AFRL-RY-WP-TP-2019-0173

MACHINE LEARNING TECHNIQUES FOR SAR DATA AUGMENTATION (Preprint)

Benjamin Lewis, Michael Levy, John Nehrbass, Theresa Scarnati, Elizabeth Sudkamp, and Edmund Zelnio
Multi-Sensing Knowledge Branch
Multi-Domain Sensing Autonomy Division

AUGUST 2019
Final Report

AIR FORCE RESEARCH LABORATORY
SENSORS DIRECTORATE
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7320
AIR FORCE MATERIEL COMMAND
UNITED STATES AIR FORCE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
August 2019	Book Chapter Preprint	29 August 2019 – 29 August 2019

4. TITLE AND SUBTITLE
MACHINE LEARNING TECHNIQUES FOR SAR DATA AUGMENTATION (Preprint)

5a. CONTRACT NUMBER
In-house

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
N/A

6. AUTHOR(S)
Benjamin Lewis, Michael Levy, John Nehrbass, Theresa Scarnati, Elizabeth Sudkamp, and Edmund Zelnio

5d. PROJECT NUMBER
N/A

5e. TASK NUMBER
N/A

5f. WORK UNIT NUMBER
N/A

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory, Sensors Directorate (AFRL/RYAP)
Wright-Patterson Air Force Base, OH 45433-7320
Air Force Materiel Command, United States Air Force

8. PERFORMING ORGANIZATION REPORT NUMBER
AFRL-RY-WP-TP-2019-0173

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory
Sensors Directorate
Wright-Patterson Air Force Base, OH 45433-7320
Air Force Materiel Command
United States Air Force

10. SPONSORING/MONITORING AGENCY ACRONYM(S)
AFRL/RYAP

11. SPONSORING/MONITORING AGENCY REPORT NUMBER(S)
AFRL-RY-WP-TP-2019-0173

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
PAO case number 88ABW-2019-2698, Clearance Date 3 June 2019. To be published in the book "Deep Neural Network Design for Radar Applications". This is a work of the U.S. Government and is not subject to copyright protection in the United States. Report contains color.

14. ABSTRACT
Application of open-source machine learning algorithms to publicly-available SAR data. Nearly all of the work contained in this chapter has been published and cleared for public release previously.

15. SUBJECT TERMS
generative adversarial networks (GAN), machine learning, MSTAR, SAR, synthetic data generation

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT:	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON (Monitor)
a. REPORT	b. ABSTRACT	c. THIS PAGE			Benjamin Lewis
Unclassified	Unclassified	Unclassified	SAR	45	19b. TELEPHONE NUMBER *(Include Area Code)* N/A

Chapter 1

Machine Learning Techniques for SAR Data Augmentation

Benjamin Lewis[1], Theresa Scarnati[1], Michael Levy[1], John Nehrbass[2], Edmund Zelnio[1] and Elizabeth Sudkamp[1]

1.1 Introduction

Automatic target recognition (ATR) for synthetic aperture radar (SAR) is a difficult problem due to the sensitivity of recognition performance relative to operating conditions (OC's). These operating conditions can be divided into three categories: target conditions, sensor conditions, and environmental conditions. Examples of target conditions include target configuration, component articulations, and target pose. Sensor conditions, including sensor resolution, signal to noise ratio, and sample spacing all affect the image scale and composition. As a result, different radars can produce images that vary in significant ways. Environmental conditions such as background clutter, obscuration, and object adjacency can make discerning and classifying targets even more difficult. These OC's can be combined in any number of ways, which indicates that a large amount of imagery is required to represent the variability inherent in the SAR target recognition problem.

For some applications, such as traditional electro-optical (EO) camera imagery, data is plentiful and is representative of the variabilities due to the OC's or nuisance parameters. However, for target recognition applications that use SAR as a sensor, representative, measured data is not readily available and is expensive to collect. This lack of data is exacerbated when applying deep learning to the SAR target recognition problem, as deep learning learns about the variability of target, sensor, and environmental conditions by sampling data over all these conditions. Limited data does not adequately represent the OC's, which causes problems with classifier generalization.

