5-8-2020

Synthetic Aperture Radar simulation by Electro Optical to SAR Transformation using Generative Adversarial Network

Jason Slover
jks7592@rit.edu

Synthetic Aperture Radar simulation by Electro Optical to SAR Transformation using Generative Adversarial Network

by

Jason Slover

B.S. Angelo State University, 2013

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science
in the Chester F. Carlson Center for Imaging Science
College of Science
Rochester Institute of Technology

May 8, 2020

Signature of the Author ______________________________________

Accepted by ____
Coordinator, M.S. Degree Program Date

CHESTER F. CARLSON CENTER FOR IMAGING SCIENCE
COLLEGE OF SCIENCE
ROCHESTER INSTITUTE OF TECHNOLOGY
ROCHESTER, NEW YORK

CERTIFICATE OF APPROVAL

M.S. DEGREE THESIS

The M.S. Degree Thesis of Jason Slover
has been examined and approved by the
thesis committee as satisfactory for the
thesis required for the
M.S. degree in Imaging Science

Dr. Michael Gartley, Thesis Advisor

Dr. Charles Bachmann

Dr. Michael Jay Schillaci

Date

Synthetic Aperture Radar simulation by Electro Optical to SAR Transformation using Generative Adversarial Network

by

Jason Slover

Submitted to the
Chester F. Carlson Center for Imaging Science
in partial fulfillment of the requirements
for the Master of Science Degree
at the Rochester Institute of Technology

Abstract

The CycleGAN generative adversarial network is applied to simulated electo-optical (EO) images in order to transition them into a Synthetic Aperture Radar (SAR)-like domain. If possible this would allow the user to simulate radar images without computing the phase history of the scene. Though visual inspection leaves the output images appearing SAR-like, examination by t-distributed Stochastic Neighbor Embedding (t-SNE) shows that CycleGAN was insufficient at generalizing an EO-to-SAR conversion. Further, using the transitioned images as training data for a neural network shows that SAR features used for classification are not present in the simulated images.

Contents

List of Figures

List of Tables

Chapter 1

Background/Theory

1.1 Objectives

The goal of this thesis is to answer the question as to if it is possible to convert a simulated electro-optical image into the domain of Synthetic Aperture Radar (SAR) using the impulse response and a generative adversarial network instead of simulating the full phase history. This would allow non-physics based approaches to simulating radar images for situations that would normally be computationally over intensive and impractical for physics based models. Ultimately this means that approaches must be made which generate radar imagery with the same features as synthetic aperture radar at reduced modeling cost. In order to perform these tasks imagery is generated in Digital Imaging Remote Sensing Image Generation (DIRSIG) and used as a simple Radar Cross Section map. This forms the basis of resultant pixel intensities in the final image. A histogram operation is performed to convert a forward looking image with no radar shadow into the typical top down view of a radar image. Next the simulated impulse response of a SAR system is applied to the image and finally it is fed into a conditional generative network to apply radar specific features.

Testing is performed in two sections. First by using t-distributed Stochastic Neighbor Embedding to visualize the distribution in high dimensional space then by training Automated Target Recognition (ATR) networks on the images of the MSTAR dataset with varying levels of simulated images mixed into the training set. *If the performance of the network remains the same or improves with simulated imagery over that of using only real images then the simulated imagery is sufficiently modeling collected data. However if performance of the ATR network drops significantly then the method of simulation is insufficient for the purposes of modeling to train a neural network.*

1.2 Synthetic Aperture Radar Basics

1.2.1 Synthetic Aperture Radar Image Chain

Synthetic Aperture Radar (SAR) is an imaging method which uses an active transmitter to illuminate an area on the ground (scene) with a radio signal as it travels. This transmitted radio wave then interacts with the scene and propagates to the receiver. Objects in the scene which have a higher return signal strength compared to objects at the same range are said to have a higher ”radar cross section”. This radar cross section is material-dependent. A receiving antenna, which is typically monostatic and coincident in time with the transmitter, then receives the returned signal which now holds information about the radar cross section distribution of the scene on the ground due to interaction with the transmitted signal. The received signal is then mixed with a copy of the transmitted signal in what is called ”demodulation” and then the demodulated signal is sampled from analog to digital and recorded into memory [1]. This is called the ”phase history”. At a later time, the phase history is processed using one of a number of algorithms to form an image of the scene which can be displayed to a person. A portion of phase history and a backprojection focused image from AFRL’s Wide Angle SAR dataset is presented in Figure 1.1. SAR processing algorithms are very similar to those used in computer aided tomography and magnetic resonance imaging.

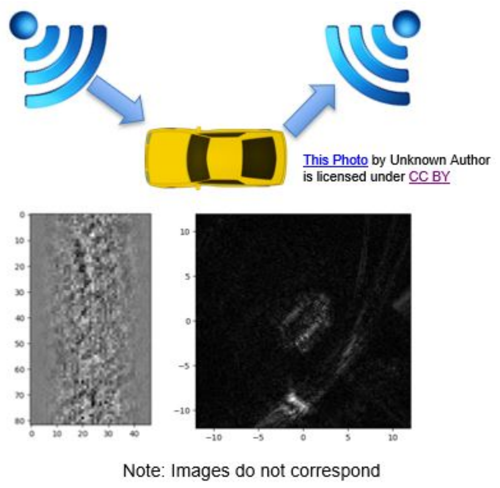
This Photo by Unknown Author is licensed under CC BY
Note: Images do not correspond

Figure 1.1: SAR system Image Chain

[1]Detailed discussion of demodulation can be found in MIT open courseware at `http://web.mit.edu/6.02/www/s2012/handouts/14.pdf`

1.2.2 Radar Basics by Simulation

For a detailed explanation of Synthetic aperture radar image basics, it is recommended the reader follow *Understanding Synthetic Aperture Radar Images* [4] by Chris Oliver and Shaun Quegan as well as *SAR image formation toolbox for MATLAB* [5] by LeRoy A. Gorham and Linda J. Moore. An abbreviated adaptation is included here for the purpose of deriving a basic Impulse Response (IPR) of a SAR sensor for use in image simulations.

The transmitted signal of a SAR platform, $s(t)$, is typically a linear frequency modulated chirp of the form:

$$s(t) = Ae^{-2\pi j(f_0 t - \gamma t^2)}$$

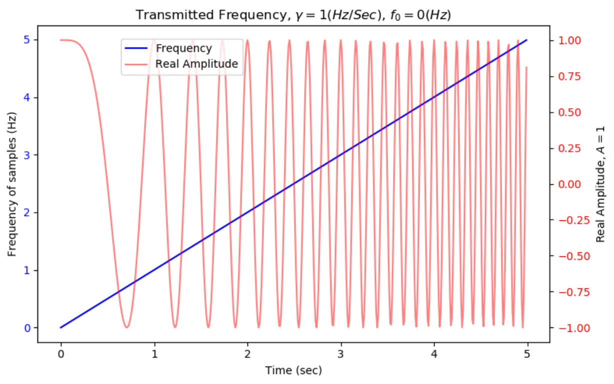

Figure 1.2: Example of LFM chirp, $A = 1, \gamma = 1, f_0 = 0$. For display only, published MSTAR value for f_c is 9.6GHz.

Where f_0 is the minimum frequency in units of Hertz and γ is the chirprate in units of $1/s/s$. With no loss in generality and only a slight change in processing, the center frequency f_c can be used in place of f_0 [2]. The time, t, it takes for the transmitted signal to travel a distance r to a target and back is determined by the target's distance by the equation $t_r = \frac{2r}{c}$. The signal is sampled at discrete points $t_k | k = 0, 1, ...K$, thus the frequency associated with each sample is:

$$t_k | k = 0, 1, ...K$$

$$f_k = \gamma t - f_0 = \gamma t_k - f_0$$

[2]When processing using the center frequency instead of the minimum frequency, time samples are denoted as $t_k | k = \frac{-K}{2} ... -1, 0, 1, ... \frac{K-1}{2}$. Using center frequency also requires changes to formulas for resolution, scene extent, and phase reference.

We can then modify the equation for the transmitted signal by incorporating these frequency samples as:

$$s(t, t_k) = Ae^{-2\pi j(f_0 - \gamma t_k)(t)}$$

$$s(r, t_k) = A\delta(t_k - \frac{2r}{c})e^{-2\pi j(f_0 - \gamma t_k)(\frac{2r}{c})}$$

Furthermore, the bandwidth $\mathbf{B}$ of the system is given by:

$$\mathbf{B} = \gamma t_K = \gamma K \Delta t = K \Delta f$$

Using the Nyquist criterion, we can assume that the Analog to Digital sample rate of the system is at least double the bandwidth and dictates the relationship $\Delta t \leq \frac{1}{2B}$. This frequency cutoff is associated with a range in scene. This creates a number of trade-offs between chirp rate and sample rate, primarily being that the change in frequency $\Delta f = \frac{B}{K}$ between samples also dictates the maximum size of the imaged scene before aliasing, called the alias free range extent, by [5]

$$W_r = \frac{c}{2\Delta f}$$

For the purposes of simulating an impulse response for the sensor which recorded the Moving and Stationary Target Acquisition and Recognition (MSTAR) dataset, the choice was made to match the published characteristics in the headers of the MSTAR dataset by assuming that the alias free range extent (differential range of the frequency bin associated with the nyquist cutoff) is greater than the size of the image in both the range and azimuth directions.

A similar limitation exists regarding sampling in the cross range direction, specifically in that the alias free cross range extent is determined by the angular distance $\Delta\theta$ between pulses and center frequency [5].

$$W_x = \frac{c}{2f_c \Delta\theta}$$

Then using the range and cross range resolution formulas[5] the parameters for simulation construction can be derived. In particular the user can use the bandwidth and center frequency with the given resolution of the image to select an appropriate number of samples and pulses filling a particular angle in order to simulate a phase history.

$$\delta_r = \frac{c}{2B} = \frac{c}{2(K)\Delta f}$$

$$\delta_x = \frac{c}{2f_c\theta} = \frac{c}{2f_c N_p \Delta\theta}$$

For the advantage of simplicity, the assumption is made that the platform maintains a constant elevation during imaging and traverses some angle θ along a circular path parallel to the XY plane. [3] Using this assumption we can use a fixed range reference and differential range (relative range from scene center, r_c). Thus $r = r_c + dr$.

