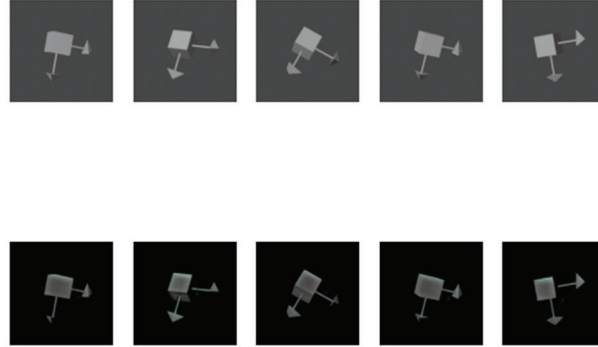


Figure 85. Five examples of S2R, CycleGAN trained on POC for 12 epochs



Figure 86. Five examples of R2S, CycleGAN trained on POC for 12 epochs

Past epoch 12 in its training, CycleGAN started to add structural aberrations (i.e. blurring borders and corners) and artifacts (as minor as false shading that's seen consistently across different images) into the image. This is noticeable intermittently in the progression from epochs 14-67 (Figure 93 and Figure 94 to Figure 97 and Figure 98), with noticeable structural aberrations appearing in R2S epoch 14, and disappearing by the convergence of both S2R and R2S at epoch 24 (see Figure 89 and Figure 90). This pattern repeats itself, as from this convergence, aberrations appear again (notably at epoch 41; see Figure 91 and Figure 92) only to disappear again from epochs 45-52 (Figure 93 and Figure 94 to Figure 95 and Figure 96).

Presumably, this would repeat again, as from the strong convergence in both S2R and R2S at epoch 52, aberrations dominate at epoch 67 (see Figure 97 and Figure 98). This appears to be a bimodal stability, and is discussed more generally in section 6.1.3.

Note again for this training run that convergence in S2R is typically suggestive of a convergence in R2S as well, indicating that for this training run the two Generators are strongly coupled.

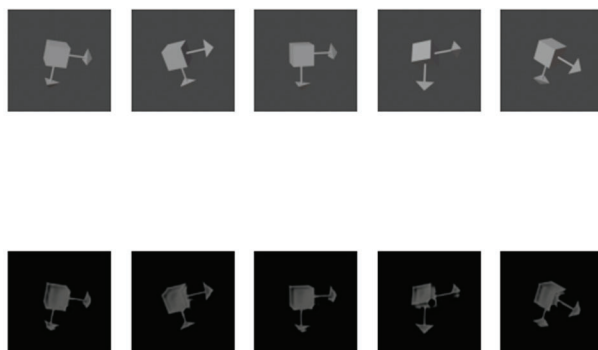


Figure 87. Five examples of S2R, CycleGAN trained on POC for 14 epochs



Figure 88. Five examples of R2S, CycleGAN trained on POC for 14 epochs

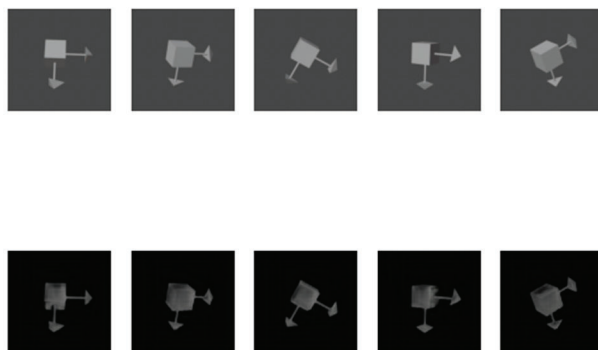


Figure 89. Five examples of S2R, CycleGAN trained on POC for 24 epochs



Figure 90. Five examples of R2S, CycleGAN trained on POC for 24 epochs

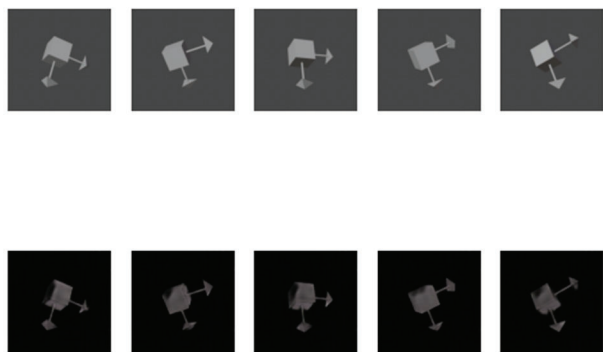


Figure 91. Five examples of S2R, CycleGAN trained on POC for 41 epochs

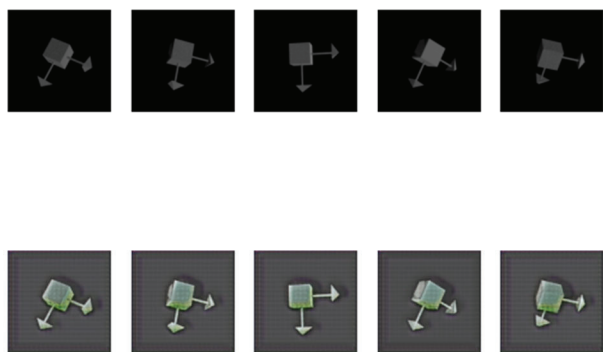


Figure 92. Five examples of R2S, CycleGAN trained on POC for 41 epochs

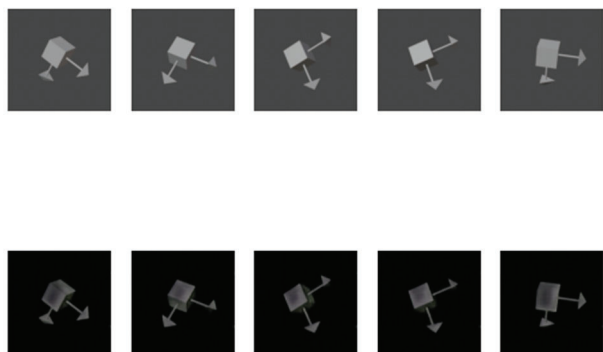


Figure 93. Five examples of S2R, CycleGAN trained on POC for 45 epochs

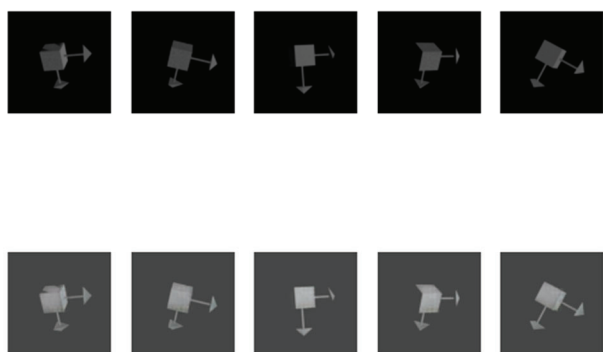


Figure 94. Five examples of R2S, CycleGAN trained on POC for 45 epochs

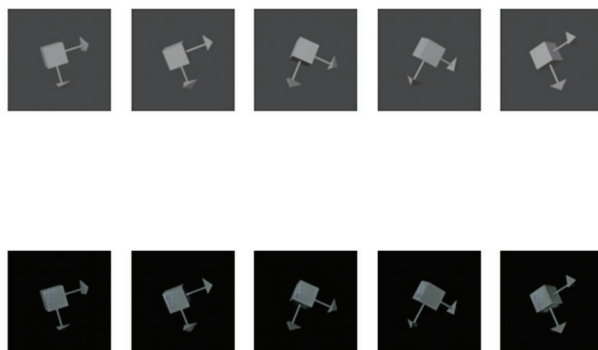


Figure 95. Five examples of S2R, CycleGAN trained on POC for 52 epochs



Figure 96. Five examples of R2S, CycleGAN trained on POC for 52 epochs

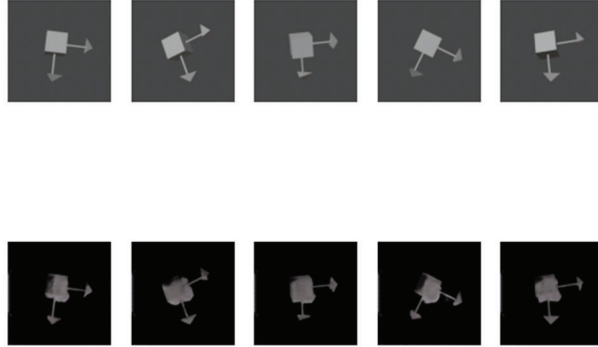


Figure 97. Five examples of S2R, CycleGAN trained on POC for 67 epochs



Figure 98. Five examples of R2S, CycleGAN trained on POC for 67 epochs

5.1.5 Training with POC2

The following section is broken into separate sections for S2R and R2S. Note that, as with the POC training run, an ‘epoch’ was declared to be 1200 (600 images per dataset) images despite the total of 54,000 images in POC2.

For S2R, initialization noise present in epoch 1 (Figure 99) was mostly eliminated by epoch 8 (Figure 100). Past this, the remainder of this training run sees very little variation in CycleGAN’s S2R output, and thus it is very difficult to provide any deeper visual analysis. The majority of epochs are associated with visually favorable output, examples including epochs 11, 49, 57 and 78 (Figure 101, Figure 105, Figure 106, and Figure 107). Stylistic noise is introduced occasionally in the form of slight color scheme disturbances (epochs 13, 28; Figure 102 and Figure 103), and at its most a grid-like ambient noise (epoch 29; Figure 104). Structural aberration doesn’t appear to occur, even late in training. Mode collapse doesn’t occur noticeably.

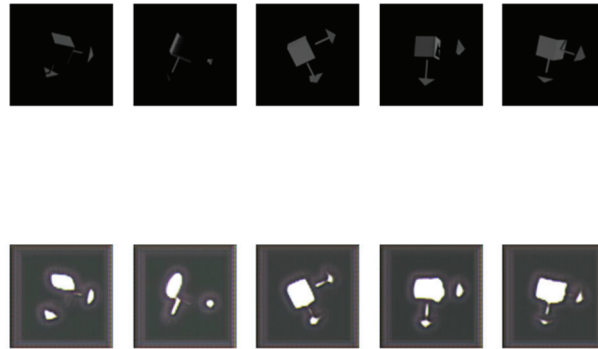


Figure 99. Five examples of S2R, CycleGAN trained on POC2 for 1 epoch



Figure 100. Five examples of S2R, CycleGAN trained on POC2 for 8 epochs



Figure 101. Five examples of S2R, CycleGAN trained on POC2 for 11 epochs

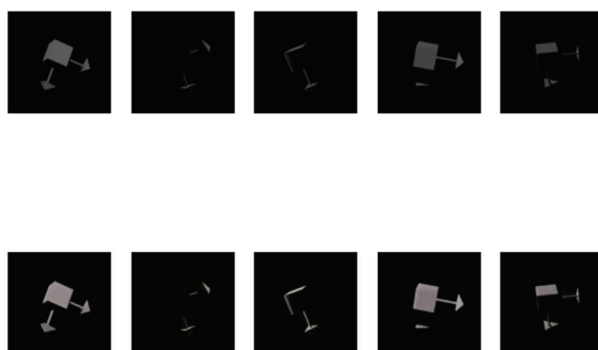


Figure 102. Five examples of S2R, CycleGAN trained on POC2 for 13 epochs

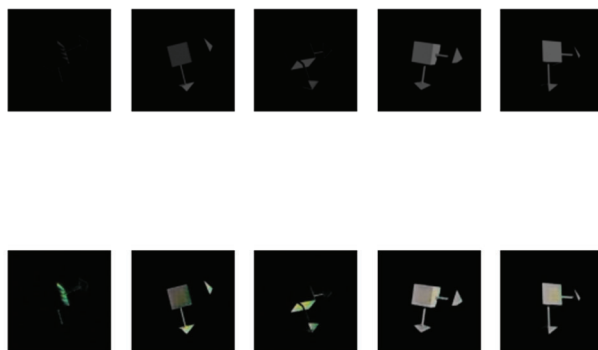


Figure 103. Five examples of S2R, CycleGAN trained on POC2 for 28 epochs

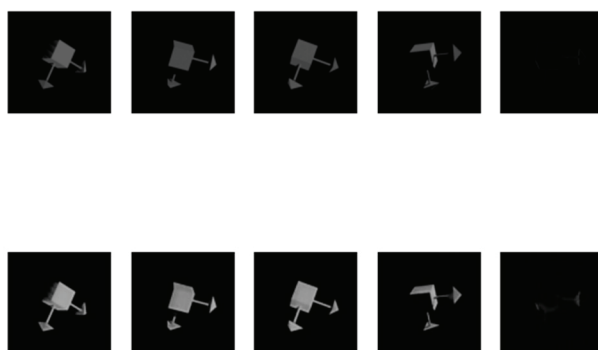


Figure 104. Five examples of S2R, CycleGAN trained on POC2 for 29 epochs



Figure 105. Five examples of S2R, CycleGAN trained on POC2 for 49 epochs



Figure 106. Five examples of S2R, CycleGAN trained on POC2 for 57 epochs



Figure 107. Five examples of S2R, CycleGAN trained on POC2 for 78 epochs

Upon review of the first few epochs for R2S, the bright pixel groups in epoch 1 (Figure 108) appear to shrink to brighter parts of the satellite object as in epoch 6 (Figure 109) and 9 (Figure 110). These spots migrate from the object body to the background, and continue to focus into many points as in epoch 13 (Figure 111). These points dissipate entirely by epoch 19 (Figure 112), and past this, the remainder of this training run sees very little variation in CycleGAN's R2S output (as with S2R).