Although deep learning algorithms have been successfully applied to images in the visible spectrum, limited attention has been given to applying deep learning to SAR. Like visible sensors, SAR is an imaging sensor, but there are some significant differences that make the SAR ATR problem unique and will likely require some additional innovation for the successful application of deep learning.

[1]Air Force Research Laboratory, Sensors Directorate, Dayton, OH
[2]Wright State University, Dayton, OH

Some of the differences between radar and visible wavelengths favor SAR. A major advantage of SAR is that it is an active sensor that controls its own illumination. In contrast, visible sensors are, in general, passive and must rely on a relatively uncontrolled illumination source. This illumination control allows SAR to produce images in which the scale is independent of range, whereas the scale of visible sensor images inherently depends on range.

Other differences favor EO sensors, which operate at a significantly shorter wavelength. At these shorter visible spectrum wavelengths, most surfaces are rough, whereas the majority of surfaces, including most man-made ones, are fairly smooth at radar wavelengths. Thus surfaces viewed at visible wavelengths tend to be observable from all viewing orientations, while the same surfaces exhibit specular reflection at radar wavelengths. In effect, this means that many man-made surfaces are only visible from a few viewing locations in the radar domain, generally only when the surface is perpendicular to the sensor line of sight.

The imaging geometry is also different for SAR than for EO. In the EO domain, images always project along the line of sight of the camera. When forming SAR images, the image is a product of the flight path and radar look direction, which creates additional signature variability. Fortunately for SAR, there are a few reflection mechanisms that persist over wide viewing angles and make up the predominant scattering in a SAR target signature. Despite this, the longer wavelength and specular scattering makes SAR signatures much less interpretable and stable than images in the visible spectrum.

The effect of operating conditions and the physics of the problem also affect the way that deep learning is applied to the SAR ATR problem. One of the common approaches to training deep learning algorithms is data augmentation, where the training imagery is augmented by translations, reflections, rotations, scaling, additive noise, etc. These techniques can help the limited SAR data problem if applied judiciously to a point, but many of the variabilities of SAR, particularly considering target and environment variability, are not amenable to image transforms to a limited amount of data. Hence, a promising form of data augmentation is to add synthetic data to the training set. The advantage of synthetic data is that a wide variety of operating conditions are straightforward to generate synthetically, while the same configurations are expensive and impractical to collect with measurement systems. Another key advantage of synthetic data is that the labels and accurate metadata about the target, environment, and sensing conditions are inherent in the data generation process. In contrast, the labeling and truthing process for collected data is very expensive and often very time-consuming. Human annotation of SAR imagery is significantly more difficult as the longer wavelength and specular nature of SAR makes the imagery highly non-literal and, therefore, much more difficult to correctly label.

Unfortunately, electromagnetic data is also very difficult to model, which makes the generation of high-fidelity SAR imagery complex and computationally challenging. Substantial progress has been made on improving both the quality and speed of SAR synthetic image generation. However, even with state of the art synthetic SAR imagery, there remains a gap between the measured SAR imagery and synthetic SAR

imagery when using these two image sources to train deep learning algorithms. This gap will be explored in Section 1.3.

This chapter investigates several strategies to close the synthetic/measured SAR data gap so that synthetic data can augment measured data to effectively allow deep learning algorithms to be used to perform SAR synthetic data augmentation. These strategies include:

- Improvements on the fidelity and speed of SAR synthetic signature prediction (Section 1.2.1.2)
- The development of a measured and synthetic dataset to be used to explore solutions to this problem, including a way to measure progress and compare results (Section 1.2)
- Optimization-based preprocessing to smooth speckle and to help normalize the relative amplitudes between measured and synthetic data (Section 1.4.2)
- Using joint measured and synthetic training strategies that share features at the lower layers of a deep learning architecture, yet contend with the different measurement and synthetic data manifolds at the later classification layers (Section 1.4.3)
- Using generative adversarial networks (GANs) to alter synthetic data to look closer to measurement data, in order to make more effective training data (Section 1.4.5)
- Using Siamese and Triplet networks to learn feature spaces that reduce the gap between measured and synthetic data features while still discriminating among the target classes (Section 1.4.6)

These strategies will be described and results will be presented. These approaches are not the only approaches that can be used to attack this important synthetic data augmentation challenge for SAR. Although progress has been made, many issues still remain as this key data augmentation problem remains an open problem.