Since the range to scene center remains constant for every pulse and the differential range to a target at scene center is zero, it is advantageous for us to shift variables such that the sample times are about the scene center. This means that our sample times are now relative to the return time from an object at scene center, $\frac{2r_c}{c}$.

$$t_{dr} = t_k + \frac{2(r_c + dr)}{c}$$

$$t = \frac{2(r_c + dr)}{c} \rightarrow t = \frac{2dr}{c}$$

Thus our phase history for the spotlight synthetic aperture radar is simply:

$$s(t_{dr}) = A\delta(t_{dr})e^{-2\pi j(f_0 - \gamma t_{dr})(\frac{2dr}{c})}$$

$$s(t_{dr=0}) = A\delta(t_{dr=0})e^{-2\pi j(f_0 - \gamma t_{dr=0})(\frac{0}{c})} = A\delta(t_{dr=0})e^0 = A\delta(t_{dr=0})$$

While having a constant value for every sample of the phase history present during the pulse duration is very useful, the most important parameter is that we were able to choose platform locations for each pulse and can begin image formation at any point in time.

For image formation, it is necessary to determine the XYZ coordinates of points that make up the image, called the image plane, and the algorithm to be used to focus the image. The RITSAR package on github [6] by Doug McDonald features functions to perform creation of the image plane and multiple algorithms used to focus SAR imagery. This package was updated to work with Python 3 and included/modified functions from the Matlab SAR toolbox [5] to assist in determining the impulse response of a particular SAR system[4].

Incorporation of methods from the Matlab SAR toolbox[5] into RITSAR allows the user to set the sample spacing of the final image to that of the MSTAR dataset and use the backprojection algorithm to form an image of the impulse response. After the XYZ coordinates of each pixel in the image plane are selected, backprojection works by first applying a taylor weighted window then taking the Fourier transform of the received pulse return and interpolating it at the calculated differential range of each pixel. The Impulse response found is located in Figure 1.3

[3]While no position or phase history information is given with the MSTAR dataset, this assumption is based on AFRL's wide angle SAR dataset.

[4]This updated package can be found at `https://github.com/jks7592/RITSAR`

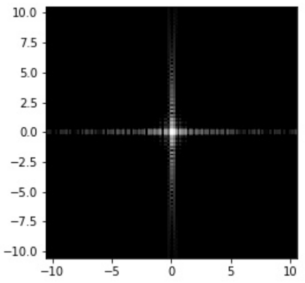

Figure 1.3: SAR system Impulse Response

In order to apply this IPR to any DIRSIG simulated image later, we will have to perform a 2D convolution between the IPR and the image. For efficiency this is best done in the frequency domain as a simple multiplication can be performed. Thus for application purposes what we really want is the Modulation Transfer Function (MTF) in order to multiply with the Fourier transform of the image. An example of the MTF is shown in Figure 1.4.

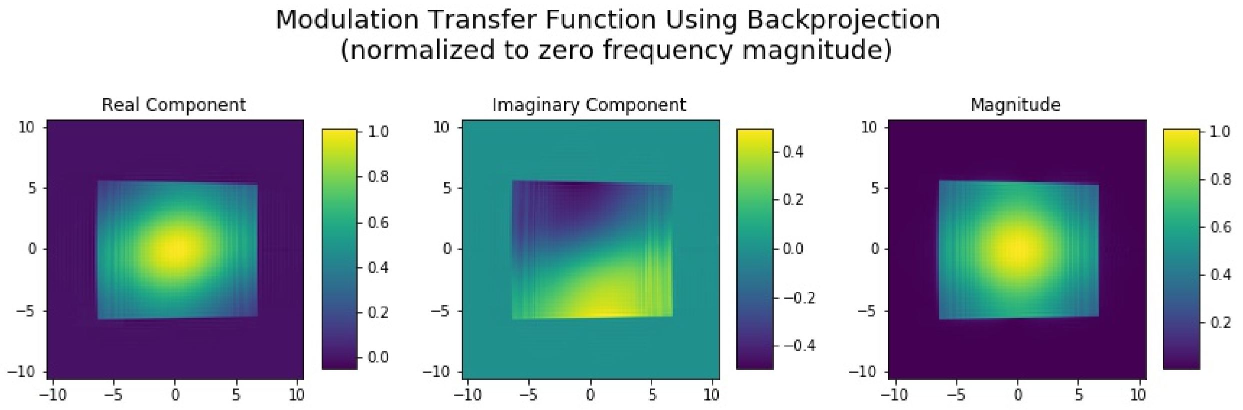

Figure 1.4: Ideal SAR system Modulation Transfer Function. Also called a Windowed Annular Ring Aperture in *Anatomy of a SAR Impulse Response*[1]

1.3 MSTAR and SAMPLE

Published by Air Force Research Labs (AFRL), the Moving and Stationary Target Acquisition and Recognition (MSTAR) dataset has led to a number of advancements in Automated Target Recognition (ATR) [7]. It consists of X-band SAR image chips of tanks, military vehicles, and non-military machinery. MSTAR data was collected by Sandia National Labs between September

1995 and November 1996 by the STARLOS sensor. It was collected at a 1 foot resolution in spotlight mode. This dataset has been the cornerstone for SAR target recognition in the same way that MNIST handwritten digit database has been the cornerstone for basic machine learning. State of the art algorithms report a 99.46% accuracy in discriminating images taken with a 15° angle of elevation when trained on images taken with 17° elevation [8]. The images somewhat form a 360 profile around each target with between 233 and 698 images per target taken at 17° and 195 to 587 images per target at 15°. Although not used here, another section of the MSTAR dataset consists of eight different variants of the T-72 tank.

It takes a very significant amount of time and cost to develop a dataset such as MSTAR, and it is unrealistic to collect data on any target across the entire SAR operating domain. In the case of finding real world military targets of interest it is also impractical due to the targets themselves being denied. A cheaper and more feasible solution is to use radar modeling software and a 3D modeled approximation to make a synthetic response from the target of interest to train an ATR model. These considerations beg about the question as to when the radar modeling parameters and the 3D models themselves are close enough to actual collected data to train a model for real world discrimination. AFRL's Synthetic and Measured Paired Labeled Experiment (SAMPLE) [9] was built to address this concern by modeling to the best of their ability a collection of targets from the MSTAR dataset and using them in a number of experiments including to train a neural network to be tested on real data. The authors of SAMPLE also demonstrated the use of t-distributed Stochastic Neighbor Embedding (t-SNE) [10] to show that the simulated images clustered together separately from their MSTAR counterparts and used the replacement of measured images with synthetic to demonstrate simulated data fidelity.

The authors of SAMPLE suggest possible methods to close the gap between the simulated and real data, one of which is by using neural networks to adapt the synthetic images into the domain of real images. This was later performed in *Generative adversarial networks for SAR image realism* [11] where DualGAN (a variant of CycleGAN) was used for domain adaptation to better adapt synthetic images to the SAR domain. In this paper a classifier was trained and tested on threshold masks of the output images with promising results, though results on the images themselves are not presented.

1.4 Dimensionality Reduction Basics

Dimensionality reduction is simply reducing the number of variables under consideration. Ultimately every classification problem can be thought of as dimensionality reduction, but is plagued by the accuracy of said reduction. Take for example a group of students in a class. They all take tests, do homework, and give presentations. Each student is a sample of the class, and each student's homework grade is an attribute of their sample. Thus each sample has many attributes and these attributes are reduced to one dimension, the student's final grade in the class. The mechanism for the dimensionality reduction in this case is the weights given to each assignment grade to sum up the final grade. One can even say that there is an "activation function" applicable between the students' final numerical grade and final letter grade as ranges of the final numerical grade are assigned to different named classes.

An image can easily be described as an $M \times N \times C$ matrix of elements, where M and N represent the height and width of the image in pixels and C represents the number of channels. For the purposes of this paper, only one channel is used so we can reduce this matrix to $M \times N$. One can also unwrap the image into a single vector of length MN, and this vector then represents the image in a very high dimensional space. Often these vectors in very high dimensional space are too long (ex length 16384) for algorithms that increase computational complexity with the number of dimensions. A reduction down to a lower dimensional space (ex length 500) loses information, but if the right information is kept it can often sufficiently represent the higher dimensional image.

The simplest dimensionality reduction technique for collections of images is called Principle Components Analysis (PCA) [12], and can easily be applied to a vast assortment of situations. It simply consists of decomposing a collection of the very high dimensional vectors which represent images in the space using a Singular Value Decomposition [13] and then keeping a desired number of the singular vectors. Any matrix $\mathbf{M}$ can be decomposed using Singular Value Decomposition.

$$\mathbf{M} = \mathbf{U\Sigma V}^*$$

The columns of $\mathbf{U}$ are called left singular vectors and their corresponding singular values represent the variation in the matrix with respect to each singular vector. A matrix of a select number of these singular vectors can then be used as a transform to keep a known amount of the variation in the original space in a significantly reduced form. This method however is vulnerable to simple shifts in input images as it completely disregards small local features in the image. However, if a dimensionality reduction instead uses local features like a neural network, the effects of simple shifts are no longer detrimental.

An important tool for visualization used in this work is called t-distributed Stochastic Neighbor Embedding (t-SNE) [10]. It is a dimensionality reduction technique for visualization which preserves visually the distances between points in high dimensional space on a two or three dimensional grid. T-SNE is commonly used for visualization of data in machine learning applications [14]. The scale and axes have no meaning and the representation is not necessarily repeatable due to random ordered selection and probabilistic processes involved, but it conveys which points are close to each other in high dimensional space. The technique is however computationally complex on data with too many dimensions and thus for my resources requires some other dimensionality reduction before its use.