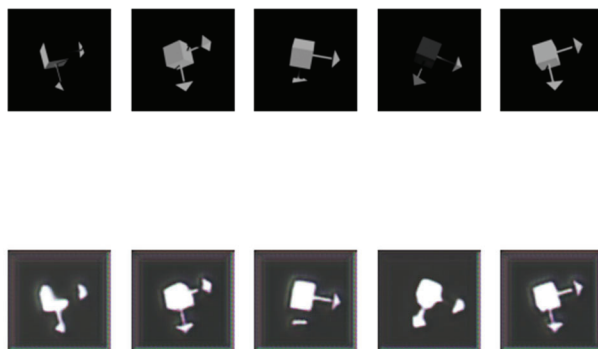


Figure 108. Five examples of R2S, CycleGAN trained on POC2 for 1 epochs

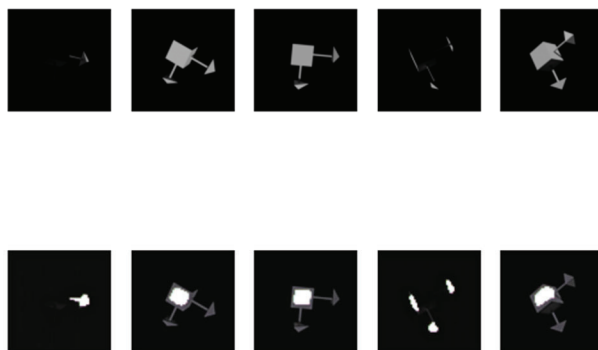


Figure 109. Five examples of R2S, CycleGAN trained on POC2 for 6 epochs

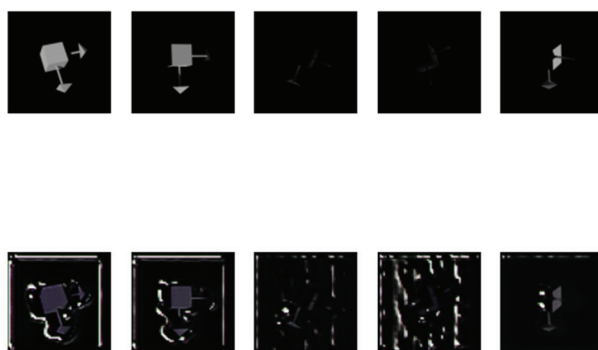


Figure 110. Five examples of R2S, CycleGAN trained on POC2 for 9 epochs

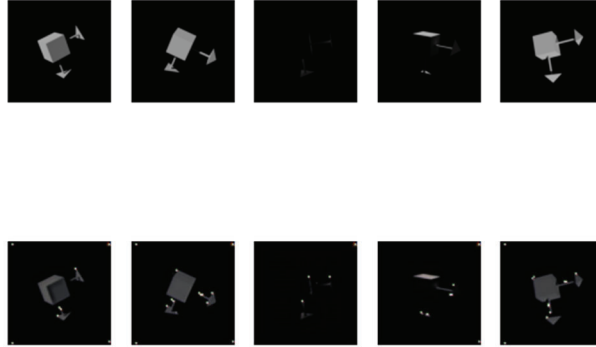


Figure 111. Five examples of R2S, CycleGAN trained on POC2 for 13 epochs

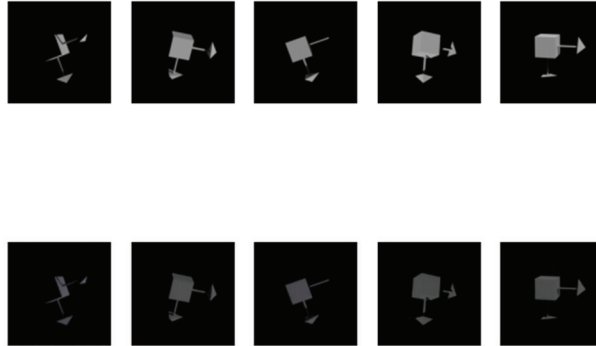


Figure 112. Five examples of R2S, CycleGAN trained on POC2 for 19 epochs

The lack of output variation is, in general, most likely due to the very subtle margin of visual differences between the two datasets.

5.1.6 *First Cropped SPEED Trials*

The first epoch of training eliminated most of the color disparity between image channels already (i.e. conformed from color splashes to the desired grayscale; see Figure 113 and Figure 114). Bright ambient

background noise is evident in S2R output at epoch 1 (see Figure 113), which appears to be CycleGAN’s initially exaggerated attempt to mimic the brighter contrast and background when transitioning from synthetic to real SPEED (see Figure 41 for reference). While this bright background noise was mostly toned down to an appropriate level by epoch 12 (Figure 117), in some epochs blobs of increased brightness still clung to the foreground and surrounded the satellite object, causing blurring of object detail (see epoch 33; Figure 127).

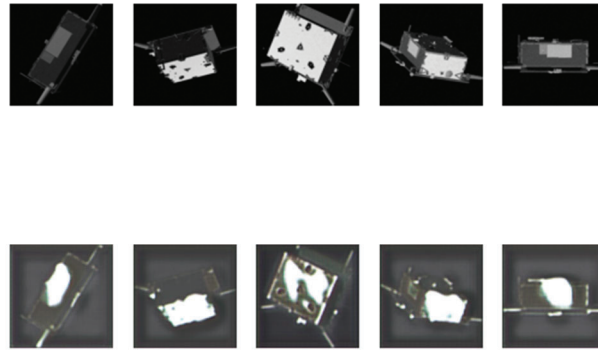


Figure 113. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED for 1 epoch

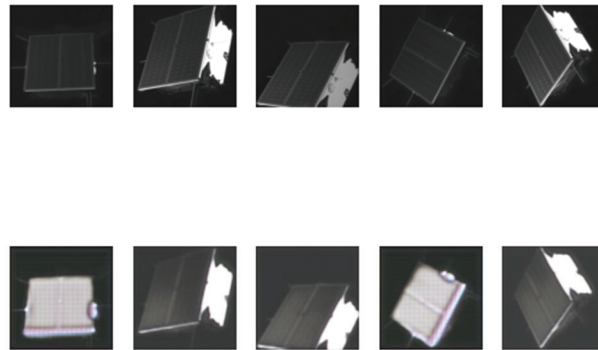


Figure 114. Five examples of R2S, CycleGAN trained on cropped sort-pare SPEED for 1 epoch

A largely binary color scheme (dark gray or bright white making up the majority of the image, with little color gradation in-between) appears to have dominated the majority of this training run. Additionally,

details of the satellite are blurred by this lack of color distinction, in addition to the discussed bright background noise around the satellite object. This can be seen notably in epochs 6 (see Figure 115 and Figure 116), 12 (see Figure 117 and Figure 118), and 15 (see Figure 119 and Figure 120).

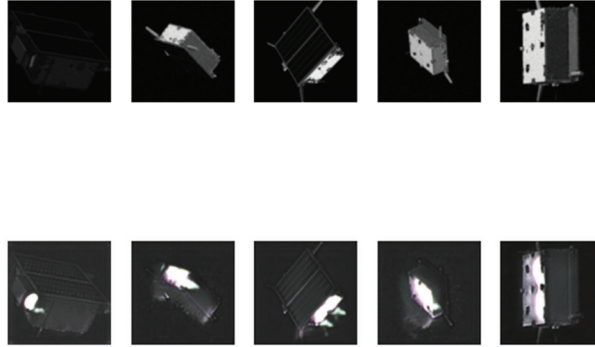


Figure 115. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED for 6 epochs

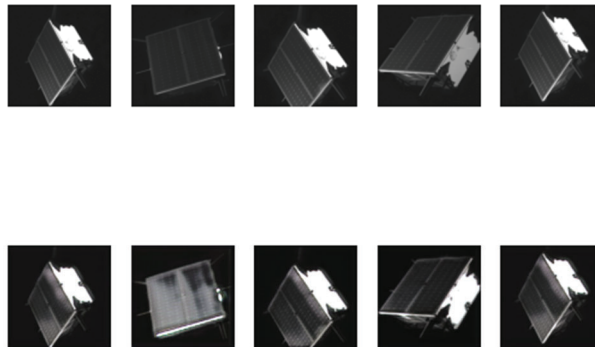


Figure 116. Five examples of R2S, CycleGAN trained on cropped sort-pare SPEED for 6 epochs

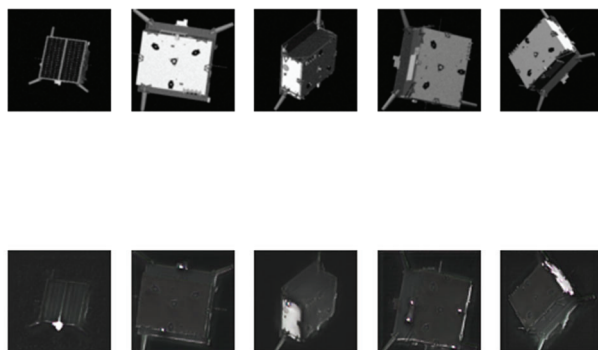


Figure 117. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED for 12 epochs

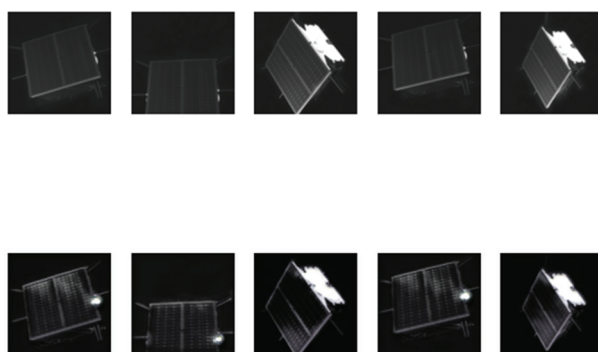


Figure 118. Five examples of R2S, CycleGAN trained on cropped sort-pare SPEED for 12 epochs

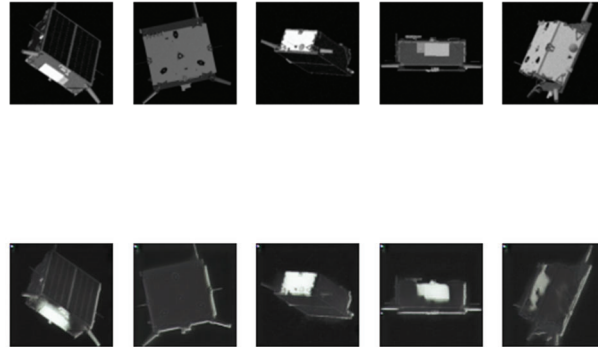


Figure 119. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED for 15 epochs

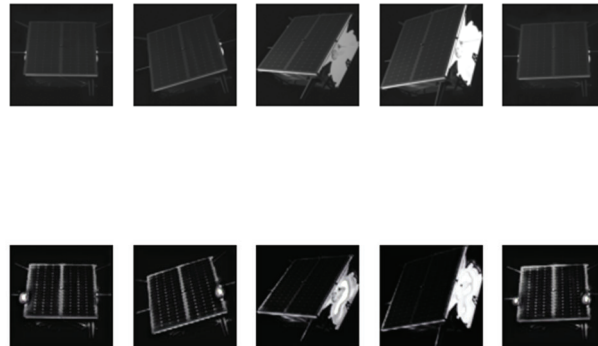


Figure 120. Five examples of R2S, CycleGAN trained on cropped sort-pare SPEED for 15 epochs

Past epoch 15 (Figure 119), CycleGAN appeared to output in S2R either very dark (epoch 22; Figure 121) or very warped (epoch 26; Figure 125). In fact, it seems to regularly vacillate between the two types of images: the dark output images at epoch 22 became more strongly warped through epoch 24 (Figure 123) and resulted in the type of images at epoch 26, which transitioned back to images with dark and largely unwarped satellite objects at epoch 33 (Figure 127). Note that, while there is a lack of significant brightness in the darker satellite object images, their details were still visible enough to determine that they were not nearly as warped as those brighter images with structural aberrations.