1.2 Data Generation and the SAMPLE Dataset

A key enabler for any work in machine learning is a rich dataset. Unfortunately, extensive radar datasets are difficult to come by. One of the best-known SAR datasets was collected during the late 1990's by the Moving and Stationary Target Acquisition and Recognition (MSTAR) Program [1] which, despite the name, only contains imagery of stationary vehicles. This program was a collaboration among DARPA, the U.S. Air Force Research Laboratory, and Sandia National Laboratory, as well as a number of contractors.

The publicly available portion of the MSTAR dataset has enabled a large amount of research in SAR ATR. Prior to the widespread use of machine learning algorithms, this research utilized many traditional computer vision techniques, such as engineering feature extractors, leveraging pattern recognition, and designing expert model-based and rule-based knowledge systems [2, 3]. These systems perform well under

certain circumstances but, as is the case when applied to traditional camera images, require expert knowledge and great effort to maintain.

The advent of convolutional neural networks was a disruptor in this field in the same way as the computer vision field, and much research has been devoted to achieving good SAR ATR performance using these techniques. The technology is quite mature for classifying vehicles within the MSTAR dataset, with typical classification results when training and testing on MSTAR data at about 99% [4, 5, 6].

To go further, more data is needed. As discussed in the introduction, data collection is expensive and time-consuming. The most promising and cost-effective form of SAR data augmentation is simulated data, but this is not without problems. There is an inherent divide between measured and synthetic data, mainly due to the way in which data simulation over-represents some SAR phenomenology.

To more fully investigate these differences, the Synthetic and Measured Paired Labeled Experiment (SAMPLE) dataset was created [7]. This dataset uses a part of the MSTAR dataset and pairs synthetic data with each measured image. Using available data about the position of vehicles during the MSTAR data collects, computer models of each target were created to match this configuration. The targets included in the SAMPLE dataset are listed in Table 1.1. This section discusses methods of data generation, model truthing, and image formation techniques used to create the SAMPLE dataset.

Vehicle	2S1	BMP2	BTR70	M1	M2	M35	M548	M60	T72	ZSU23
Serial	B01	9563	C71	0AP00N	MV02GX	T839	C245HAB	3336	812	D08

Table 1.1 A list of the vehicles and serial numbers included in the SAMPLE dataset

1.2.1 Electromagnetic Data Generation

In an ideal world, all SAR images would be acquired from real-world measuring systems such as radars on board aircraft (air-to-ground), land-based radar, air-to-air radar, satellite systems, or from laboratory research measurement systems. Unfortunately, creating the required volume of data is generally not feasible. For example, it is not possible in many cases to adequately control radar systems to consistently collect data in the desired configuration, as the air turbulence and wind will bounce an aircraft, resulting in nonlinear measurement collections caused by speed variations.

In some cases, real-world radar measurement is not even possible. It is often desirable to have information about the electromagnetic properties of a product that is still in development, especially when those properties are a key characteristic of that product. Particularly in early stages of development, measurement-ready devices have not yet been manufactured, rendering physical assessment impossible.

In these cases, electromagnetic simulation software is an invaluable tool for the SAR ATR developer. This type of software leverages Maxwell's equations and a computer model of the target to calculate the response of the target to electromagnetic radiation. The consistency, speed, and low expense of simulation allow the

creation of large datasets that serve to augment the physical collection of radar data and SAR imagery.