1.5 Convolutional Neural Network Basics

A neural network can be simply described by a collection of operations performed in sequence on an input in order to produce some desired output. The simplest of neural networks output a label associated with features of the given input. Features in the case of images are typically demonstrated by the use of various filters around each pixel in the image such as in the example below. Each layer of a neural network applies a great many of these filters simultaneously.

8	6	2	5	9	7	1	4	3
9	4	5	1	3	6	7	2	8
3	1	7	2	8	4	5	6	9
2	7	3	4	1	9	8	5	6
5	8	1	3	6	2	9	7	4
4	9	6	7	5	8	3	1	2
1	5	4	8	2	3	6	9	7
6	2	8	9	7	5	4	3	1
7	3	9	6	4	1	2	8	5

(a) Simulated Image (Sudoku) with example filter location boxed [15]

1	1	1
1	1	1
1	1	1

(b) Simple 3×3 Filter

45	33	42	45	50	42	45
41	34	34	38	51	52	56
37	36	35	$\boxed{39}$	52	55	59
45	48	36	45	51	52	45
43	51	42	44	44	48	48
45	58	56	54	43	42	36
45	54	57	45	34	41	45

(c) Filtered Image with example result boxed

While the user dictates the filter size, the filters are not typically defined explicitly by the user but instead 'learned' through training by calculating a gradient decent based on examples of paired inputs and outputs desired from the network. These gradients are first calculated implicitly during a forward pass through the network then explicitly in a backward pass after the calculated error (loss) has been determined. Losses come in many different types depending on application whether it is based on the output classification or a difference in pixel values between input and output images.

More complex network architectures called convolutional neural networks (CNNs) have starting layers which perform the same local operation across an entire image by using pixel values from a small region around each pixel. This is called a convolutional layer. Typically the next layer pools together neighboring results and saves the maximum value. Subsequent layers can then use the pooled data to perform different operations on large areas of the input image while the final layers use information gathered from the entire image to make a decision on the final output.

Some networks such as DenseNet [16] maintain the outcome of all previous layers and use them as inputs in subsequent operations.

A generative adversarial network (GAN) is a type of neural network training architecture which uses multiple neural networks in competition in order to incrementally perform better on a given task [17]. First a ”Generator Network” is given some input from a random number generator and it performs a large series of operations to create some output. Next a ”Discriminator Network” is given this output and examples from the domain of what the generator is attempting to create and determines if the generated output is ”real” or ”fake”. The outcome of the discriminator then determines how the generator needs to be improved.

The GAN ’pix2pix’ [2] adds another layer of complexity in that instead of the discriminator having a single output ”real” or ”fake” it instead gives a judgement on each region of the image in order to encourage high frequency content. Referred to in Isola et. al [2] as PatchGAN, this allows the network to go from one domain to another while learning fine details of the output domain. An example is black & white to color in Figure 1.5. Code for this feature was not included in CycleGAN as it is replaced by the cycle consistency loss described below.

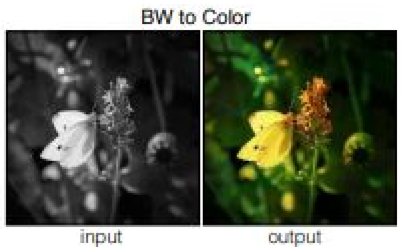

Figure 1.5: Example of Black and White to Color from Isola et. al 2016 *Image-to-Image Translation with Conditional Adversarial Networks* (pix2pix) [2].

For CycleGAN [18] [19], two generators $\mathscr{G}_{AB}, \mathscr{G}_{BA}$ and two discriminators $\mathscr{D}_A, \mathscr{D}_B$ are used to develop a two-way transition between domains A (ex. Horses) and B (ex. Zebras). The loss functions include a ”cycle consistency loss” which pushes the networks towards learning transforms in which a cycle between the two returns similar values.

$$\mathscr{G}_{BA}\{\mathscr{G}_{AB}\{\mathbf{X}_A\}\} \rightarrow \mathbf{X}_A$$

$$\mathscr{G}_{AB}\{\mathscr{G}_{BA}\{\mathbf{X}_B\}\} \rightarrow \mathbf{X}_B$$

There is also an ”identity loss” which uses the difference of pixel intensities to push the network such:

$$\mathscr{G}_{AB}\{\mathbf{X}_B\} \rightarrow \mathbf{X}_B$$

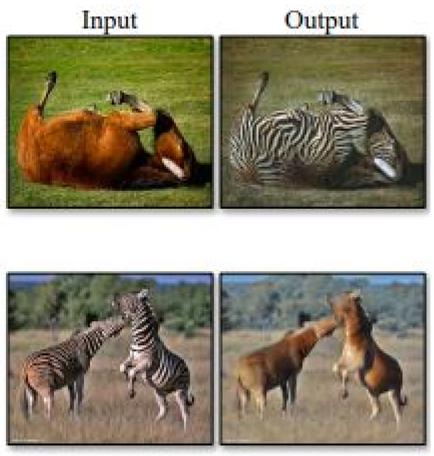

Figure 1.6: Example of horses to zebras transition from Zhu et. al 2018 *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks*

$$\mathscr{G}_{BA}\{\mathbf{X}_A\} \rightarrow \mathbf{X}_A$$

Finally included is the "GAN Loss" which uses the mean squared error between the transformed image and the target image in order to construct the the transformation.

$$\mathscr{G}_{AB}\{\mathbf{X}_A\} \rightarrow \mathbf{X}_B$$

$$\mathscr{G}_{BA}\{\mathbf{X}_B\} \rightarrow \mathbf{X}_A$$

The combination of these allows for unpaired image translation, thus sophisticated matching between input and output domains is no longer essential. This allows for transitions between settings which no longer require being strictly paired, and still learns exceptional outputs such as in Figure 2.3.

1.5.1 Neural Networks for Augmentation/Generation of Datasets

Algorithm accuracy is typically limited by the quality and quantity of the data used to train the model. When getting additional real data is cost prohibitive, the first thing to turn to is dataset augmentation of already collected data. There is a wide variety of methods for augmentation from simple image translation, flips, and cropping to more complex operations such as adding noise. With these augmentations the features present in the data remain largely the same, while modification by generative network instead imparts features into the seed image which were not previously present. Using a generative adversarial network to augment data has already proven itself in medical imaging applications [20] [21] [22] [23] [24]. The barrier in using a generative adversarial network however is that sometimes output images exhibit features which are completely outside the target domain. Images must be reviewed after generation to ensure that the output image does not exhibit unrealistic features.

1.5.2 Neural Networks for Dimensionality Reduction

Neural Networks operate by warping the distribution they were trained on such that samples from different classes become far apart [14]. In this process, neural networks tend to step down in size until fully connected layers near the end represent a vector of significantly reduced size compared to the original image. This reduction can be simply harnessed by training a neural network on data from a similar domain to the data you want to reduce and simply removing the final classification layer. The neural network then warps the input data such that dissimilar samples are non-adjacent, while ones that display the same features are represented close to one another. For this purpose, a modified DenseNet [16] was trained on MSTAR data collected at 17 degrees depression and tested on data collected at 15 degrees depression. As this network is used for dimensionality reduction, the network was saved during each of 10 training epochs and selected the epoch (4th) with the best testing accuracy of 95.7%. Removing the last layer, DenseNet was modified to reduce the images from size $128 \times 128 = 16384$ to a vectors of length 512. This process enables the network to discriminate distinct features of the MSTAR domain. While it's possible to use an untrained neural network, this would give a wide range of inputs a similar output. Figure 1.7 shows the difference between using a trained and untrained networks for dimensionality reduction on a selection of the training set. If the GAN is modifying the images into the same domain as MSTAR reliably we should see output images reduced to be near (or intermixed with) those in the corresponding portion of the MSTAR dataset, and distant from the unprocessed images.

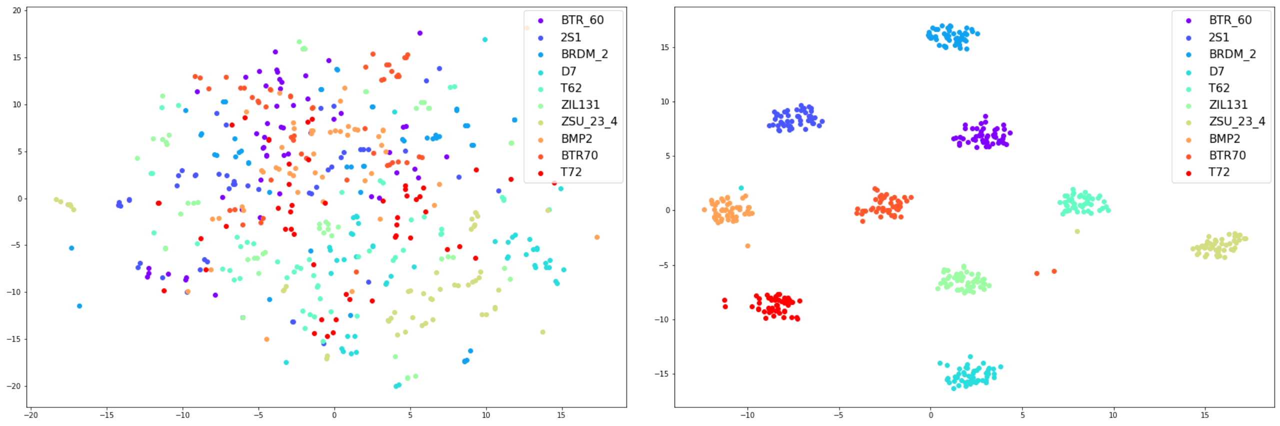

Figure 1.7: Dimensionality Reduction of 50 samples from each class of training set using untrained and trained DenseNet.