A parallel trend appears to exist in the R2S data, but the ‘dark’ images were not nearly as deprived of brightness in their details: the brighter, warped images at epoch 22 (Figure 122) became less warped and darker at epoch 24 (Figure 124) and resulted in the darker, unwarped images at epoch 26 (Figure 126), which was similar to the output at epoch 33 (Figure 128). This parallel vacillation suggests somewhat of an inverted coupling between the Generators.

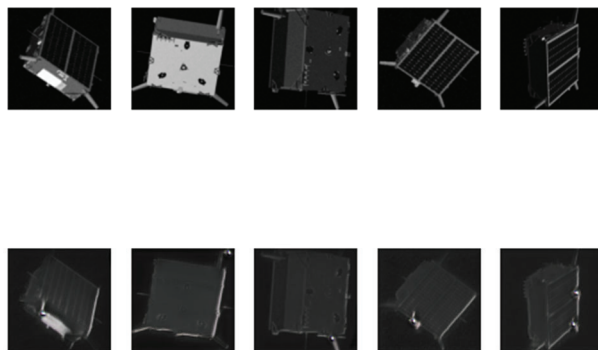


Figure 121. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED for 22 epochs

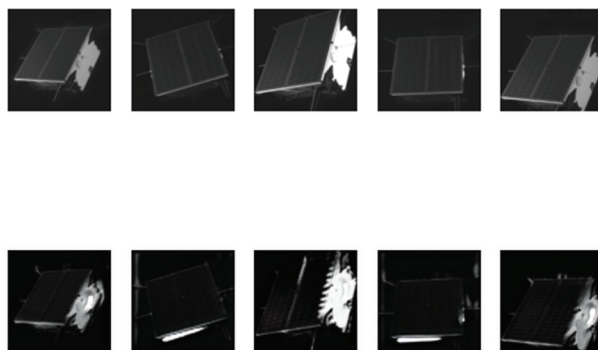


Figure 122. Five examples of R2S, CycleGAN trained on cropped sort-pare SPEED for 22 epochs

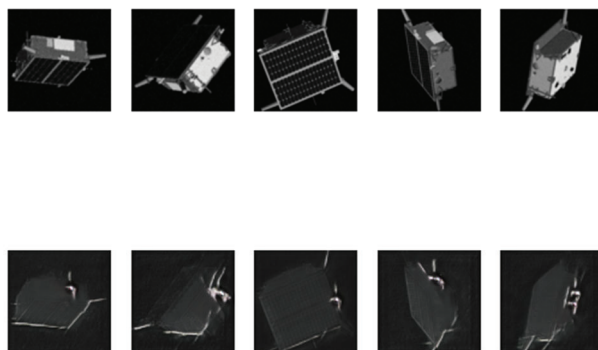


Figure 123. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED for 24 epochs

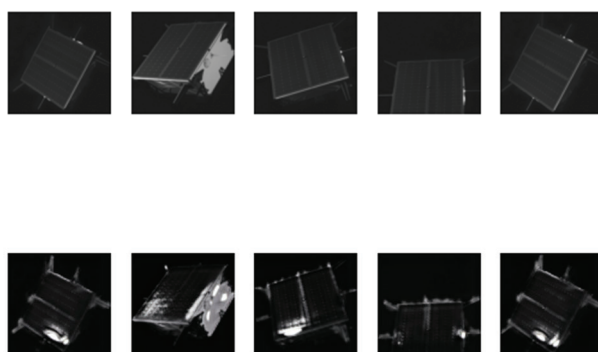


Figure 124. Five examples of R2S, CycleGAN trained on cropped sort-pare SPEED for 24 epochs

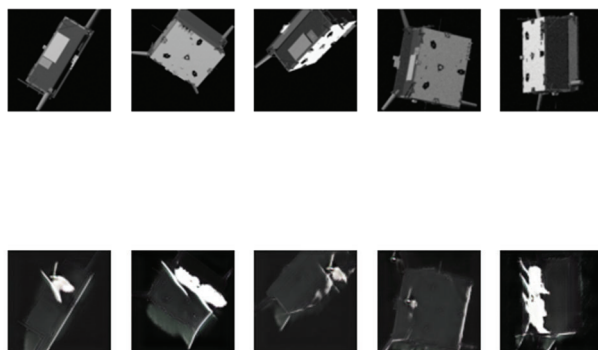


Figure 125. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED for 26 epochs

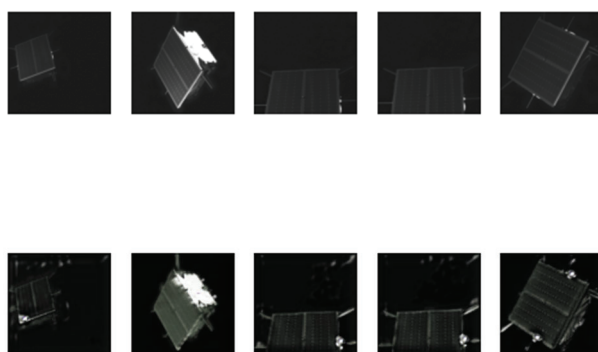


Figure 126. Five examples of R2S, CycleGAN trained on cropped sort-pare SPEED for 26 epochs

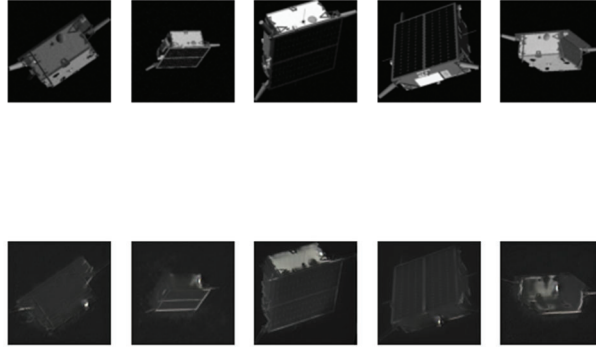


Figure 127. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED for 33 epochs

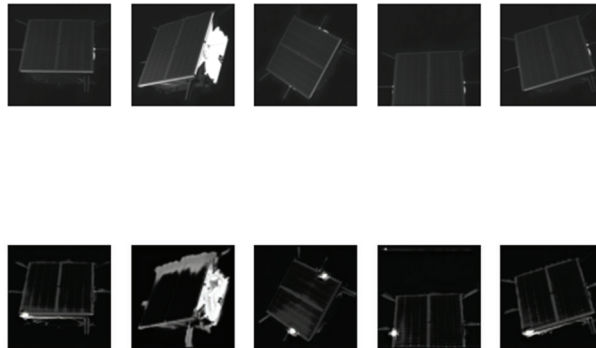


Figure 128. Five examples of R2S, CycleGAN trained on cropped sort-pare SPEED for 33 epochs

Past epoch 50 (Figure 129 and Figure 130), this training run had few visually favorable epochs and suggests a continuation of the aforementioned inversely coupled vacillation pattern in S2R (dark images at epoch 50/Figure 129 transitioned to brighter and warped ones through epoch 57/Figure 131 and to epoch 58/Figure 133) and in R2S (strongly warped images at epoch 50/Figure 130 transitioned to darker and unwarped ones through epoch 57/Figure 132 and to epoch 58/Figure 134). Oddly enough, this is somewhat reminiscent of the bimodal stability discussed in the POC training in 5.1.4, but instead of converging to visually favorable results, CycleGAN is collapsing to a very dark mode.

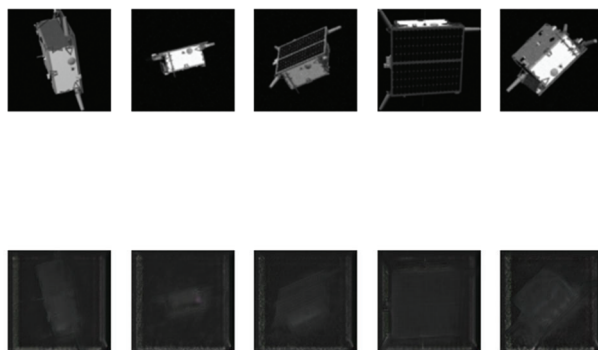


Figure 129. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED for 50 epochs

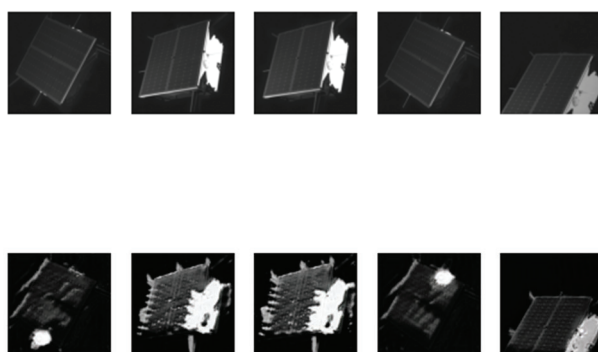


Figure 130. Five examples of R2S, CycleGAN trained on cropped sort-pare SPEED for 50 epochs

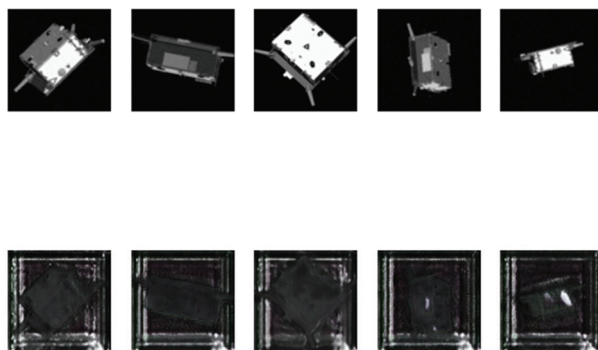


Figure 131. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED for 57 epochs

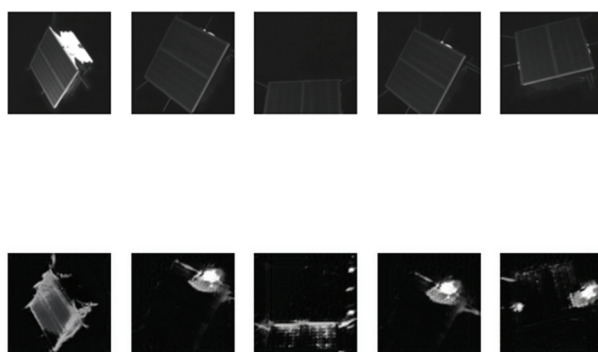


Figure 132. Five examples of R2S, CycleGAN trained on cropped sort-pare SPEED for 57 epochs

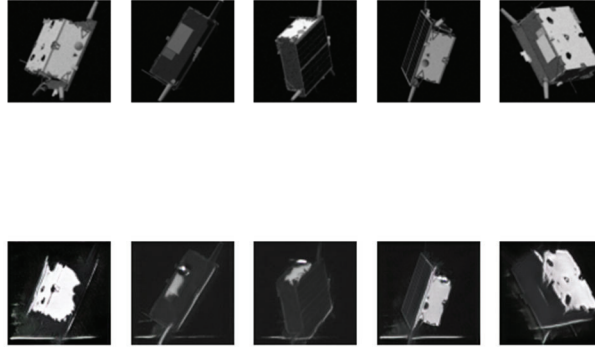


Figure 133. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED for 58 epochs

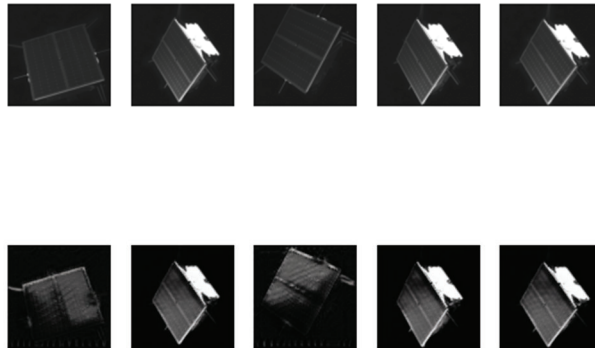


Figure 134. Five examples of R2S, CycleGAN trained on cropped sort-pare SPEED for 58 epochs

5.1.7 Modified Approach to Cropped SPEED (Data Augmentation)

This training run started largely in the same manner as the previous. The first epoch of training conformed the output to the desired grayscale color scale (see Figure 135). While the early S2R output resembles the binary color scheme seen in the previous training run, there is significantly more detail brought out in S2R by color gradation throughout training -- this is best observable when comparing the previous trial's early epochs with the current's epochs 2 and 5 (see Figure 136 and Figure 137). Similarly, the initial

bright ambient background noise in the S2R output was suppressed by epoch 8 (Figure 138). This allowed for more detail acuity to be brought out in the satellite object early in training.