Because simulated data is generally meant to be used alongside or to inform real-world data and design, it is important that the software be able to accurately model the physics of real radar systems. As is the case in many software techniques, however, there is a tradeoff between performance and fidelity.

1.2.1.1 Simulation Types

Electromagnetic simulation codes fall into two different classes. Full wave solvers, which provide very accurate electromagnetic results, are implemented by solving Maxwell's equations over the surface or volume of interest. Solutions generated by these solvers are very accurate, but the computational requirements for such fidelity are very high and scale exponentially. In contrast, approximate solvers, also known as asymptotic solvers, use methods such as ray tracing to more quickly provide information at the cost of accuracy. The computational cost of approximate methods scales linearly with the complexity of the problem, making it much more feasible for large problems.

Approximate methods become more accurate as the electrical size of the target grows, but these methods perform poorly in regions containing cavities and detailed components. Therefore, hybrid methods exist to take advantage of the strength of both approaches, balancing computational cost and accuracy. These mixed solvers implement full wave methodology exclusively over difficult regions and use asymptotic methods elsewhere. The two independently calculated solutions are then combined.

The complexity and intricacy of the target model under simulation also influences the type of code used. This is due to the ray sampling density required to adequately simulate the response of a target to electromagnetic radiation. When using computers to model the real world, the continuous electromagnetic waveforms and the computer-based model of the target are discretely sampled. Exactly how these sampling locations are selected is dependent on the simulation code and is beyond the scope of this chapter but, in general, the Nyquist sampling rate is chosen as a fraction of a wavelength $L = c/f$ (e.g., 10 samples per wavelength). The sampling rate, in conjunction with the electrical size of the target, determines the number of calculations required to simulate the response of a target to an electromagnetic wave. This computational complexity is directly proportional to the size of the target and the frequency of the radiation. For this reason, simulations of large targets (such as a school bus) simulated at high frequencies (e.g., 10 GHz) require sampling rates that are too costly to solve with full wave solvers, and instead rely on asymptotic or mixed-method software.

1.2.1.2 Simulation Workflow

The output of an electromagnetic simulation of a target from one viewpoint produces the equivalent of a single radar pulse from a monostatic system. Such radar pulses are a measure of the complex radar cross section (RCS) of an area of interest over

many different frequencies in both phase and magnitude. A collection of these pulses is referred to as phase history data (PHD).

Collecting multiple radar pulses from a variety of angles allows for a richer electromagnetic view of the target. The additional degrees of freedom enable the data to be assembled into higher-dimensional views of the target. A few simple equations, which are summarized in Data Box 1.1, determine the resolution of this data.

A single radar pulse with a given bandwidth B (Hz) is the Fourier transform of a one-dimensional range profile. Range profiles describe the intensity of radar return as a function of distance from the radar. The down-range resolution dr (meters) is inversely proportional to the bandwidth according to the equation $dr = c/2B$. If a number of discrete frequencies (N_f) are sampled over the bandwidth, the range profile provides salient data over an unambiguous range extent (meters) according to RE$= (N_f - 1) \times dr$.

To create two-dimensional radar images, radar pulses must be simulated over several different locations. The elevation angle is generally fixed and radar pulses are collected at several different azimuth angles over a collection aperture of θ radians. This extra degree of freedom enables the formation of 2D complex SAR images. Similarly, samples over many elevation angles introduce another degree of freedom with which to create a 3D image.

In the two-dimensional SAR case, the cross-range resolution dx (meters) is dependent on the azimuth extent θ and the frequencies present in the radar pulse. If L_{max} is the wavelength of the lowest frequency of B, then the cross-range resolution is given by $dx = L_{max}/2\theta$. The unambiguous extent that can be represented by the collection of radar pulses is dependent on the number of radar pulses in the azimuth N_s. This maximum cross-range extent is given by XRE$= (N_s - 1) \times dx$; any image that is larger than this will exhibit aliasing. This implies that to form a reasonable SAR image, many closely-spaced radar pulses must be simulated or collected.