Chapter 2

Research Procedures

2.1 DIRSIG

3D models of the BMP2, BTR70, T62, T72, ZIL131, and ZSU_23_4 which were used to create the SAMPLE dataset were acquired from Air Force Research Labs. These models were originally in the AFacet [25] format and converted into .obj files using the AFacet2DIRSIG converter tool written by Jacob Westerkamp [26]. Models were then placed into simple scenes for generation with Digital Imaging Remote Sensing Image Generation (DIRSIG) [27] [28] in order to make a radar cross section like map from the view of the sensor. Scenes consisted of a standard blank background with the target which rotated through 360 degrees in single degree increments. Simulated images used a panchromatic camera with a 256x256 pixel focal plane to create the "Radar Cross Section" map. In order to match the MSTAR training set parameters, angle of source and angle to collection platform were selected to 73° off zenith (17° from horizontal) and co-oriented such that no shadow would be present in the image. The primary advantage of DIRSIG was its ability to output truth data for every pixel in scene, particularly the XYZ coordinates and view angle cosine were used so that the image could be shifted from the view of the sensor to the typical top-down view of a processed SAR image. While atmospheric affects can be modeled by DIRSIG, this feature was ignored as its affect is often negligible for the SAR domain.

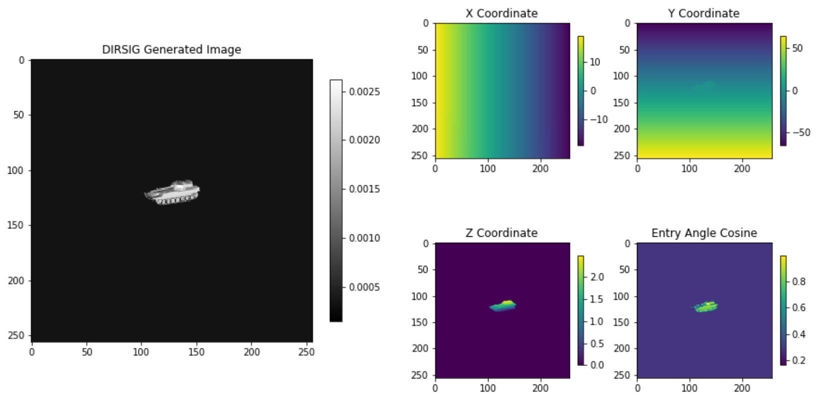

Figure 2.1: Left: DIRSIG output image. Right: Cartesian coordinate truth data and view angle cosine (also cosine of "Avg Entry Angle" in DIRSIG 4)

2.2 Simulated Image Pre-processing (Radarize)

For pre-processing, the simulated images were converted to a 'top-down' view by way of a histogram operation, hereafter referred to as being Radarized. First, a vector was determined from platform center to scene center then projected into the XY-plane and normalized. This is considered the ground range or $\hat{\mathbf{u}}$ direction. Next to be calculated is the perpendicular vector in the XY-plane called the ground cross range, $\hat{\mathbf{v}}$. The platform position is subtracted from the XYZ coordinates of every pixel in scene to give a vector from the platform to each pixel and range is calculated by the distance formula $r = \sqrt{(X - X_p)^2 + (Y - Y_p)^2 + (Z - Z_p)^2}$. The cross range coordinate for each pixel is calculated by dot product with $\hat{\mathbf{v}}$. The pixel value itself is then multiplied by a power of the view angle cosine to form the pixel weight to be used in the 2d histogram. A cosine weight of 1 was chosen as the human visual contrast between background and shadow matched most closely, note that pixels were also cosine weighted by DIRSIG during generation. A range and cross range grid is established about the scene center which matches that of images in the MSTAR dataset, specifically a ground pixel sample spacing of .202148 meters in the range direction and .203125 meters in the cross range direction. The 'histogram2d' function from the Numpy package [29] is then used to place pixel weights into their associated range and cross range bins. Finally, the Fourier transform of the output histogram is taken and then multiplied by the modulation transfer function discussed in Chapter 1, an inverse Fourier transform is then used to complete the image generation. This multiplication by the modulation transfer function in the Fourier domain applies the SAR impulse response from Figure 1.3 to finish the Radarized image.

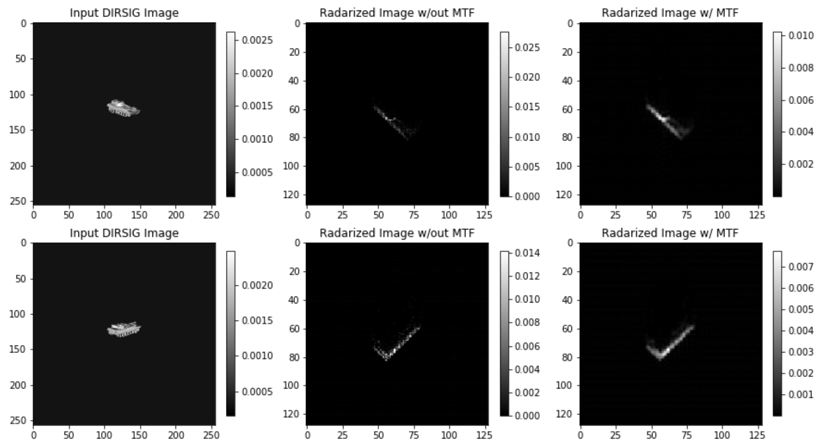

Figure 2.2: Radarize process from input to histogram operation to application of MTF. Note changes in both image shape and pixel values.

2.3 Dataset Preparation

For MSTAR images only the magnitude channel was used and *the phase channel was discarded*. Multiple instances of training were performed with various instances of the generated images. First, the Radarized images were paired with their closest corresponding MSTAR image (see Figure 3.1 in Chapter 3) by a simple nearest neighbor algorithm with a cross check to ensure matching. During training instance 1 all images including the 2S1 class were used in order to test if the network could match features when examples of the desired output are present. Second, the network was retrained with the class 2S1 removed from the strictly paired training set and kept in reserve in order to facilitate testing and see how well paired images could learn the domain without the target class present. Third, images were left unpaired but the 2S1 class was still not present. This unpaired method allowed the use of images from the generated data for which there was no corresponding MSTAR image as well as MSTAR images for which there was no model. For each of the three instances, MSTAR images were center cropped to 128 by 128 pixels if required and then stacked and saved into a numpy array. The following torchvision [30] transforms were performed in sequence during training instances one and two: ToPILImage(), CenterCrop(128), ToTensor(), Normalize((0.5,), (0.5,)). The transform RandomHorizontalFlip() was added during training instance three since images were unpaired.

2.4 Generative Adversarial Network

Cyclegan [18] was chosen for its ability to perform unpaired image translation and adaptability for different training methods. Code from `github.com/aitorzip/PyTorch-CycleGAN` was used with modifications. The network architecture can be seen in Figure 2.3 [1]. Primarily, modifications were made to adapt to the new dataset and to remove the dependency on the python ”visdom” package. During the first few training runs it was noticed that the model was prone to mode collapse. Mode collapse is described by the network learning a single output or family of outputs that cheat the discriminator and using them repetitively regardless of input. With the assistance of Ryne Roady, modifications to the architecture were made to implement Generative Adversarial Network regularization from *Which Training Methods for GANs do actually Converge?* [31] and prevent mode collapse. An actual Generator cycle example can be found in Figure 2.4.

[1]`https://hardikbansal.github.io/CycleGANBlog/`

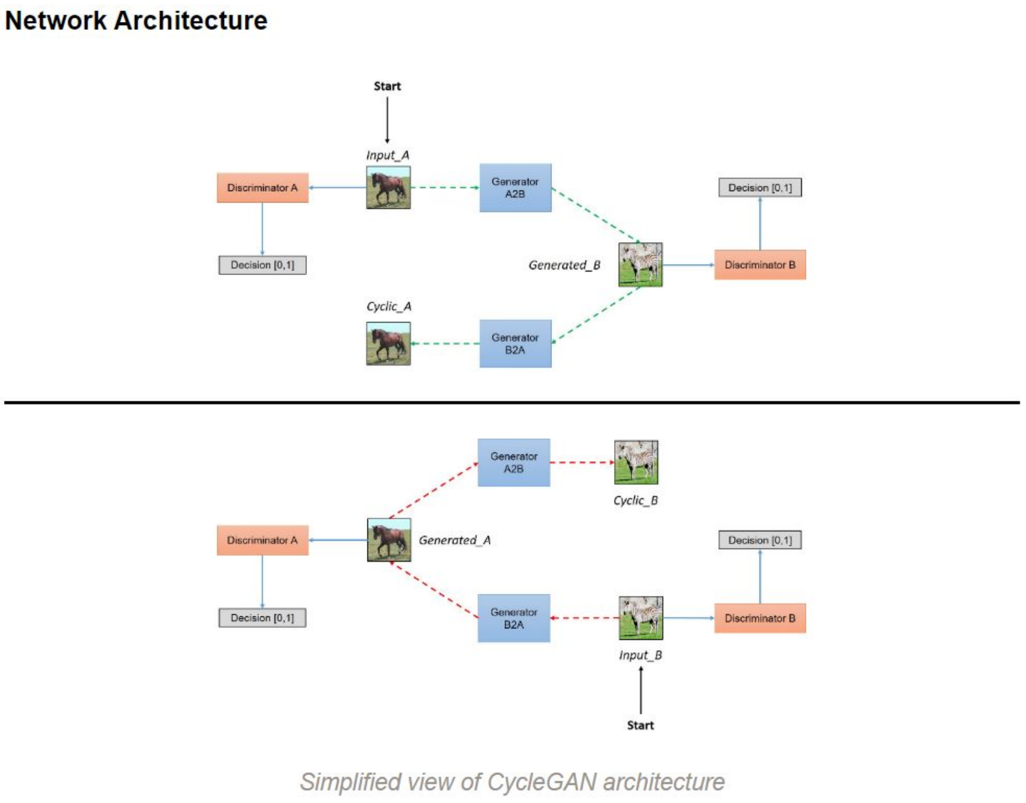

Figure 2.3: CycleGAN Architecture from github CycleGANBlog [3].