There is an interesting glare-like artifact visible in both S2R (as a bright spot near the root of the satellite's top antenna). This is visible early at epochs 8 and 14 (Figure 138, Figure 139) and dims appreciably afterwards. It is unknown what exactly in the data causes it, and is presumably a focused residue from CycleGAN's early stochastic noise.

In strong contrast with the previous training run, CycleGAN appears to converge to visually favorable results in S2R at epoch 14 (Figure 139). This is supported by the uniform background color, absence of noise, high contrast in color scheme, and acute, unblurry details in the satellite object. Following this convergence, CycleGAN appears to transition to the kind of noisy imagery seen in the previous training run; images at epoch 18 (Figure 140) are dark and somewhat warped. Note that the ambient background noise is prevalent in these images.

Similarly to the past training run, CycleGAN appeared to regularly vacillate between convergence and noisy images in S2R: an apparent convergence appeared to be at epoch 36 (Figure 144), which was blurred by increasing background noise through epoch 47 (Figure 148), which again appeared to converge at epoch 50 (Figure 150). Note, however, that the quality of this convergence appears to be blurrier as training progresses, which is perhaps due to slow overtraining.

Following epoch 50, overtraining is more and more supported by the fact that images are blurrier due to ambient background noise, less acute in detail due to a return to binary color scheme, and more structural aberration and warping. This materializes as soon as epoch 52 (Figure 152), and persists through epochs 76 and 82 (Figure 153 and Figure 154).

As with the previous training runs, there appears to be a definite correlation between the visual quality of the results from R2S with those of S2R. Higher levels of detail and lower noise are present at epochs 18, 36 and 47 (Figure 141, Figure 147, and Figure 149), with visually poorer results appearing to dominate in between at epochs 23 and 34 (Figure 143 and Figure 145; note that the R2S convergences are slightly offset from those of S2R, but in general are close to the same points in training).

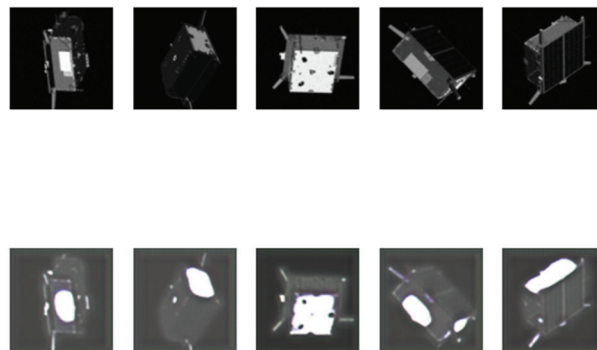


Figure 135. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 1 epoch

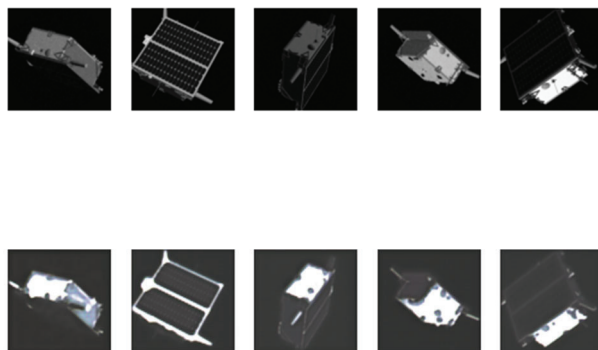


Figure 136. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 2 epochs

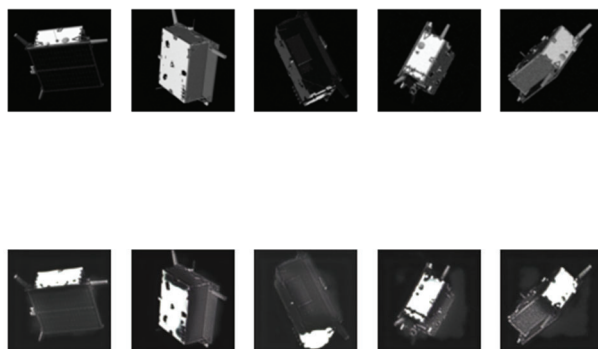


Figure 137. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 5 epochs

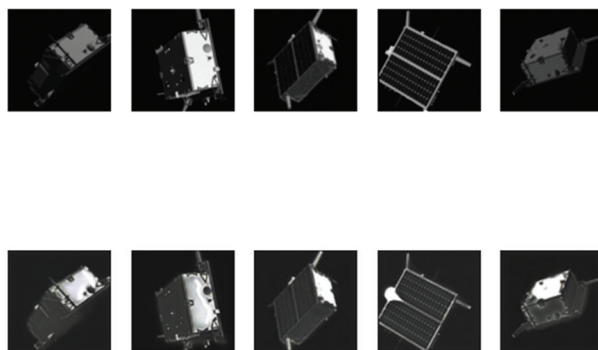


Figure 138. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 8 epochs

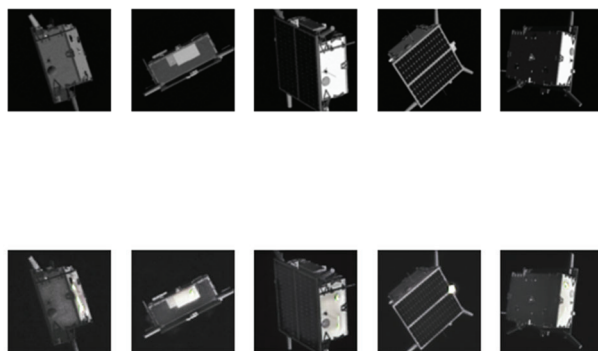


Figure 139. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 14 epochs

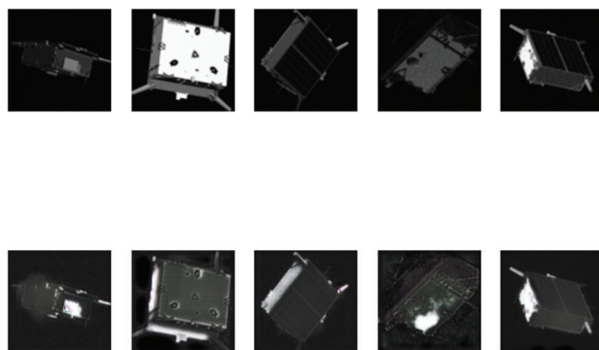


Figure 140. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 18 epochs

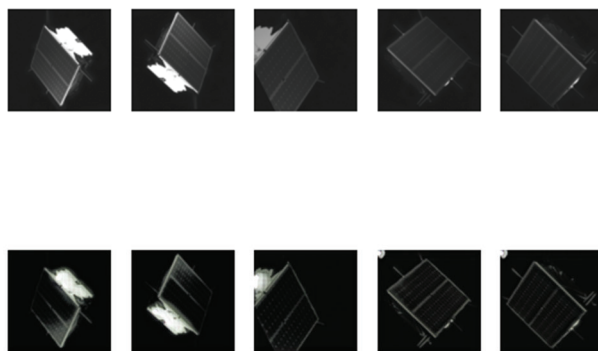


Figure 141. Five examples of R2S, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 18 epochs

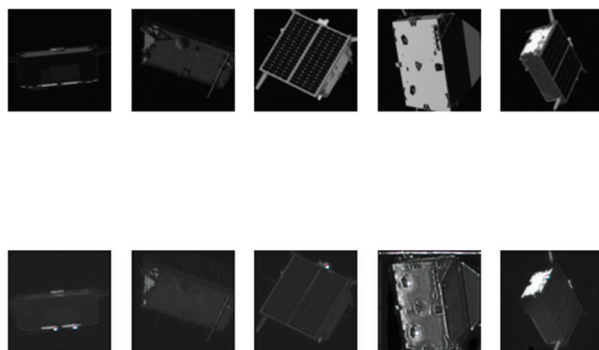


Figure 142. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 23 epochs

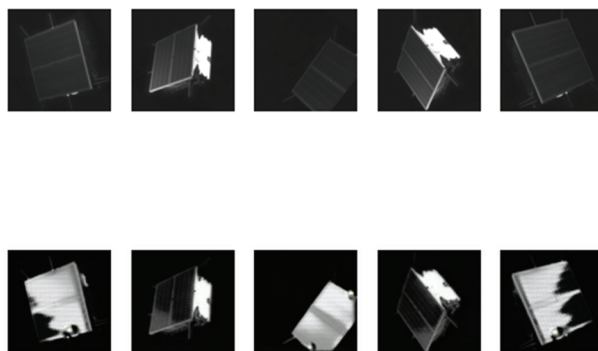


Figure 143. Five examples of R2S, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 23 epochs

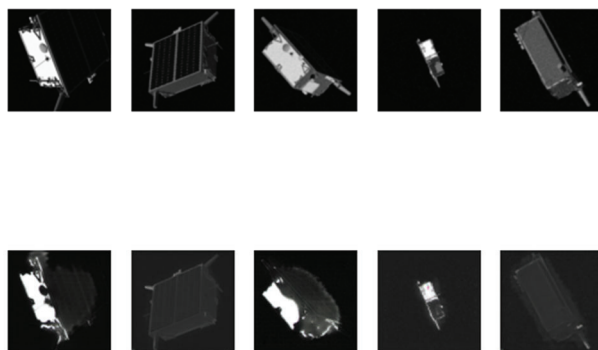


Figure 144. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 34 epochs

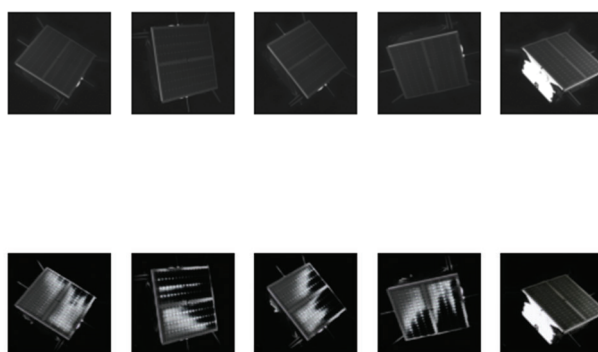


Figure 145. Five examples of R2S, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 34 epochs

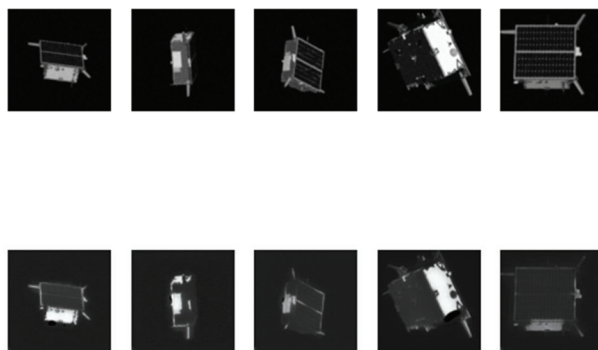


Figure 146. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 36 epochs

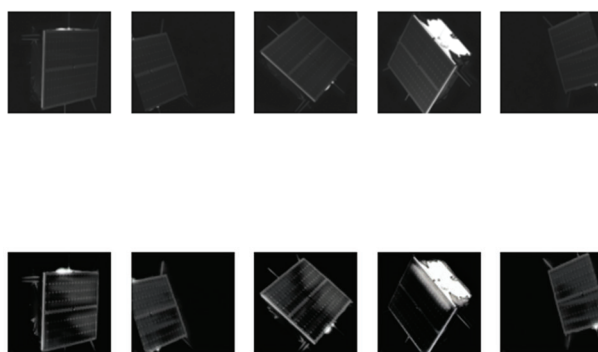


Figure 147. Five examples of R2S, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 36 epochs

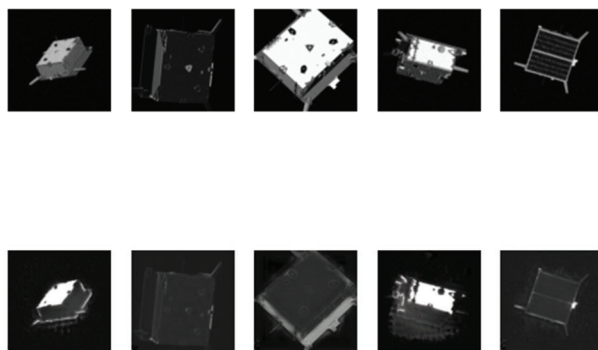


Figure 148. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 47 epochs

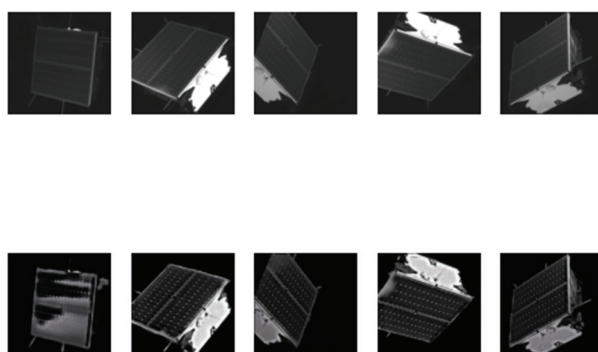


Figure 149. Five examples of R2S, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 47 epochs