Although a true SAR image is a function of a collection of independent radar pulses, electromagnetic simulations may use an approximation method to create a reasonable estimate of these pulses using only the rays traced from one location at the center of the SAR aperture. This single-ray approximation method then extrapolates a sufficient number of additional pulses (the number of which is dictated by the above equations) by bi-static scattering. This method works best when targeting well-known sensors because knowledge about the sensor can aid in the particular extrapolation method. While single-ray scattering inherently introduces inaccuracies, SAR images may be produced in minutes using this method, magnitudes faster than when computing each radar pulse from its actual geometric location.

A more accurate but costly method of forming SAR imagery is to compute a phase history data dome, in which radar pulses are finely simulated over 360 degrees in azimuth and a swath in elevation. In real-world collects, SAR phase history radar pulses are collected at defined points along the radar flight path. With synthetic data, radar pulses from these same flight path points are instead extracted from the data dome to form the phase history. This method is ideal when the target sensor is not well understood or when modeling different sensors, as the specific SAR image

Data Box 1.1

SAR Resolution Equations

$dr = c/(2B)$ $\quad$ $\text{RE} = (N_f - 1) \times dr$ $\quad$ $L = c/f$

$dx = L_{max}/(2\theta)$ $\quad$ $\text{XRE} = (N_s - 1) \times dx$

Variable Definitions

c - Speed of light (m)
f - frequency
L - wavelength
B - Bandwidth (Hz)
θ - Azimuth aperture (radians)
dr - Down-range resolution (m)
dx - Cross-range resolution (m)
XRE - Unambiguous cross-range extent (m)
RE - Unambiguous range extent (m)
L_{max} - Wavelength of lowest frequency in B (m)
N_s - Number of samples in θ
N_f - Number of frequency samples in B

formation parameters may be chosen during postprocessing. In contrast, it is difficult to reform the SAR images generated by the single-ray method according to the parameters of a different sensor.

The computational burden of forming a phase history data dome is not trivial, however. As an illustrative example, consider a small SAR training set consisting of 30 different targets. Suppose that the data dome for a given articulated target requires 25 samples per degree over 20 degrees of elevation and 360 degrees in azimuth, and that co- and cross-polarizations are required. A data dome for one target would require 18 million ($360 \times 25 \times 20 \times 25 \times 4$) radar pulse simulations. Furthermore, target articulations and changes in ground materials each require the data dome to be completely re-computed. If each of the 30 targets were placed in four articulations (which is a small number of configurations) and placed over three ground materials (e.g., tar, sand, and soil), the entire dataset would require nearly 6.5 billion ($30 \times 4 \times 3 \times 18$ million) radar pulse simulations.

The combinatoric growth of the number of pulses required to form a synthetic dataset means that such simulation quickly becomes computationally expensive. Computing a single RCS return for a given target can take several hours, especially since it is common to simulate 2^n frequencies over the simulated bandwidth (with n typically ranging from 2 to 11). In short, the development of large simulated datasets requires the use of massively parallel high performance computers (HPC).

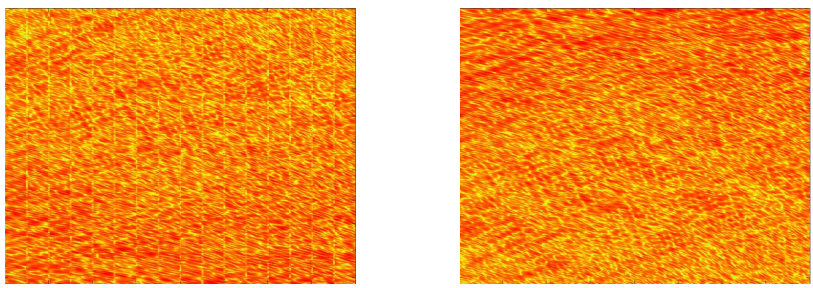
Even when using HPC systems, it is desirable to find ways to accelerate simulations and reduce computational costs. One simple method is to leverage general-purpose graphics processing units to accelerate the ray tracing portion of asymptotic simulations, which constitutes approximately 80% of the computational burden. More complex techniques can reduce the number of ray traces required by making a few assumptions about the geometry of the problem.