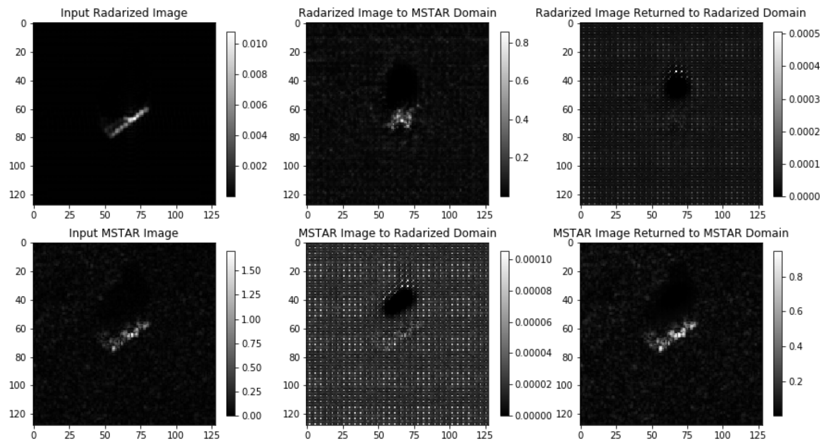

Figure 2.4: Inputs and outputs from CycleGAN architecture including those which return the input images to their original domain.

2.4.1 Training the GAN

Training begins with gradients deactivated for the discriminator and activated for the generator networks. After the torchvision transforms, the images are passed through the generator network to create fake images, $\mathbf{Y}$, in the output domain of each network.

$$\mathscr{G}_{AB}\{\mathbf{X}_A\} = \mathbf{Y}_B$$

$$\mathscr{G}_{BA}\{\mathbf{X}_B\} = \mathbf{Y}_A$$

The Discriminator network is then used to provide an estimate between zero (fake) and one (real) of the generated images. The target (all pixels appearing to be real) is used with this determination and used to calculate the ”GAN loss” using a mean squared error to improve upon the Generator.

The real images of each domain are passed through the networks which represent transitions into their own domain and an ”Identity loss” is calculated using an L1 (least absolute deviation) metric.

Next, the generated (fake) images are passed through the opposite generator in order to create a ”recovered image” from their own domain and the ”Cycle Loss” is calculated using another L1 loss between the original and recovered images.

Identity, GAN, and Cycle losses are summed and since gradients were activated for the generator, they can then be used to improve the generator. Gradients are then deactivated on the generators and activated on the discriminator networks. The discriminators are again given the generated images and used to give a per pixel mask of determinations toward real or fake. This time the mean squared error loss is used with the truth value of real or fake to improve the discriminators. Gradient regularization is used at this step to prevent mode collapse of the network. All networks are then saved at the end of every 10th epoch.

2.4.2 Testing the GAN

The dataset for testing is limited to 2S1 Radarized images from those strictly paired with SAMPLE to test each of the saved Generator network coefficients. Testing is performed by saving the output images of the generator networks to a different file for each saved network weights file.

In testing part 1, images from every group (GAN, MSTAR, SAMPLE, Radarized) are converted to vectors of length 512 by way of using all except the last layer of a DenseNet neural network trained on real MSTAR data. This vector is then bundled with reduced vectors from generated data by other network coefficients as well as reduced 2S1 images from MSTAR and SAMPLE (both real and synthetic). The collection of these vectors is finally used with the t-SNE algorithm to create a 2D plot of their relative distances in high dimensional space. The best performing network coefficients are selected for continuation to testing part 2. A selection of paired images is then chosen to facilitate visual comparisons between the images. Matplotlib's monotonically increasing 'gray' colormap is used to display images with a linear mapping look-up-table between minimum and maximum values in each image. Colorbars are given next to each image to illustrate intensity value scaling.

Testing part 2 includes incorporating these generated images of the 2S1 class into the training set of neural networks labeled as '2S1' and testing the performance of the network with the augmented training set while monitoring the performance of the 2S1 class. If the Generative network has successfully learned to incorporate SAR specific features into the images than we should expect a gradual falloff to a non-zero value. Otherwise, the expectation is that testing performance will slowly decrease until the minimum number of images to reliably define the 2S1 class is remaining and then fall off sharply thereafter to a precise 0% accuracy.

1. Testing Part 1
   - Images from every group reduced to vectors with trained DenseNet
   - t-SNE used to create 2D plot of relative distances
   - Best performing networks chosen for Testing Part 2
2. Testing Part 2

- Replace training set ’2S1’ images with GAN ’2S1’ images
- Train ATR network on new set for 10 epochs
- Test ATR performance on test set
- Repeat with more images replaced

Chapter 3

Results

3.1 Testing preparation

In order to visualize the changes performed by the network, the common target orientations between MSTAR, SAMPLE, and Radarized DIRSIG images were identified for use with the Generative Adversarial Network. The classes of the SAMPLE dataset only overlap with the MSTAR dataset for a few of the vehicles, and each class presents at most 58 images instead of the 299 in most MSTAR classes. Luckily, the target azimuth is maintained in the meta-data for each source which makes matching the SAMPLE images with their MSTAR origin simple. For the Radarized images the target azimuth is in increments of 1 degree, while each of the other sources have somewhat irregular spacing.

The simplest method to pair the images was to build a nearest neighbors algorithm using the target azimuth. Taking this further, the situation may occur such that the nearest neighbor to 359.7° should be 0° and not 345°, and thus the degree measurements were converted into a vector of their position on the unit circle and then the bi-directional nearest neighbor was selected from that point. Bi-directional in this case meaning that for $a \in A \& b \in B$ that the nearest point in B to a is b and the nearest point to b in A is a. This pairing method can also be called 'nearest neighbor with crosscheck'. The '2S1' and 'ZSU_23_4' classes were the largest remaining with 58 images each and thus the '2S1' class was chosen as the exemplar testing class. A set of paired images from each of the MSTAR, SAMPLE Real(MSTAR with small integer pixel shift), SAMPLE synthetic, and Radarized can be seen in Figure 3.1.

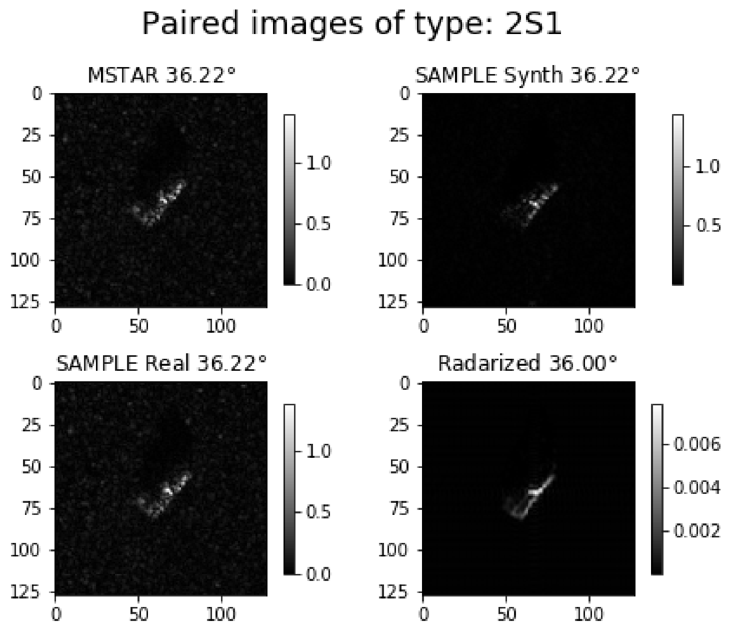

Figure 3.1: Images from MSTAR, SAMPLE, and Radarized output.

After pairing was complete, the Radarized images were saved to be used as a comparison case for the generative network so that comparisons could be made across all of the domains including the real and synthetic images from SAMPLE. At first, PCA was used to reduce the dimensionality of the images enough to be used with t-SNE and then it was realized that the authors of SAMPLE used Random Sample Consensus (RANSAC) on the real images of MSTAR to determine small pixel shifts to align better with their simulated images. At a human visual level the images appear identical. But for the purposes of PCA on unwrapped images this was a significant problem to use with t-SNE as vectors for the exact same image became very distant from each other. It was discovered by the researcher at this time that the typical method to use with t-SNE is to have a trained neural network perform the dimensionality reduction. The authors of SAMPLE did maintain the un-shifted images with the shifted ones and the decision was made to use the shifted ones for demonstration of the robustness to shifts during dimensionality reduction by neural network. DenseNet was trained on the real MSTAR images and all except its last layer was used for the dimensionality reduction.

3.2 Phase 1: t-SNE Visualization

The GAN was trained in three separate instances to represent three common situations for using a GAN to augment training data:

First, one in which all of the paired data was used including the 2S1 class. This represents the situation in which some data of the desired target is available, pairs have been found between data sources, and more simulated data is desired. As you can see from Figure 3.2 the GAN is insufficient at bringing data into the MSTAR domain. This is evidenced by the fact that the ”T72” class from MSTAR and the simulated data from SAMPLE are much closer than any GAN modified images. What we can see however is that the GAN did bring unprocessed images much closer to those of the MSTAR dataset. Though still distant, it appears that training epoch 10 provides the closest match to the MSTAR domain and is used with the real data in training a neural network classifier in the second phase of testing. This is further evidenced by Figure 3.3 which displays the different network outputs at each epoch.