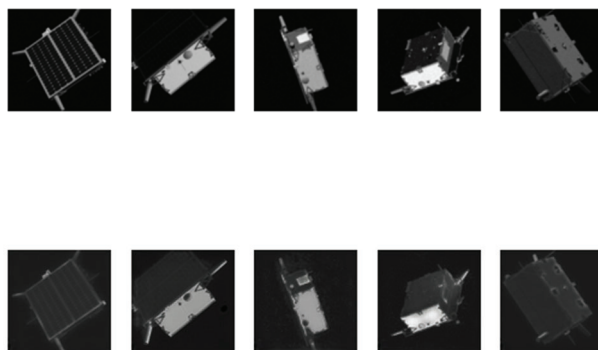


Figure 150. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 50 epochs

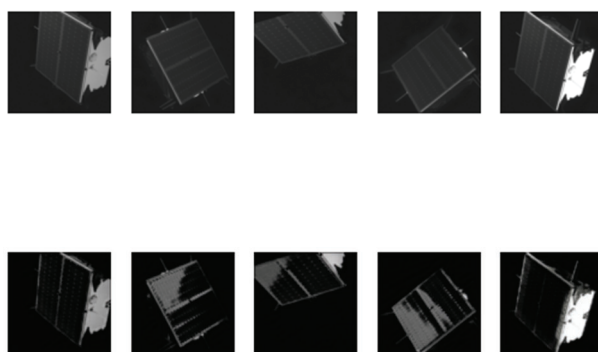


Figure 151. Five examples of R2S, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 50 epochs

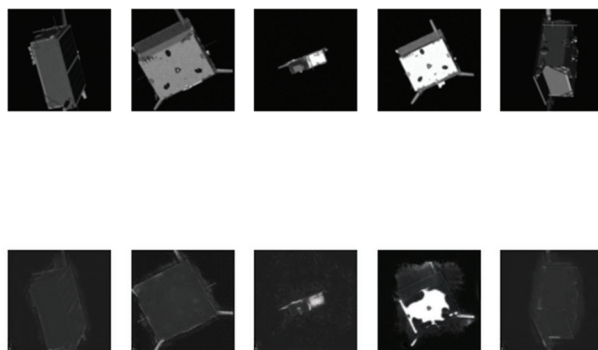


Figure 152. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 52 epochs

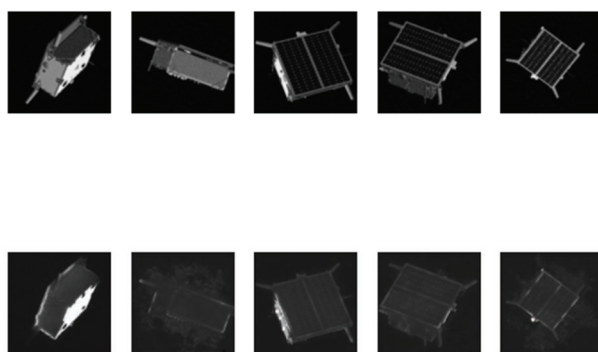


Figure 153. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 76 epochs

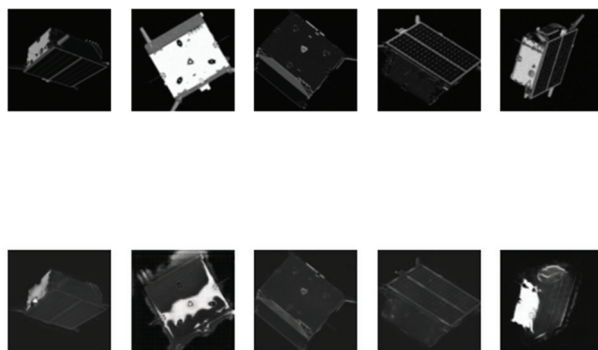


Figure 154. Five examples of S2R, CycleGAN trained on cropped sort-pare SPEED with data augmentation for 82 epochs

6 Analysis and Reduction of Results

The CycleGAN training process was observably dynamic, and even on runs yielding great results, the system is likely to resort to noise and re-converge several separate times during its training. Admittedly, to visually inspect the output of CycleGAN and to find patterns, as in the past few sections, is largely qualitative and may seem borderline artistic. However, the following sections will reinforce the qualitative analysis and make suggestions for future analysis of this type.

6.1 CycleGAN Output

Evident through the CycleGAN trials analyzed in section 5.1 are several predominant themes, discussed in the following subheadings.

6.1.1 *Hierarchy of Introduced Noise*

Over the course of any given CycleGAN training run, different types of visual noise tend to be present at different stages of the training. This list is by no means comprehensive, but follows are several general observed types of this noise and when they tend to occur:

- **Random initialization noise**
 - *Random discoloration (high order) and warping (deep) noise due to CycleGAN's untrained initial weights.*
 - Occurs: epochs < 8
- **Mode collapse**
 - *Defined in 2.2.2, ranges from minor to major common elements of noise across variety of inputs.*
 - Occurs: at variable points in training, depending largely on diversity of training data.
 - Lower diversity (e.g. early un-augmented SPEED trials, with 600 images total) leads to mode collapse as early as epoch 23.
 - More diversity (e.g. POC datasets, or augmented SPEED) leads to minor collapsing anywhere from epoch 67 to not at all.
 - Collapse occurs also when CycleGAN is trained on unbalanced synthetic and real dataset quantities (see 5.1.2).
- **High order (stylistic)**
 - *Named for its impact on the 'high order' features of an image, or those which are more superficial and do not tend to impact image geometry. Examples include discoloration, changes to shading, and ambient noise.*
 - Occurs: throughout training, and is initially the only kind of noise once initialization noise is dispelled. Later in training, occurs in coordination with deep, structural noise.

- **Deep (structural)**

- *Named for its impact on the ‘deep’ features of an image, or those which are more structural and have strong impacts on image geometry. Examples include blacking out, displacement, or warping of image object features and more significant forms of mode collapse.*
- Occurs: later in training, roughly around the same time that the first convergence occurs (see discussion in 6.1.3).

6.1.2 Generalization Error

Notably in CycleGAN’s SPEED trials are distortions that appear consistently when the S2R Generator attempts to transform an image in the synthetic dataset that, for one reason or another, is not represented with an equivalent image in the real dataset. An example of this is the phenomena forming the rationale for sorting SPEED (discussed in 4.5.2), where the S2R Generator did not have equivalents to the Earth-backgrounded synthetic images in the real dataset, and thus attempted to ignorantly convert said images into space backgrounds (see Figure 40, resulting in the generalization seen in Figure 45). Another subtler example occurred in 5.1.3, coming out when attempting to use the visually favorable S2R Generator at epoch 100 (Figure 71) to process the entire 7,048 synthetic images of Sorted SPEED. This Generator produced output that was very reminiscent of the real imagery from the larger-scaled satellite images, as seen in Figure 155.

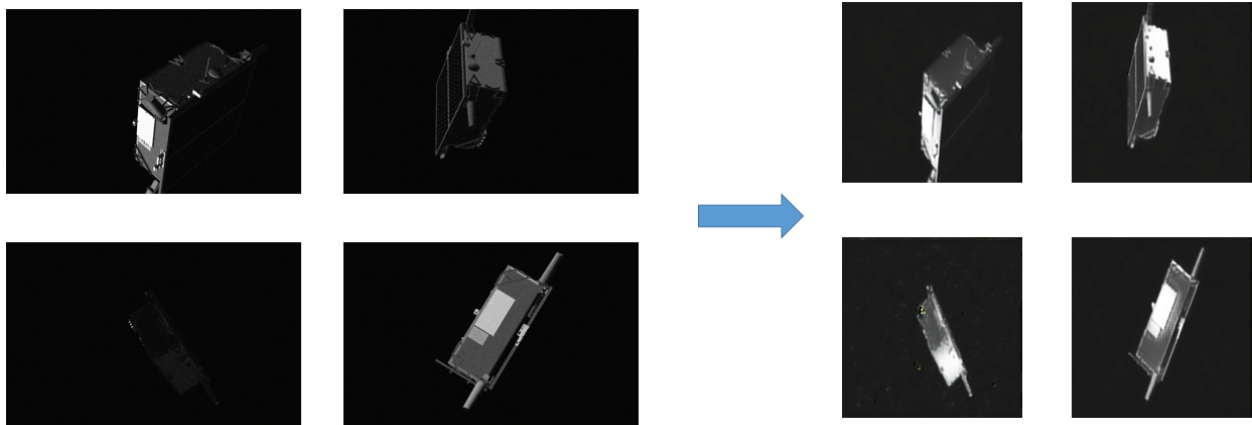


Figure 155. Selected favorable results from CycleGAN synthetic-to-real Generator, trained on uncropped Sort-Pare SPEED for 100 epochs

However, this Generator also produced output that was very noisy specifically for the minority of images where the satellite was very distant, as seen in Figure 156.

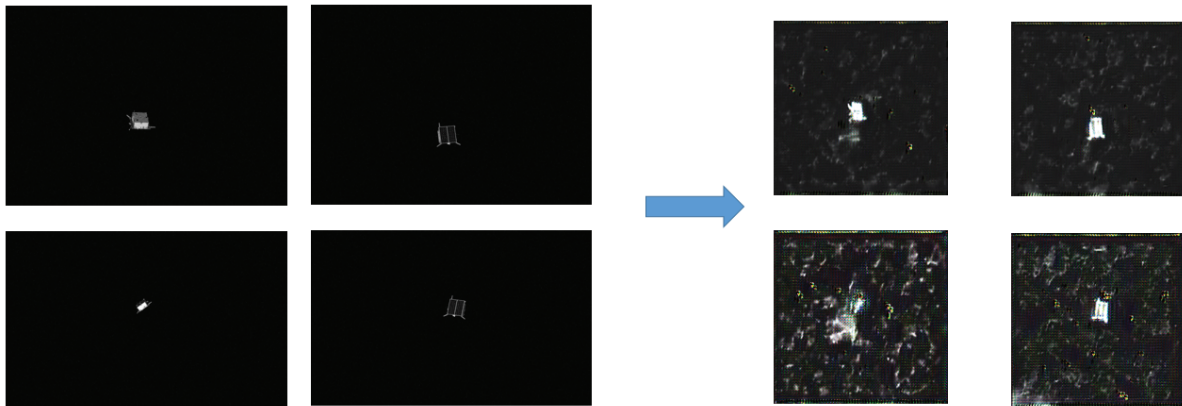


Figure 156. Selected unfavorable results from CycleGAN synthetic-to-real Generator, trained on uncropped Sort-Pare SPEED for 100 epochs

This is ultimately due to the fact CycleGAN did not have equivalents to the distant synthetic satellite images in the real dataset, as shown in Figure 157 below.

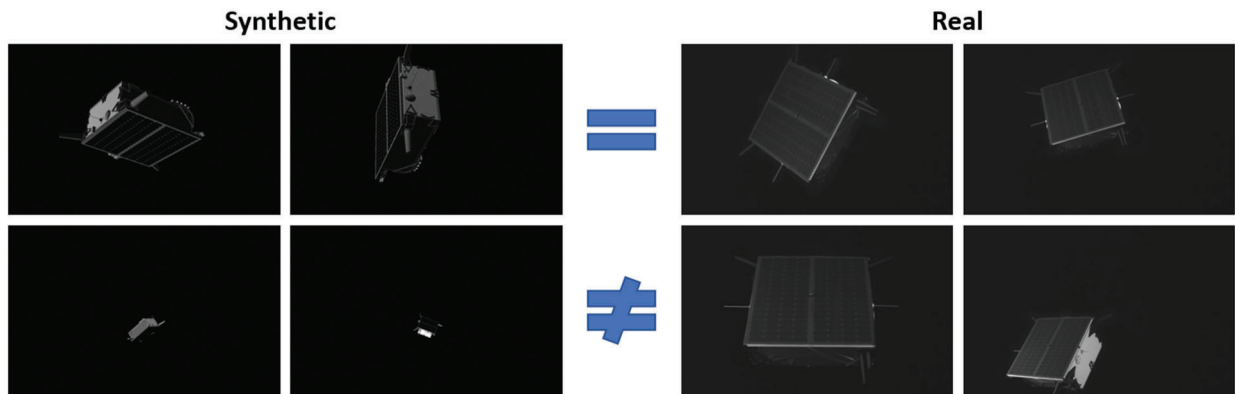


Figure 157. Lack of distant-satellite image representation in real SPEED imagery

This was mostly resolved by cropping SPEED, which mitigated the noted differences in position and scale (see Figure 41).