In most cases, the variation in ray paths between closely-spaced vantage points is extremely minimal. However, the relative distance along each ray path from source to target, and thus the phase of each electromagnetic waves, will change. Rather than re-computing the problem from the start, including the expensive ray tracing, it is much more efficient to instead correct the phase of each ray according to the geometry of the target. This extrapolation method is generally used for samples between computed points (e.g., between samples at one degree increments). For problems with 25 samples per degree (the resolution used to create the SAMPLE dataset), the speedup can theoretically be $625\times$ while providing nearly the same results as computing a ray trace for every sample.

With that said, the extrapolation between points is, in general, discontinuous. For example, rays traced from a 10° azimuth angle will follow different paths than those traced from 11° azimuth. While the paths followed by rays from the 10° pulse may adequately approximate the rays seen from 10.04°, it is less of a valid approximation for the problem as seen from 10.5°. In that case, the ray geometry as seen from 11° azimuth is just as valid an approximation as that from 10°.

To avoid discontinuities as extrapolation locations change, each extrapolated point is computed as a combination of adjacent computed points with smoothing function applied. This smoothing function makes the final simulated data more cohesive. Inter-pulse extrapolation also reduces electromagnetic scintillation from the angular facets of the models, which appears as undesirable cross-range smearing in SAR images.

A useful sanity check when implementing inter-pulse extrapolation is to create and view a matrix of several adjacent pulses. Discontinuities will clearly be present at the boundaries where the smoothing function is not robust, such as in Figure 1.1a . Likewise, good implementations will produce "fingerprint-like" patterns where there is close agreement between ray shoots and extrapolated points, as in Figure 1.1b.



(a) (b)

Figure 1.1: An example of the "fingerprint" produced when viewing data generated using the inter-degree extrapolation technique discussed here. The erroneous data in (a) exhibits obvious discontinuities along vertical lines, while the data in (b) is continuous across the entire extent.

For the purposes of this chapter, the end goal of creating simulated data is to convince a detection system that there is no difference between simulated data and

measured data. Once this objective has been obtained, the simulated data gains great utility for the end user. Several factors make this objective challenging. CAD models of targets, which are at best approximate representations of real vehicles, are pristine. In contrast, real-world targets are deformed, dented, dirty, and subject to variable conditions. Thus a collection of several measurements of different buses of the exact same make and model will be different.

The material properties used in simulations are also often inadequate. While simulation codes are very capable at accounting for electromagnetic material properties, which can be assigned to individual parts of the computerized target model, good information about these parameters is difficult to obtain. In a similar vein, the ground plane on which targets are placed is generally represented as a trivial flat surface with statistical scattering properties. While this is obviously a poor recreation of the realities of the real world, which includes rocks, trees, grass, potholes, etc., it is simply not feasible to model all these features.

To this end, the work of this chapter is fundamentally about exploring machine learning techniques to overcome the limitations discussed here. For best results, the simulation expert must work closely with the ATR developer throughout the process to ensure that the generated data is adequate for their purposes. Failure to do so ultimately affects the ability of such algorithms to perform adequately in deployment.

1.2.2 Model Truthing

Simulated data generation is heavily influenced by the quality of the computer-aided design (CAD) models of each target. Electromagnetic simulation works by computing the interaction of simulated radiation with the surface of each model. CAD models are generally represented as collections of triangular facets. The size of each facet impacts how much of a contribution it will make to the overall electromagnetic return of the target. For this reason, CAD models that are more finely faceted are a better representation of actual targets than coarsely defined models, and the radar return from these targets will more closely match that of targets in the real world.