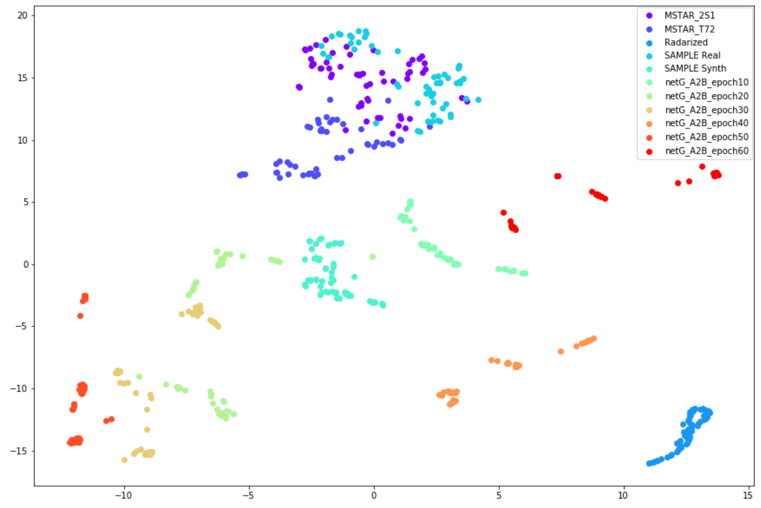

Figure 3.2: Instance one t-SNE reduction

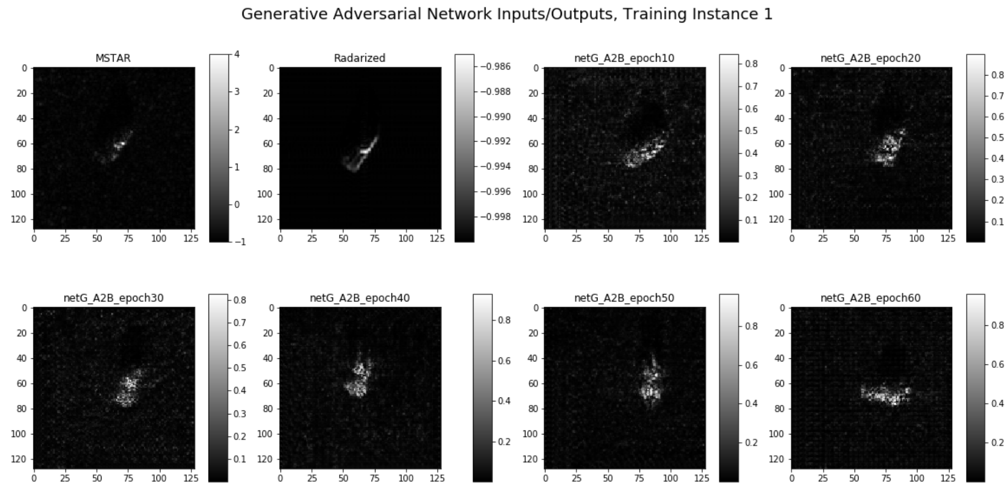

Figure 3.3: Example of training instance 1 inputs and outputs from GAN

For the second training instance, the 2S1 class was removed from the training set and the GAN was retrained. This represents the situation where paired data for other targets is available, but none of the targets of interest in the specified domain are present. The assumption in training this method is that a general transform is possible but paired data must be used. The reader can see in Figure 3.4 there is still a large gap between images of the MSTAR dataset and images from the generator. As expected the removal of the 2S1 class has made the network suffer, but only slightly. There are many possible reasons for this, first of which being that the methods used are insufficient as a whole but it is also possible that if the transition exists it may be more general than expected. As the network from epoch 10 was the closest in high dimensional space to MSTAR it was chosen for phase 2 of testing. Visual inspection of outputs in Figure 3.5 show that epoch 50 demonstrates a regularly spaced GAN artifact across the entire image and epoch 10 appears to show a spatial shift but overall likeness to the MSTAR image.

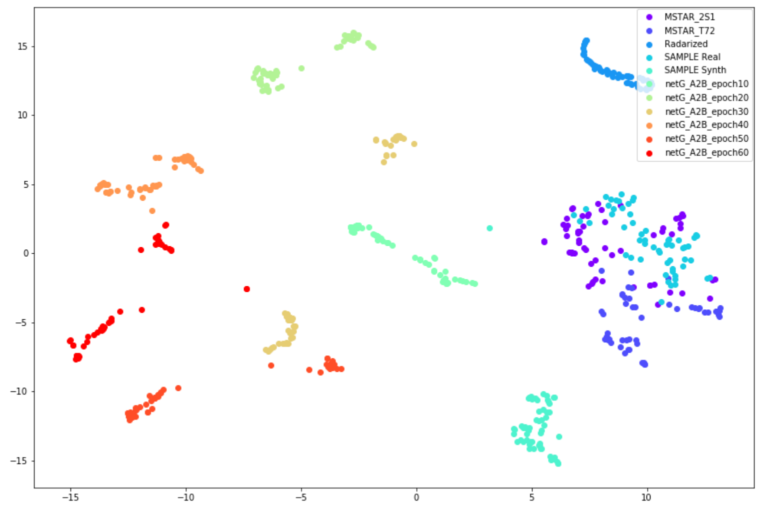

Figure 3.4: Instance two t-SNE reduction

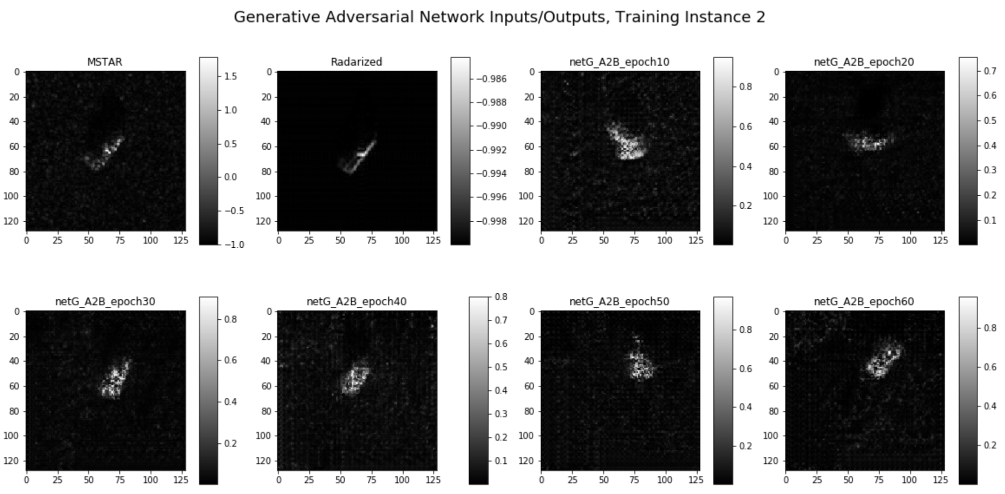

Figure 3.5: Example of training instance 2 inputs and outputs from GAN

In the third training instance, all data except the 2S1 class is used, but in an unpaired state. This represents the situation where a large amount of data is available from each domain but pairing the images is not feasible, and one wishes to see if a general transform between the modalities is possible. As the reader can see from Figure 3.6 the GAN outputs are still quite distant from the MSTAR data, but it appears that the epoch 20 output is starting to get as close to the MSTAR distribution as that of the synthetic radar data from SAMPLE. The opinion of the author is that this network, while still not reasonable for generating training data for a neural network, may be able to create radar-like images. Upon inspection of the actual outputs, epoch 20 no longer appears reasonable due to a grid like artifact on all output images.

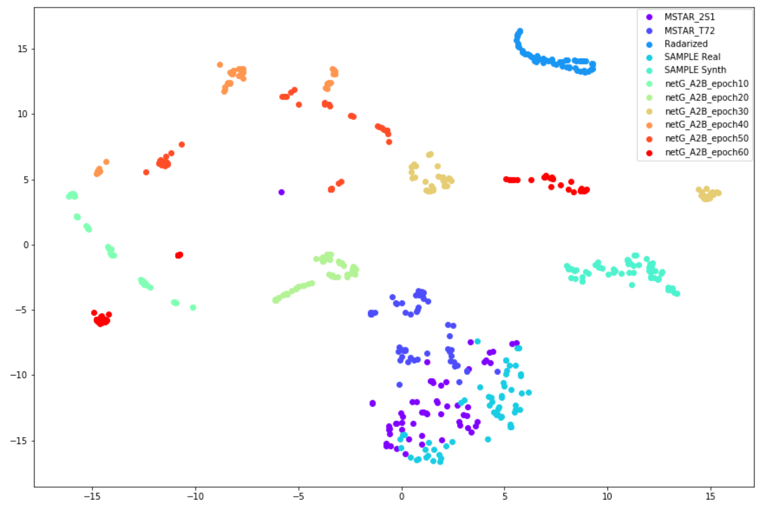

Figure 3.6: Instance three t-SNE reduction

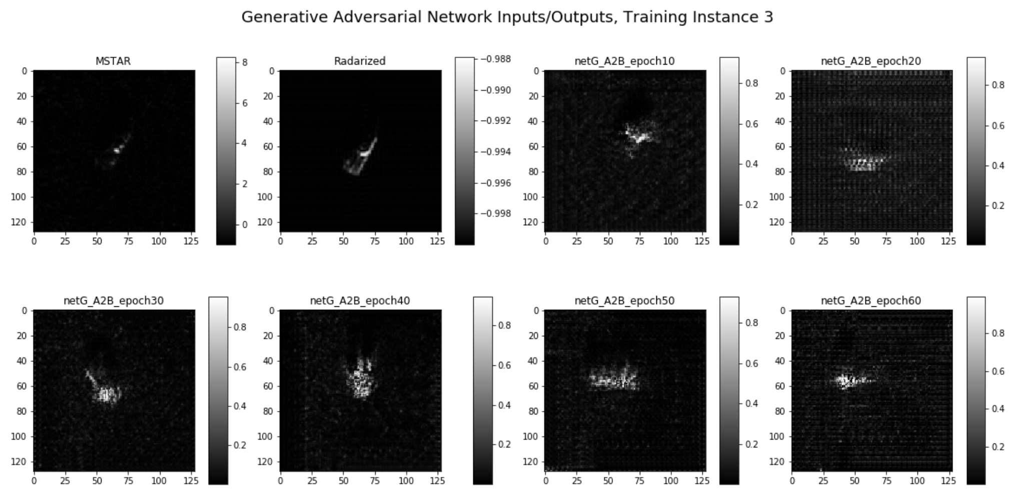

Figure 3.7: Example of training instance 3 inputs and outputs from GAN

3.3 Phase 2: Dataset Augmentation

The best performing network weights from each of the three first phase instances were kept in order to be used to augment the MSTAR training set. Two methods of augmentation are used, the first being to use the generated images as a replacement to the data already present in the dataset for the 2S1 class. The second method is to take advantage of the ”Identity Loss” used during training and modify a subset of the images already present in the dataset to replace members of the dataset. The choice to replace instead of append to the dataset was made based on concepts from simple machine learning algorithms. Specifically a K-nearest neighbors algorithm simply determines the distance between the target it is meant to classify and every member of the training set, then chooses the majority consensus of the K closest. Assuming that the new data is not representative of the domain, simply appending to the dataset would leave the already present data as the nearest neighbors and never affect the training outcome. Replacing data in the dataset however means that less and less of the previous neighbors are present. This would cause the training process to suffer if the new data is not representative but remain the same (or possibly improve) if it is.