While avoiding this mode of error was tricky with the limited resources of SPEED, this was a promising aspect of CycleGAN: as it is limited in its ability to broadly generalize over semantically very different types of inputs (e.g. those mentioned in the above discussion), it is thus more capable of specifically generalizing over a few different input cases (e.g. two synthetic SPEED images with different orientations, but the same relative position and scale in the image) and thus producing more meaningful output for carefully-constrained use cases and not over-generalized, meaningless output.

6.1.3 Regularity of Convergences

Each training run had several instances of where the S2R Generator appeared to have converged, whether it was to a visually favorable set of outputs, or whether it suffered mode collapse. Follows in Table 5

are the training runs covered in 5.1 and the epochs at which these apparent convergences occurred. Note that the POC2 training run was not included, as the lack of image variance hampered any ability to visually distinguish points of convergence.

Table 5. Apparent convergences in analyzed CycleGAN trials

CycleGAN Trial	No. Epochs Trained	Epochs of Apparent S2R Convergences	Average Epoch Spacing
Uncropped SPEED	100	23, 45, 71, 100	25.7
POC	67	12, 24, 45-52	19.0
Cropped SPEED	100	(22, 33, 50, 58)*	12.0
Cropped, Augmented SPEED	100	14, 36, 50	18.0

*Convergence via mode collapse

A primary observation is that epochs of convergence are relatively evenly spaced for a given training run, with the exception of the location of the first epoch. Thus, ‘Average Epoch Spacing’ (*AES*) was computed as

$$AES = \frac{(final\ convergence) - (first\ convergence)}{number\ of\ convergences} \quad [21]$$

intentionally not considering where in the training the first convergence occurred.

Note that *AES* was notably uniform across the different trials listed, but note that this is most likely a coincidence, given the differences in training dataset and epoch size. Within trials conducted on similar datasets (i.e. POC or SPEED -derived datasets), suspected reasons for its differences across trials, as well as those for the differences in first convergence epochs, include quantity/diversity of training data, stochastic differences in CycleGAN initialization weights, as well as the hyperparameter settings of the CycleGAN (which were largely untested in this course of research). Note that this discussion is largely speculative given the relatively few datapoints analyzed, but was far from the focus of this research and thus not meant to be comprehensive. Rather, this information in the context of the preceding data in 5.1 provides practical advice for the CycleGAN practitioner: expect a regularly spaced convergence pattern.

Moreover, as noted for several trials in 5.1, CycleGAN alternates between these convergences and a characteristic mode of noisy output, often involving heavy structural distortions. This is indicative of a bimodal stability characteristic to CycleGAN as it progresses in its training, something that is regular and perhaps even mathematically modellable upon further analysis.

6.2 Training Pose Estimation CNNs

Several of the visually converged epochs from each training run were implemented in the pipelines developed in 4.5, and validated using the method discussed in 4.4. Their results, per dataset, are in the discussed in the following sections.

6.2.1 POC

The first tests were done on the POC dataset, using the intentionally geometrically-simple model. The differences between the synthetic and pseudo-real subsets are low-fidelity versus higher-fidelity rendering, with differences in background and texture. Following in Figure 158 are the resulting classification accuracies of the synthetically-trained CNN (in yellow), and the photorealistically-trained CNNs (in blue) using the CycleGAN S2R Generator at training epochs 23, 45, and 52. Note that, in all cases, the CNN architecture was trained on the full 54,000 images but that an ‘epoch’ was declared to be 1200 images (see section 5.1.4).

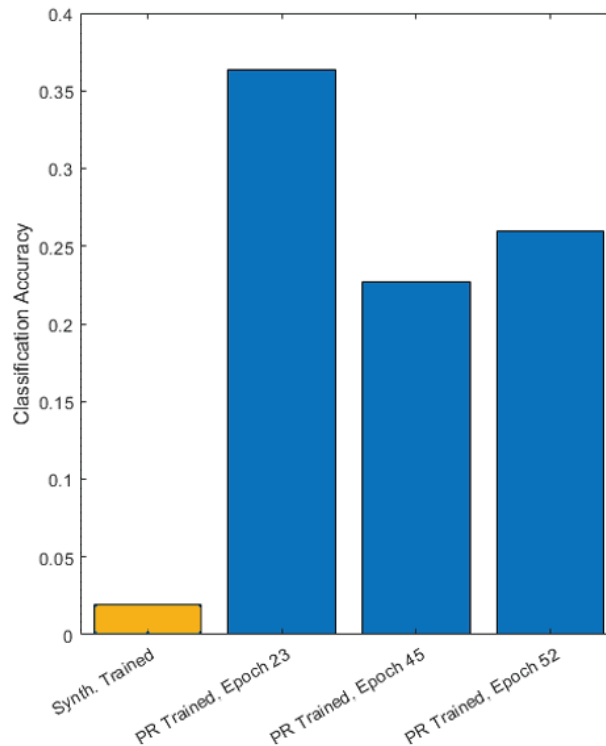


Figure 158. Pose classification accuracies for synthetically- and photorealistically- trained CNNs on POC dataset

Here the results show the large gap between these datasets. The lower percentages for synthetically-trained and significantly higher for that of the photorealistically-trained indicates that a significant, meaningful change (with respect to the CNN’s interpretation) is being made by the CycleGAN. At the same time, the early and universal (across all tested photorealistic datasets) acquisition of this change may mean that this change was somewhat trivial (i.e. shifting the background color).

6.2.2 POC2

To draw attention to some other possible modes of change between datasets, and incentivizing the CycleGAN to focus on the subtler, foreground-based changes not emphasized by POC, POC2 was generated

in the same fashion as POC with the same background color, more differences in texture and lighting/contrast in the foreground/satellite object, and independently random lighting between the synthetic and pseudo-real datasets (see Figure 29). Following in Figure 159 are the resulting classification accuracies of the synthetically-trained CNN, and the photorealistically-trained CNN using the CycleGAN S2R Generator at training Epochs 48, 49, and 57. Note that, in all cases, the CNN architecture was trained on the full 54,000 images but that an ‘epoch’ was declared to be 1200 images (see section 5.1.5).

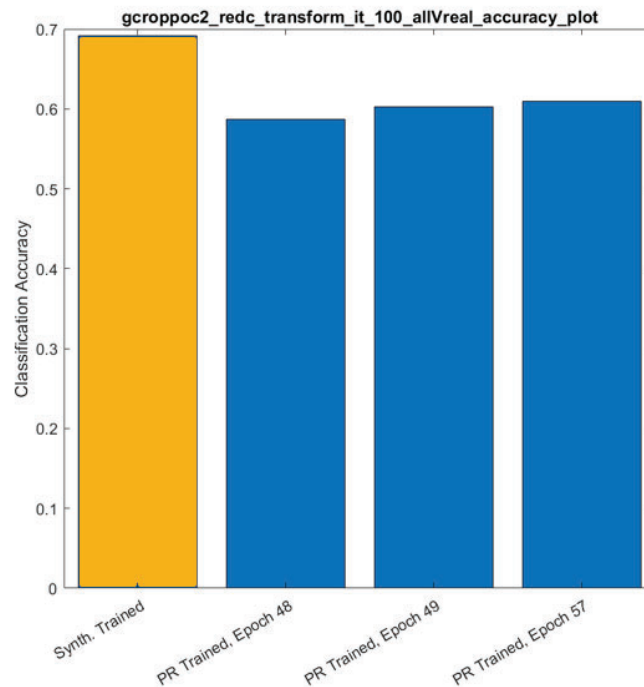


Figure 159. Pose classification accuracies for synthetically- and photorealistically- trained CNNs on POC2 dataset

As you can see, the results contrast significantly with the previous. This is perhaps due to the fact that the CycleGAN is making too much change to an already very similar pair of datasets, and that the important features to the CNN aren’t those accentuated by this dataset (such as the salt and pepper noise texture that was intentionally brought out in this dataset).

Also note that the network was able to generalize over the randomized shading rather well, as can be seen by the high (~70%) classification accuracy of the synthetically-trained network against data with different random shading patterns (see above).

6.2.3 SPEED

The practices learned from POC and POC2, as well as the previous uncropped SPEED tests, were utilized in subsequent SPEED trials. However, noting the aspects of this dataset that make it more challenging for implementation in training CycleGAN and CNN (see 3.2), these practices did not immediately lead to the same good quality of visual output seen in epoch 100 of the included uncropped SPEED trial (Figure 71). Rather, these images suffered the same type of aberrations as when the CycleGAN was

overtrained on real imagery. As a result, when attempting to train the pose estimator CNN on this output, the many visual disturbances prevented it from training properly; i.e., the CNN failed to converge to any appreciable accuracy of classification, as seen in Figure 160 below.

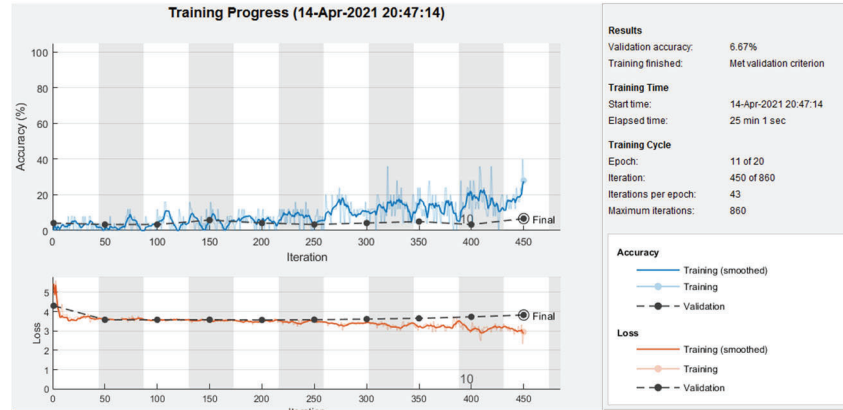


Figure 160. Failure to converge in CNN pose estimator training

This is in direct comparison to an example successful training on the unmodified synthetic data, which reaches 80%+ validation accuracy (as seen in Figure 161).

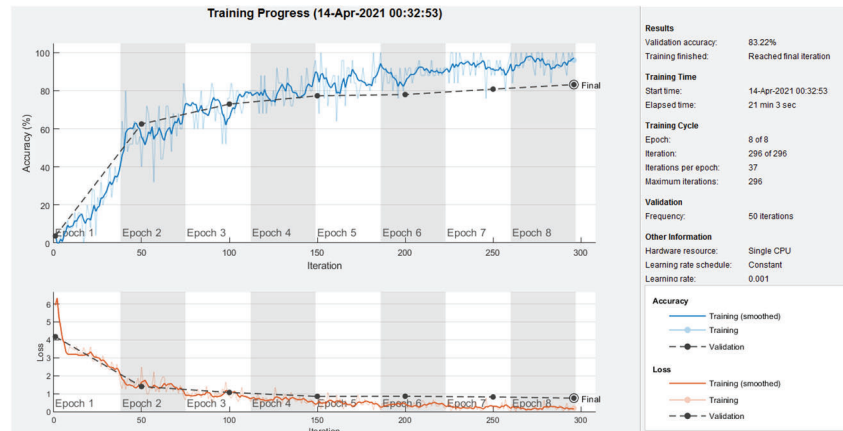


Figure 161. Properly-trained, converged CNN pose estimator

To solve this, data augmentation was introduced during CycleGAN training, as discussed in 4.5.2. As a result of the artificially increased diversity in the training imagery introduced to CycleGAN, the resulting output was significantly more visually photorealistic and the pose estimation CNN converged on the produced photorealistic datasets, as in Figure 161. Follows is the discussion of how well these photorealistic datasets prepared the CNN for real imagery.

Figure 162 and Figure 163 are the same networks and training datasets using the two different pose estimation training configurations (configurations 5 and 6 considered in Table 3).