The CAD model's articulation (i.e., the position of movable parts) is also a determining factor that affects the radar return from a target. While no given articulation is inherently better than any other, certain configurations may be of more interest. The radar returns from a vehicle with its door open and shut are much more distinct than returns from a vehicle with its windshield wipers in different positions. With that said, exposing the interior of a vehicle by opening a door or a hood requires much more consideration when modeling the vehicle, as the exposure of internal components (such as the engine) makes the radar return much more complex.

The SAMPLE dataset is based off of measured SAR images from the MSTAR collect. In creating the simulated SAR images, it was decided that minimizing the configuration differences between the measured data and the synthetic data was of high value. By so doing, the bulk of the difference between images from the two domains is due to the inherent mismatch between simulation and measurement, rather than controllable configuration differences.

CAD models for targets were sourced from MSTAR Program data products. While the MSTAR dataset contains more than the ten targets listed in Table 1.1,

the provided CAD models for many of these targets were unfit for electromagnetic simulation. Obvious defects such as major missing parts, excessively coarse representation, or completely missing models disqualified the targets from inclusion in the SAMPLE dataset.

Although the MSTAR Program made efforts to configure CAD models to the extent described here, these efforts were incomplete even on the viable models. Large and small differences necessitated work to bring the CAD models into the ideal configuration. This section describes the effort to do so by adjusting the CAD models to match the positions of the chosen MSTAR targets, including collecting and organizing truth data, standardizing the CAD models, and iteratively validating the CAD models with scrupulous comparison to truth data.

1.2.2.1 Truth Data

The configuration of each target's CAD model is based on data from the MSTAR data collect. Information such as photographs of vehicles from the data collects, a data handbook written by the Air Force Research Laboratory, data file headers, contemporary reports, and information about baseline vehicle configurations all were collected to serve as ground truth information for each vehicle.

The MSTAR data handbook includes information about the MSTAR Extended Operating Conditions (EOCs). Of particular use were sections detailing the intentional variations in articulations and target obscurations as well as unintentional variations (such as damaged or missing parts). The handbook also contain lists of the articulation angles of major parts (such as the gun barrel or turret) for each target.

One of the most helpful sources of information were the original MSTAR CAD model reports, created by Sverdrup Technology Inc. These validation reports were created jointly with the CAD models in the late 1990s. Although these reports were not without eccentricities, they served as a helpful supplement to the other information.

Perhaps the most useful information came from the contemporary MSTAR photographs (such as the example shown in Figure 1.2b). In general, the visual information presented in the photos allowed for rapid configuration of the CAD models. However, some images were taken under suboptimal conditions, such as poor lighting and abundant shadows, which at times made it difficult to see the details in the photos and limited their utility.

1.2.2.2 CAD Model Standardization

Before model truthing, a CAD model cleaning process was undertaken. Many vehicles exhibited a mismatch between equipment that was present on the vehicle during the collect and equipment represented on the CAD model. Accordingly, the models were adjusted to have the appropriate items. Many models also contained residual solids, or CAD objects that served as temporary stand-ins during the modeling process (such as cylinders in place of wheels) that were necessary to remove.

The CAD models were also converted to a standard file format. Initially, the computer models were stored in a format used by an outdated CAD model program. This format was difficult to work with and the software lacked documentation. To

(a) (b)

Figure 1.2: An AFacet CAD model of a M60 tank (a) that has gone through the iterative validation process described here. The image in (b) is one of the collected images with which the CAD model was compared. Note the correct position of large elements, such as the turret, and small elements, such as various hatches.

address this issue, the AFacet format [8] and associated software tools were developed in-house. In contrast to the binary format of Modelman files, the AFacet format consists of text files that describe the target and break it into separate part assemblies. Using this format, individual part assemblies can be added, removed, and articulated separately. AFacet also supports conversion to industry-standard CAD file formats, such as .stl and .obj.

To increase the utility of the models, electromagnetic material properties (such as those of glass, metal, or various man-made materials) were assigned to each surface. Many of the specific values for these properties were derived from in-house material properties libraries. Because electromagnetic waves interact with each material differently, the addition of material properties greatly increases the fidelity of simulation results.