ChaNet was chosen for its use in SAR Automated Target Recognition research and it's network parameters can be found in table 3.1. This is different from Cha et al. [32] in that the dropout layer was removed to facilitate more rapid training. Cross entropy loss was used for loss computations. The Adam optimizer was used with a learning rate of .002 and batch size of 32.

ChaNet	$1 \times 48 \times 48$
relu(Conv2D())	$9 \times 48 \times 48$
pool	$9 \times 24 \times 24$
relu(Conv2D())	$18 \times 24 \times 24$
pool	$18 \times 12 \times 12$
relu(Conv2D())	$36 \times 12 \times 12$
pool	$36 \times 6 \times 6$
relu(Conv2D())	$60 \times 6 \times 6$
flatten	1×2160
linear	1×60
linear	1×10

Table 3.1: ChaNet architecture starting with input layer through 10 output classifier layer.

First, the three winning generator weights from testing part one were each used to generate 361 images of the 2S1 class. A random selection of these was used to replace images in the 2S1 class of the MSTAR dataset. The number randomly selected varies from zero (no images replaced) to 300 (full replacement of the 299 MSTAR images) in increments of 25. Replacement image pixel values were scaled such that the maximum and minimum of the replacement images matched that of the MSTAR dataset 2S1 class. The network from Cha et al. seen in Figure 3.8 was used due to published results with SAR Automated Target Recognition, here referred to as ChaNet. ChaNet's training cycle consisted of ten epochs of training before testing then being reinitialized for training again. The resultant accuracy on 15° images in the MSTAR dataset is presented in Figure 3.8. Each value for number of images to replace was used ten times (blue dots) and the red dots represent the mean value of those 10 trials.

As the reader can see, 2S1 class accuracy drops only slightly when MSTAR dataset images still make up a majority of the class, then falls sharply thereafter to zero when the full collection of data is replaced. This clearly demonstrates that the output of the generator network on Radarized data is not representative of the real 2S1 class.

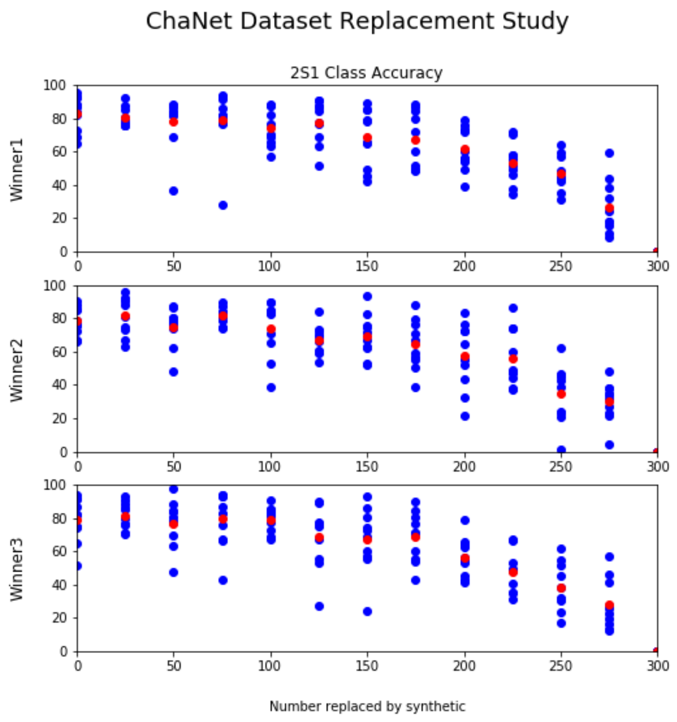

Figure 3.8: ChaNet accuracy results for training/testing cycle when images in 2S1 class replaced with images generated by GAN.

For comparison, a similar experiment with the SAMPLE synthetic data was performed. As the number of images is limited in the SAMPLE dataset, the real data collected at 16° was chosen as the testing set. Real data collected at 17° slant was used as the initial training set, and $N|0, 5, 10...50, 55, 58$ images were randomly selected for replacement by SAMPLE synthetic data at 17° slant. Images from all classes were standardized by subtracting the mean and dividing by the standard deviation of their respective class and type before the experiment began. The results of this experiment are presented in Figure 3.9.

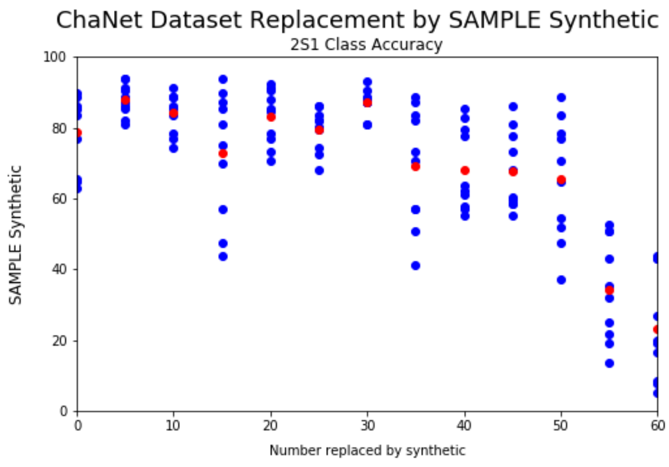

Figure 3.9: ChaNet accuracy results for training/testing cycle when images in 2S1 class of SAMPLE 17° data were randomly replaced with images from the SAMPLE synthetic dataset 2S1 17° data.

Second, the three winning generator weights were used to test if the identity loss metric pushed the network towards learning a transformation such that images in MSTAR remain in the MSTAR domain, and preferably their own class. To do this, the 299 images in the 2S1 class of MSTAR (domain B) were put through the domain A to domain B generator, $\mathscr{G}_{AB}\{\mathbf{X}_B\} \rightarrow \mathbf{X}_B$. These images were then used to replace images in the MSTAR dataset for use in training the same ChaNet neural network. The resulting accuracy on 15° images is presented in Figure 3.10.

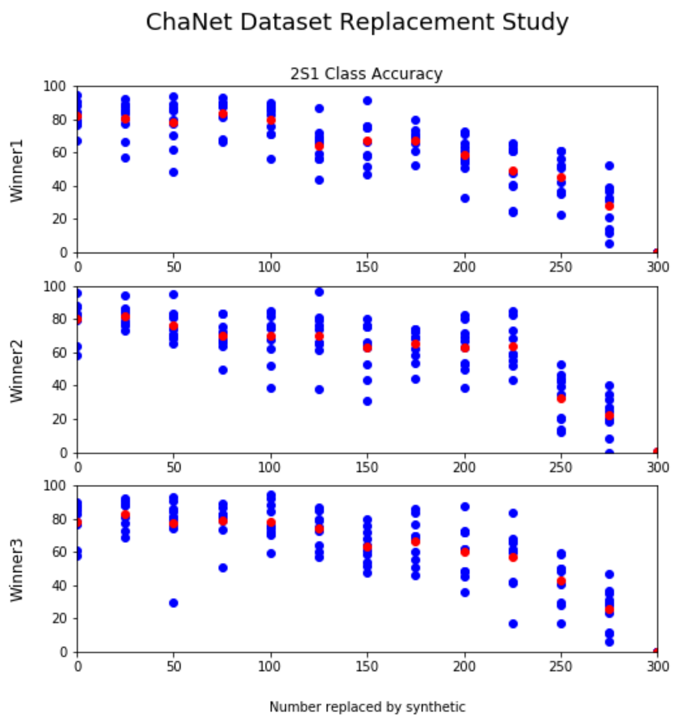

Figure 3.10: ChaNet accuracy results for training/testing cycle when images in 2S1 class replaced with MSTAR images modified by GAN.

Chapter 4

Discussion

4.1 Conclusion

In this thesis a procedure was presented to attempt conversion from simulated Electro-Optical Images to simulated Radar-Like images. While this procedure does create images which appear radar-like to a person, they do not exhibit enough of the features necessary for automated classification. In fact, testing the Identity loss metric by using actual radar images for input to the network and using both transforms returned the fact that information necessary to classify the target in the image was effectively destroyed, even while in Figure 2.4 (repeated in Figure 4.1 below) the input and returned images appear nearly identical to a human observer.

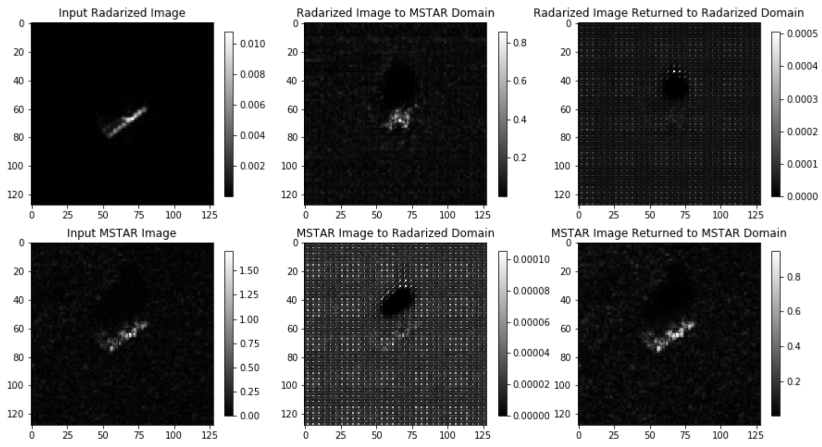

Figure 4.1: Inputs and outputs from CycleGAN architecture including those which return the input images to their original domain.