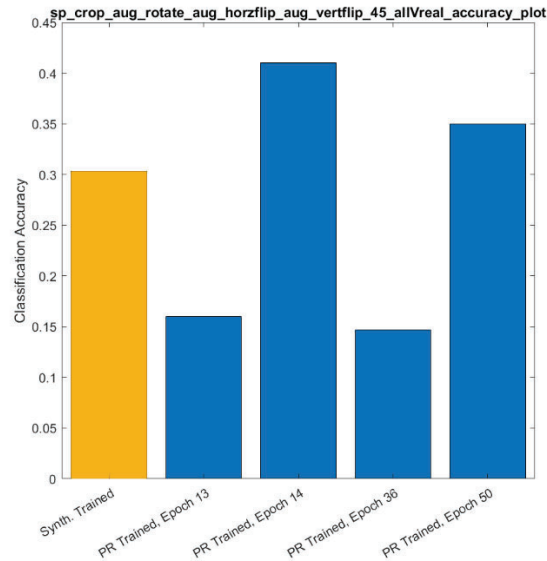


Figure 162. Pose classification accuracies for synthetically- and photorealistically- trained CNNs on SPEED; pose label configuration 5 as per Table 3

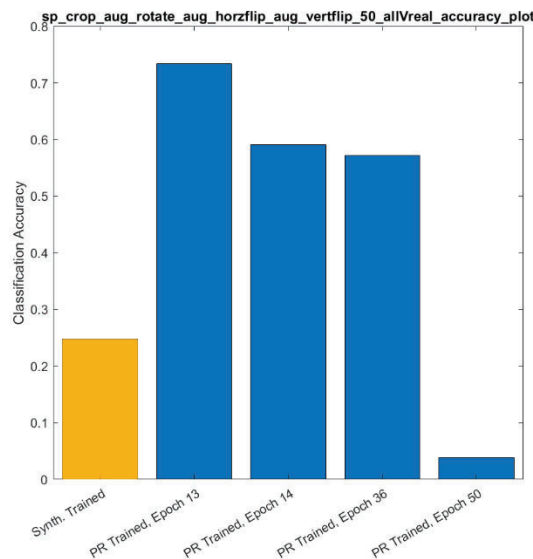


Figure 163. Pose classification accuracies for synthetically- and photorealistically- trained CNNs on SPEED; pose label configuration 6 as per Table 3

While there is inconsistency between these two configurations in terms of which photorealistic dataset improves performance against real imagery, both generally agree that CycleGAN adds the kind of information that predisposed these networks to the real data better than the Synthetic data did. In addition, this indicates that the style of 300 images was at least partially successfully transferred to a dataset roughly 23 times its size; i.e. the fact that the 2300-2770 image subset (see Table 3) of Sorted SPEED (7,048 images) was

more meaningfully photorealistic than the corresponding synthetic data probably indicates that the same logic is applicable to the entirety of Sorted SPEED. Note that the amount of real imagery used in the tests varied between configurations 5 and 6, see Table 4.

6.3 Non-Visual Methods of CycleGAN Evaluation and Implications

As with all neural networks, it is useful to look at CycleGAN’s own training ‘metrics’ (i.e. Generator and Discriminator losses, Discriminator accuracies) throughout each of its trained epochs in order to gain insight on training progress. After all, the loss functions of the S2R and R2S Generators (Equation [15]) and the synthetic and real Discriminators (Equation [13]) is what drives each network’s training; for the Discriminators, their performance can also be rated by how accurately they can predict whether an image is from the parent dataset, or if it was a result of Generator style transfer.

These two classes of quantities were plotted for the data-augmented cropped SPEED training run of CycleGAN for each epoch, and can be seen in the center of Figure 164 below. Losses are labeled for the S2R (‘gen-S2R’) and R2S (‘gen-R2S’) Generators, and the synthetic (‘dSynth’) and real (‘dReal’) Discriminators for both the images they encounter that are legitimately from the parent dataset (‘d__-real’) and those that are the result of their related Generator style transfer (‘d__-fake’). Discriminator accuracies are labeled using the same convention, preceded by ‘acc.’

The epochs of apparent convergence for this training run were 13, 14, 36, and 50, which were selected from the rest based entirely on subjective visual acuity (see 5.1.7). In Figure 164, each corresponding S2R Generators’ example output for the input image (marked ‘STARTING POINT’ on the top left) are marked with their locations in the training metric plots for comparison.

Additional to Figure 164 are image histograms, included for the example synthetic (left, top) and real (right, top) images. These can be compared to example histograms from the S2R output early (left, bottom) and later (right, bottom) in the training.

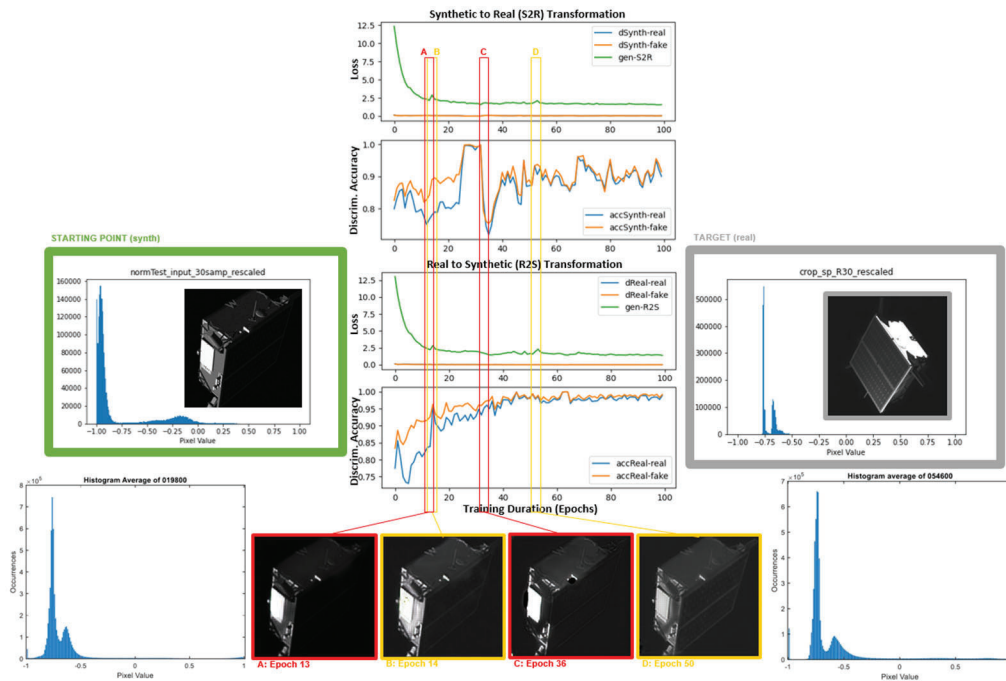


Figure 164. Visually-selected photorealistic SPEED associated with the point in CycleGAN training at which they were generated, alongside histograms for reference

There are several observable features in Figure 164. Firstly, note the two major spikes in the S2R Generator's loss (top plot, green line) indicate that there is something new in the training data that upsets the previous balance of the Generator. As marked in yellow, they coincide with two of the selected epochs, 14 and 50. This might suggest fascinatingly that that visually favorable output has a correlation with this spike phenomenon in the loss function value, or when the Generator encounters and recovers from a training perturbation. Note that, for the epochs marked in red, there was no such correlation in the loss value.

Additionally, referencing Figure 162, which utilized pose estimation configuration 5 (as per Table 3) that involved more real imagery in its tests (see Table 4), epochs 14 and 50 are coincidentally the only ones that surpass the synthetically-trained network in accuracy, where 13 and 36 fall short. Perhaps this is indicative that, of the epochs chosen, the ones that correspond with spikes in the data are the only ones that are meaningfully photorealistic. This sweeping statement tempered with the accuracy results from pose estimation configuration 6 (see Figure 163), but it is important to note that this trial used significantly less real data than that from pose estimation configuration 5 (again, see Table 4). Overall, this can be used to show that visually favorable results may be predictable from trends in the loss function, but as a result of being entirely based on qualitative judgement, will not always be completely encompassed by them.

Secondly, note that the S2R Generator, when input with synthetic imagery with histograms reminiscent of that on the top left, process said imagery in a manner that gradually changes the shape of the histogram to match that of the real imagery (top, right). This is significant, as while the output imagery might appear to be trivially brightened by the Generator, the pattern in the histogram reveals a much more sophisticated and perhaps intrinsically meaningful process.

Other notable observations from Figure 164 include that the S2R and the R2S Generator losses have roughly the same trend with respect to training epoch, including the spikes noted above, indicating their

strong coupling and substantiating the discussions throughout, at least, the respective training run in 5.1.7. For the purposes of this analysis, this can be generalized to suggest that that each Generator network experiences periods of convergence in the same frequency as their counterpart, and generally in sync with (e.g. 5.1.3 and 5.1.4) or at a phase shift to the other Generator (e.g. 5.1.6), as promoted through the connection of their loss functions (see Equation [15])

7 Future Work

7.1 Using Style Transfer as a Means of Augmenting Real Training Data

In cases where labels for real imagery are available, it may be beneficial to compile a training dataset of this real imagery combined with photorealistic imagery output by CycleGAN. It is important to note that, if the real imagery used in this training dataset is the same used to train the CycleGAN on real style, that the resulting photorealistic data will emulate this real data and perhaps lead to overtraining in the pose estimator CNN. (Note that the usage of ‘augmenting’ in this section differs from that of ‘data augmentation,’ defined in 4.5.2.3).

7.2 Structural Change Analysis Using Classical Feature Detection Comparison

As proposed in [23], classical methods of edge detection such as SIFT are utilized to ensure structural features are preserved between synthetic and photorealistic datasets. This may be useful in ensuring that the data output by the CycleGAN are preserving the geometry and major identifiable features of a target satellite.

7.3 Range Determination and More Sophisticated CNNs for Pose Estimation

Additional CNN-based pose estimators proposed in the ESA/Stanford challenge ([42]) are able to obtain very low margins of synthetic pose estimation error using the synthetic SPEED dataset. Having shown that the CycleGAN is able to communicate valuable information to the example CNN’s feature extraction, it should do the same for these architectures.

Additionally, the process for extracting range information in these pose estimators is often accomplished simultaneously with detection and cropping models. Since this is a largely classical process, one would not need to train these models, and could implement CycleGAN style transfer in between this range determination and the CNN-based pose estimation.

As discussed in 4.4, the ultimate goal for CNN pose estimation is to design systems that do more than classify, and that have enough of an attachment to physical reality to provide a gradient of pose estimates. The CNNs proposed in [42] do so, and thus instead of accuracy they are evaluated in terms of their error. Likewise, the CNN pose classifiers utilized in this research have demonstrated some rudimentary ability to root their classifications physically and not simply arbitrarily. I.e., the classifier is able to determine which pose labels are physically nearest to each other, and when being tested the errors are most often in labels that are close to the correct answer; this is in contrast to before, where the classifier was assumed to be ignorant of this, and viewing pose labels as mutually exclusive classes.

7.4 Methods of Model-Agnostic GAN Style Transfer Evaluation

A fundamental challenge with using such high-level machine learning techniques is the tendency to view them as ‘black boxes.’ Due to the peculiar dynamic nature of GANs and the high dimensionality of

input and output data, it is difficult to evaluate and compare them from a standpoint that is not subjective, let alone a mathematical one. See Figure 165 for a generalized method used to approach objective analysis.

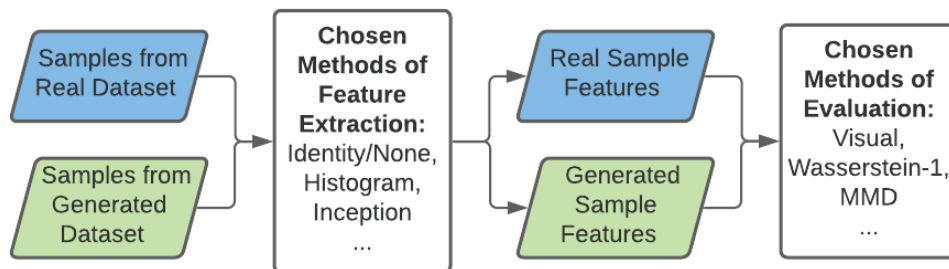


Figure 165. General evaluation pipeline for GAN output (inspired from [38])

Feature extraction as defined in Figure 165, in general, is a way to interpret the image that allows the analyst to view members of a dataset as vectors, and by extension the dataset as a whole as a statistical distribution, which then lends itself to the use of quantitative methods of analysis. Feature extraction includes anything from simply viewing the images as matrices of their pixel values, to more advanced forms of data reduction such as processing the image through Google’s pretrained InceptionV3 CNN and storing its neuron activations at the pooling level. [39]

Many quantitative metrics for evaluation as it is defined in Figure 165 have been proposed in research. Originally, those utilized in [38] were implemented using the Keras library of Python for use in this research. Six metrics were implemented in order to avoid the shortcomings of any one in particular; they are items of active research and can oversimplify the output of the GAN (advised in [38]). Upon computing these metrics for each epoch of the training runs highlighted in 5.1, there was little interpretability in and across the resulting trends, and were not discerned as providing explanation for any of the results obtained.