In broad terms, this thesis demonstrates that assuming an idealized annular impulse response for a collection of scatter points in scene is not sufficient for training an Automated Target Recognition network on MSTAR data. In *Anatomy of a SAR Impulse Response* there is a collection of impulse responses and demonstration that they change heavily depending on choice of processing algorithms and windowing. Many assumptions were made concerning the processing and windowing functions used to determine the IPR in this case, and any number of them could be sufficient in ensuring that simulated data does not match collected data.

A useful metric for this study is the fact that full wave simulated phase history used to create the SAMPLE dataset was able to achieve 71.3% accuracy (on the SAMPLE test data) using ChaNet when trained on only simulated data and tested on real data while Radarized DIRSIG output using the ideal windowed annular SAR impulse response followed by a Generative Adversarial Network approach created data that provided a 0% accuracy when the other classes contained measured data.

Building a network with an electro-optical to radar conversion may be effective for making an image look radar-like but it does not impart enough features on which a neural network uses to classify actual radar images. One possible fault lies in the use of a histogram operation to convert the off-nadir DIRSIG generated images into a nadir view radar-like input to the neural network. Even though the impulse response for a SAR system was calculated and applied to the input images prior to the GAN, this may have been insufficient to remove high frequency content not present in SAR images.

It should be noted that DIRSIG images were simulated from the estimated center point of the synthetic aperture for a SAR platform only. In deriving parameters for the impulse response it was found that an approximately 3° synthetic aperture was required by the SAR system to obtain cross range resolution for the MSTAR images. This means that information is present in the SAR images from 1.5° off of the DIRSIG simulation view in each direction in azimuth. Pieces of the scene which would be revealed at any location other than the center of the synthetic aperture are not accounted for in the synthetic images. Thus if the scene contains a narrow specular lobe which intersects the synthetic aperture but not through its center, the lobe is then absent in the simulated image. In order to extend this method for use on other SAR datasets such as AFRL's wide angle SAR dataset with 360° of azimuth, multiple images would need to be compiled together in the "slow-time" direction in order to provide information from these pulse locations in the synthetic aperture.

Another source of potential error lies in the fact that the normalization transform was not tailored to the dataset ahead of training. While a person can see that the cycle images for MSTAR data are minimally affected by the $\mathscr{G}_{BA}\{\mathscr{G}_{AB}\{\mathbf{X}_A\}\} \rightarrow \mathbf{X}_A$ cyclic transform, the Radarized data is decimated by the cyclic domain transitions due to its compressed nature/very small variance. It may be possible to expand the variability of Radarized data by ensuring the standard deviation of the Radarized sets matches that of the MSTAR dataset.

Chapter 5

Future Work

5.1 Future Work

Since the direct method of using a Generative Adversarial Network from Electro Optical to Radar was presented here and failed, it is recommended to attempt an indirect method by using the Radar Capability of DIRSIG 4 to simulate a phase history of the targets, form the radar image, and use it for inputs to the GAN. This is similar to the work performed in [11] except that it would require using the output image itself instead of classification based on binary threshold. This would allow an improved match to the cumulative distribution function of each target.

Presently, complex valued inputs and neural networks are not supported in any of the major neural network packages. While papers and books exist on the topic of complex neural networks, few provide code of any kind. Some repositories such as `https://github.com/wavefrontshaping/complexPyTorch` have built toolboxes based on *Deep Complex Networks* by Trabelsi et al. 2018 [33] by creating layers which handle the real and imaginary portions separately. Without their own complex backpropagation algorithm, the results do not match with expectations for complex numbers as the gradients are not handled appropriately for complex numbers. The opportunity to reproduce ChaNet using this package was taken and used on the complex data of MSTAR. Findings show it did not perform sufficiently better (averaging .5-1.5% worse) than the real valued algorithms already used in the rest of this thesis.

Some research papers exist which state that they use complex neural networks on SAR data in research settings [8] without presentation of their code. As complex data becomes more accessible for Convolutional Neural Networks I recommend this problem is revisited in a fully complex manner, though I do not think the EO to radar conversion is reasonable even then.

An additional experiment to demonstrate the importance of using full wave data would be to perform machine learning experiments on phase history data itself and show the comparative accuracy with focused SAR data. A hypothesis is that the unfocused data contains a comparable amount of information to that of the focused images. That being said, the author assumes it will be important to find a way to incorporate the pulse location (or at least vector direction to platform) with the phase history data directly to the neural network in order to get the best possible results from the neural network.

Bibliography

[1] A. W. Doerry, “Anatomy of a sar impulse response,” 2007.

[2] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” *arxiv*, 2016.

[3] “Understanding and implementing cyclegan in tensorflow,” https://hardikbansal.github.io/CycleGANBlog/, accessed: 2020-07-09.

[4] C. Oliver, S. Quegan, and K. (Firm), *Understanding synthetic aperture radar images*. Raleigh, N.C: SciTech Publ, 2004.

[5] L. A. Gorham and L. J. Moore, “Sar image formation toolbox for matlab.” SPIE, 2010.

[6] D. McDonald, “Ritsar,” Oct. 2016. [Online]. Available: https://github.com/dm6718/RITSAR

[7] “Publications involving the mstar public release data,” 2003. [Online]. Available: https://www.sdms.afrl.af.mil/content/mstar/MSTAR_Public_Data_References_040421.doc

[8] L. Yu, Y. Hu, X. Xie, Y. Lin, and W. Hong, “Complex-valued full convolutional neural network for sar target classification,” *IEEE Geoscience and Remote Sensing Letters*, pp. 1–5, 2019.

[9] B. Lewis, T. Scarnati, E. Sudkamp, J. Nehrbass, S. Rosencrantz, and E. Zelnio, “A sar dataset for atr development: the synthetic and measured paired labeled experiment (sample),” vol. 10987. SPIE, 2019, pp. 109 870H–109 870H–16.

[10] L. van der Maaten and G. E. Hinton, “Visualizing data using t-sne,” 2008.

[11] B. Lewis, J. Liu, and A. Wong, “Generative adversarial networks for sar image realism,” vol. 10647. SPIE, 2018, pp. 1 064 709–1 064 709–11.

[12] K. P. F.R.S., “Liii. on lines and planes of closest fit to systems of points in space,” *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901. [Online]. Available: https://doi.org/10.1080/14786440109462720

[13] C. Eckart and G. Young, “The approximation of one matrix by another of lower rank,” *Psychometrika*, vol. 1, no. 3, pp. 211–218, 1936.

[14] C. Olah, “Visualizing representations: Deep learning and human beings,” 2015. [Online]. Available: https://colah.github.io/posts/2015-01-Visualizing-Representations/

[15] “Opensky sudoku generator,” http://www.opensky.ca/~jdhildeb/software/sudokugen/, accessed: 2020-06-24.

[16] G. Huang, Z. Liu, and K. Q. Weinberger, “Densely connected convolutional networks,” *CoRR*, vol. abs/1608.06993, 2016. [Online]. Available: http://arxiv.org/abs/1608.06993

[17] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, “Generative adversarial networks: An overview,” *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 53–65, 2018.

[18] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networkss,” in *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

[19] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, 2017.

[20] M. Frid-Adar, I. Diamant, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan, “Gan-based synthetic medical image augmentation for increased cnn performance in liver lesion classification,” *Neurocomputing*, vol. 321, pp. 321–331, 2018.

[21] C. Han, L. Rundo, R. Araki, Y. Nagano, Y. Furukawa, G. Mauri, H. Nakayama, and H. Hayashi, “Combining noise-to-image and image-to-image gans: Brain mr image augmentation for tumor detection,” *IEEE Access*, vol. 7, pp. 156 966–156 977, 2019.

[22] C. Han, H. Hayashi, L. Rundo, R. Araki, W. Shimoda, S. Muramatsu, Y. Furukawa, G. Mauri, and H. Nakayama, “Gan-based synthetic brain mr image generation.” IEEE, 2018, pp. 734–738.

[23] D. Zhao, D. Zhu, J. Lu, Y. Luo, and G. Zhang, “Synthetic medical images using f&bgan for improved lung nodules classification by multi-scale vgg16,” *Symmetry*, vol. 10, no. 10, p. 519, 2018.

[24] Z. Xu, C. Qi, and G. Xu, “Semi-supervised attention-guided cyclegan for data augmentation on medical images.” IEEE, 2019, pp. 563–568.

[25] J. Nehrbass, S. Rosencrantz, E. Zelnio, and E. Sudkamp, “”afacet”: a geometry based format and visualizer to support sar and multisensor signature generation,” 04 2018, p. 1.

[26] J. Westerkamp, C. Paulson, B. Sudkamp, and C. Bergevin, "An automated process to ingest afacet models into dirsig for multi-sensor data generation," p. 1, 07 2019.

[27] "Digital imaging and remote sensing image generation," https://dirsig.cis.rit.edu/docs/new/intro.html, accessed: 2020-07-06.

[28] A. A. Goodenough and S. D. Brown, "Dirsig 5: core design and implementation," vol. 8390, 2012, pp. 83 900H–83 900H–9.

[29] T. E. Oliphant, *A guide to NumPy*. Trelgol Publishing USA, 2006, vol. 1.

[30] S. Marcel and Y. Rodriguez, "Torchvision the machine-vision package of torch," in *Proceedings of the 18th ACM International Conference on Multimedia*, ser. MM '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 1485–1488. [Online]. Available: https://doi.org/10.1145/1873951.1874254

[31] L. M. Mescheder, A. Geiger, and S. Nowozin, "Which training methods for gans do actually converge?" in *ICML*, 2018.

[32] M. Cha, A. Majumdar, H. T. Kung, and J. Barber, "Improving sar automatic target recognition using simulated images under deep residual refinements." IEEE, 2018, pp. 2606–2610.

[33] C. Trabelsi, O. Bilaniuk, Y. Zhang, D. Serdyuk, S. Subramanian, J. F. Santos, S. Mehri, N. Rostamzadeh, Y. Bengio, and C. J. Pal, "Deep complex networks," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=H1T2hmZAb