In hindsight, many perhaps over-simplifications were employed in computing these metrics that may have led to their noisy trends. An example of which is that only 150 random images from S2R CycleGAN were used at each epoch to form the ‘distribution’ to evaluate, as any more would cause the computing resource (Michaelson Hall’s ‘Trouble/Tribble’) to encounter fatal ‘out-of-memory’ errors; this coding implementation should be revisited in order to enable it to process all of the images in each dataset as they are processed by S2R. Additionally, it is mentioned in other research that CycleGAN is unwieldy when it comes to using such evaluation metrics prescribed for general cGAN architectures. [40]

8 Final Recommendation

In general, the analysis performed somewhat falls short of being able to provide a fully-qualified recommendation on behalf of pose estimation CNNs. The primary issue lies in the degree of accuracy provided by the photorealistically-trained CNNs, which can be as high as 70% (as in the selected results from training with SPEED using one particular pose label configuration), but are more likely to be only marginally better than their synthetically-trained counterparts (as in the selected results from training with SPEED using another particular pose label configuration). While the amount of real data provided in SPEED is small, that was expected from the start and should not be blamed -- it is posited that more sophisticated CNN-based pose estimators (such as those who won the ESA/Stanford competition) would be more capable of attaining higher accuracies and thus be an important next step in this analysis.

However, this research's primary goal was to validate that GANs could add realism to synthetic data in a way that is meaningful to CNNs, which was reinforced by the results in section 6.2 that indicated that training with such GAN-processed 'photorealistic' data non-negligibly improved CNN classification accuracy of real SPEED imagery at various stages. This is by no means a negligible step, and has been approached by other articles in various fields (such as for increasing photorealism of satellite imagery of Earth features [23] and medical CT scan imagery [41]) including for satellite pose estimation, by the creators of SPEED [42].

Secondarily, this research shed light on interesting and perhaps modellable characteristics of CycleGAN for use in style transfer applications, such as the fact that the network appeared to converge periodically across the different datasets tested. It is important to note that many of the characterizations and recommendations might be pigeonholed to this use case of CycleGAN. As made abundantly clear in section 4.5.2, there are many system parameters at play in the proposed image processing pipeline, from uncropped imagery to photorealistic training dataset, many pertinent ones of which were not explored in this research. Perhaps the most principal of these is the balance of the smaller synthetic imagery set used to train CycleGAN and the greater set, which is processed by the trained CycleGAN and converted into photorealistic imagery. Generalization is a hallmark of machine learning, but to train a network to process a dataset (the greater dataset) roughly 23 times the size of its training dataset (the lesser dataset) was a large conceptual gap. Since this gap was successfully bridged using this concept, several questions must follow: what similarity characteristics in the dataset made this possible, and can this be boiled down to a statistical measure? Since the quantity of SPEED data used to train CycleGAN (300 synthetic, 300 real) was small in machine learning terms due to the constraints of the real data available, at what quantity of data does this phenomenon break down for this dataset?

Models can always be tuned, especially in the case of CycleGAN. My prediction is that, once this occurs, one will see photorealistically-trained pose estimator CNNs approach the low error margins/high classification accuracies that *synthetically*-trained networks tested against *synthetic* data have achieved. Also, keeping in mind the practical role of these networks as pose *initialization* devices for more sophisticated, classical Guidance, Navigation, and Control (GNC) elements, it is desired that they rapidly acquire pose of an object within a designated ballpark (such as $\pm 10^\circ$ in the NASA's proposed algorithm)[§], thus concerns for high precision pose classification dwindle.

In the grander scope of pursuing an autonomous pose estimation capability for use on spacecraft, this research approaches a model-based solution that must be developed specifically for every target

[§] From an individual conversation with personnel from NASA Goddard, not recorded formally.

geometry. This is missing the mark of a more significant effort to satisfy a desire to have a pose estimator that does not require such *a priori* knowledge of the vehicle geometry at all, and could be used to rendezvous with anything it comes into contact with. Such an approach would have applications in spacecraft servicing, on-orbit construction, space debris mitigation, and many more eye-watering capabilities. Potential lies in using modified model-based pose estimation methods to track salient geometrical features common to many spacecraft (such as solar panels or nozzles) using 2D visual sensors, [43] to which this research has the potential to lend more usefulness.

Bibliography

- [1] Satellite Coordinate System and Euler Angles. *ResearchGate*. https://www.researchgate.net/figure/Satellite-coordinate-system-and-Euler-angles_fig7_237089506. Accessed Jan. 10, 2020.
- [2] Robotic Servicing of Geosynchronous Satellites (RSGS). <https://www.darpa.mil/program/robotic-servicing-of-geosynchronous-satellites>. Accessed Nov. 16, 2019.
- [3] Hall, L. NASA's Restore-L Mission to Refuel Landsat 7, Demonstrate Technologies. *NASA*. <http://www.nasa.gov/feature/nasa-s-restore-l-mission-to-refuel-landsat-7-demonstrate-crosscutting-technologies>. Accessed Nov. 16, 2019.
- [4] Northrop Grumman and SpaceLogistics: Life Extension Services for Satellites. *SpaceLogistics*. <https://www.northropgrumman.com/space/space-logistics-services/>. Accessed Nov. 24, 2020.
- [5] Kelvins - Pose Estimation Challenge - Data. <https://kelvins.esa.int/satellite-pose-estimation-challenge/data/>. Accessed Oct. 30, 2019.
- [6] Kim, P. *MATLAB Deep Learning with Machine Learning, Neural Networks and Artificial Intelligence*. Springer, New York, New York, 2017.
- [7] Lee, S., Jung, S., and Lee, J. "Prediction Model Based on an Artificial Neural Network for User-Based Building Energy Consumption in South Korea." *Energies*, Vol. 12, 2019, p. 608. <https://doi.org/10.3390/en12040608>.
- [8] Gupta, T. "Deep Learning: Feedforward Neural Network." *Medium*, Dec 16, 2018.
- [9] Widrow, B., and Lehr, M. A. "30 Years Adaptive Neural Networks: Perceptron, Madeline, and Backpropagation." *Proceedings of the IEEE*, Vol. 78, No. 9, 1990, p. 1415.
- [10] Understanding Backpropagation Algorithm - Towards Data Science. <https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd>. Accessed Jan. 10, 2020.
- [11] Hornik, K., Stinchcombe, M., and White, H. "Multilayer Feedforward Networks Are Universal Approximators." *Neural Networks*, Vol. 2, No. 5, 1989, pp. 359–366. [https://doi.org/10.1016/0893-6080\(89\)90020-8](https://doi.org/10.1016/0893-6080(89)90020-8).
- [12] Gatys, L. A., Ecker, A. S., and Bethge, M. "A Neural Algorithm of Artistic Style." *arXiv:1508.06576 [cs, q-bio]*, 2015.
- [13] Arel, I., Rose, D. C., and Karnowski, T. P. "Deep Machine Learning - A New Frontier in Artificial Intelligence Research [Research Frontier]." *IEEE Computational Intelligence Magazine*, Vol. 5, No. 4, 2010, pp. 13–18. <https://doi.org/10.1109/MCI.2010.938364>.
- [14] Sarkar, D. (DJ). A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning. *Medium*. <https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a>. Accessed Apr. 26, 2021.
- [15] Horev, R. Style-Based GANs – Generating and Tuning Realistic Artificial Faces. *Lyrn.AI*, Dec 26, 2018.
- [16] Karacan, L., Akata, Z., Erdem, A., and Erdem, E. "Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts." *arXiv:1612.00215 [cs]*, 2016.
- [17] Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. "Improved Techniques for Training GANs." *arXiv:1606.03498 [cs]*, 2016.
- [18] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. "Generative Adversarial Networks." *arXiv:1406.2661 [cs, stat]*, 2014.

- [19] Working Principles of Generative Adversarial Networks (GANs) - DZone AI. *dzone.com*. <https://dzone.com/articles/working-principles-of-generative-adversarial-netwo>. Accessed Jan. 10, 2020.
- [20] Liang, L., Li, G., and Juang, B.-H. “Deep Learning-Based End-to-End Wireless Communication Systems With Conditional GANs as Unknown Channels.” *IEEE Transactions on Wireless Communications*, 2020.
- [21] Zhao, H., Li, H., and Cheng, L. *Synthesizing Images with GANs*.
- [22] Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. <https://arxiv.org/abs/1703.10593>. Accessed Jan. 10, 2020.
- [23] Fuentes Reyes, M., Auer, S., Merkle, N., Henry, C., and Schmitt, M. “SAR-to-Optical Image Translation Based on Conditional Generative Adversarial Networks—Optimization, Opportunities and Limits.” *Remote Sensing*, Vol. 11, No. 17, 2019, p. 2067. <https://doi.org/10.3390/rs11172067>.
- [24] Brownlee, J. How to Develop the CycleGAN End-to-End. In *Generative Adversarial Networks with Python*, pp. 551–579.
- [25] Brownlee, J. How to Identify and Diagnose GAN Failure Modes. In *Generative Adversarial Networks with Python*, pp. 207–242.
- [26] Foundation, B. Blender.Org - Home of the Blender Project - Free and Open 3D Creation Software. blender.org.
- [27] Kisantal, M., Sharma, S., Park, T. H., Izzo, D., Mörtens, M., and D’Amico, S. “Satellite Pose Estimation Challenge: Dataset, Competition Design and Results.” *IEEE Transactions on Aerospace and Electronic Systems*, 2020.
- [28] Berkeley, M. L. @. Tricking Neural Networks: Create Your Own Adversarial Examples. *Medium*. <https://medium.com/@ml.at.berkeley/tricking-neural-networks-create-your-own-adversarial-examples-a61eb7620fd8>. Accessed May 10, 2021.
- [29] Oliver Chocron. Euler ZYX Convention. <http://web.mit.edu/2.05/www/Handout/HO2.PDF>. Accessed May 10, 2021.
- [30] Remote Sensing | Free Full-Text | Pre-Trained AlexNet Architecture with Pyramid Pooling and Supervision for High Spatial Resolution Remote Sensing Image Scene Classification | HTML. <https://www.mdpi.com/2072-4292/9/8/848/htm>. Accessed Apr. 22, 2021.
- [31] AlexNet Convolutional Neural Network - MATLAB Alexnet. <https://www.mathworks.com/help/deeplearning/ref/alexnet.html;jsessionid=c0d14b5a8c46828ece5b6a1ac58a>. Accessed Apr. 19, 2021.
- [32] OpenCV: Interactive Foreground Extraction Using GrabCut Algorithm. https://docs.opencv.org/3.4/d8/d83/tutorial_py_grabcut.html. Accessed May 4, 2021.
- [33] Chaubey, A. “Downsampling and Upsampling of Images — Demystifying the Theory.” *Medium*, Jan 14, 2020.
- [34] Dennanni, A. How to Deal with Image Resizing in Deep Learning. *Medium*. <https://medium.com/neuronio/how-to-deal-with-image-resizing-in-deep-learning-e5177fad7d89>. Accessed May 10, 2021.
- [35] Gavin Taylor. Instructions for Interface with Computing Cluster. <https://www.usna.edu/Users/cs/taylor/courses/si486l/resources/cluster.html>. Accessed Apr. 26, 2021.
- [36] Gavin Taylor. Instructions for Interface with Tribble/Trouble GPUs. <https://www.usna.edu/Users/cs/taylor/courses/si486l/resources/tribble.html>. Accessed Apr. 26, 2021.
- [37] Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. “Analyzing and Improving the Image Quality of StyleGAN.” *arXiv:1912.04958 [cs, eess, stat]*, 2020.
- [38] Xu, Q., Huang, G., Yuan, Y., Guo, C., Sun, Y., Wu, F., and Weinberger, K. “An Empirical Study on Evaluation Metrics of Generative Adversarial Networks.” *arXiv:1806.07755 [cs, stat]*, 2018.

- [39] Brownlee, J. How to Implement the Frechet Inception Distance. In *Generative Adversarial Networks with Python*, pp. 266–278.
- [40] Ren, C., Ziemann, A., Theiler, J., and Durieux, A. Deep Snow: Synthesizing Remote Sensing Imagery with Generative Adversarial Nets. 2020.
- [41] Sandfort, V., Yan, K., Pickhardt, P. J., and Summers, R. M. “Data Augmentation Using Generative Adversarial Networks (CycleGAN) to Improve Generalizability in CT Segmentation Tasks.” *Scientific Reports*, Vol. 9, No. 1, 2019, pp. 1–9. <https://doi.org/10.1038/s41598-019-52737-x>.
- [42] Park, T. H., and D’Amico, S. “GENERATIVE MODEL FOR SPACECRAFT IMAGE SYNTHESIS USING LIMITED DATASET.” p. 16.
- [43] Uncooperative Spacecraft Pose Estimation Using Monocular Monochromatic Images | Journal of Spacecraft and Rockets. <https://arc.aiaa.org/doi/10.2514/1.A34775>. Accessed May 10, 2021.

**Appendix A: Training Validation Accuracies at Increasing Images
per Label**

Images per Label	Validation Accuracy
5	40.17%
10	51.74%
15	69.02%
20	73.11%
25	78.09%
30	80.31%
35	81.54%
40	84.17%
45	84.56%
50	86.31%
55	90.59%
60	82.19%
65	88.20%
70	91.93%
75	91.56%
80	90.89%
85	91.44%
90	93.09%
95	88.98%
100	91.04%