1.2.2.3 Iterative Model Validation

The final step in achieving good computer models was to compare each model to the available MSTAR truth information over several iterations. This was done with a human in the loop, as automated processes were unable to effectively compare the 2D images to the 3D model at the desired fidelity. Each truthing step involved attention to ever-finer details due to the wavelengths used in the electromagnetic simulation. Because simulations for the SAMPLE dataset were carried out with a radar center frequency of 9.6 GHz, the median wavelength was on the order of one inch (3 cm). In general, electromagnetic waves interact strongly with features that are larger than one wavelength. Thus, it was important to pay attention to the position of objects down to this size.

This stage of model truthing was the most time-consuming, as it required human judgment and repeated assessments of each model. At times, the documented truth was imprecise or self-contradictory. In these cases, a judgment call was made to determine which source of data was most reliable. Over the course of several passes, the CAD models eventually reached a point at which no major disagreements between the truth data and the modeled vehicle could be detected. Due to this validation, the fidelity of the models is very high relative to the MSTAR targets. An

example of a validated model of an M60 tank can be seen in Figure 1.2. In theory, this model fidelity directly affects the quality of the electromagnetic simulation and the resulting SAR images.

At the end of this process, the CAD models were well-prepared for simulation. Using simulation methods described in Section 1.2.1, a dense data dome was created for each target, consisting of 360° in azimuth and elevations from 13.5° to 18.5°, a range sufficient to cover the azimuths and elevations of the MSTAR targets of interest. Radar pulses were computed every 0.04° in both azimuth and elevation and included the 600 MHz bandwidth, centered at 9.6 GHz, that was used by the MSTAR radars.

1.2.3 Image Formation

Matching the CAD models to the configuration of each target as it appeared during the MSTAR collect was a key enabler of the SAMPLE dataset. By so doing, the simulated data domes contain information about each target in the microwave domain in which radar operates. With this data, it is possible to create any number of SAR images.

As stated earlier, a simulated SAR image is formed by computing the points along a flight path at which a real radar would collect radar pulses to create the desired image. These radar pulses are then extracted from the data dome and combined into the phase history. The image is then formed from the assembled phase history data.

The SAMPLE dataset includes a matching synthetic SAR image for every measured SAR image. Each of the measured SAR images in the MSTAR collect is the radar view of a target at a given elevation and center azimuth angle. The files that store these images include the complex-valued SAR image and detailed metadata about the conditions under which the data was collected. Included in this metadata are the elevation and center azimuth angle as well as the information required to compute the resolution and number of samples, as described in Subsection 1.2.1. This data was sufficient to reconstruct the necessary radar flight path corresponding to a given SAR image, which then was used to create the radar data phase history.

A few preprocessing steps were used to make a more appropriate synthetic SAR image. In particular, a Taylor window function with $\bar{n} = 4$ and -35 dB was applied to the phase history in order to minimize the contribution of sidelobes to the formed SAR image. Because using such a window widens the mainlobe, the equivalent SAR aperture (and corresponding flight path) was widened by a unitless factor of 1.184 to compensate.

Inevitably, most of the points required to create the phase history did not lie exactly on the evenly-spaced simulated points. Data at these points was well-approximated using bilinear interpolation from the nearest four data dome pulses to the desired viewpoint.

Monostatic SAR images were formed from the constructed phase history using the far-field polar format algorithm, a widely-used fast Fourier transform-based image formation method which was used during the MSTAR Program to form the SAR images. Images were formed with an image size of 128×128 pixels with a pixel

Range resolution	0.30 m	Bandwidth	591 MHz
Range pixel spacing	0.20 m	Center frequency	9.6 GHz
Range extent	25.8 m	Image size	128×128
Cross-range resolution	0.30 m	Polarization	HH
Cross-range pixel spacing	0.20 m	Elevations	14°-18°
Cross-range extent	25.8 m	Taylor weighting	−35 dB

Table 1.2 General parameters for the MSTAR image files, which are available in the metadata of the publicly-released MSTAR dataset.

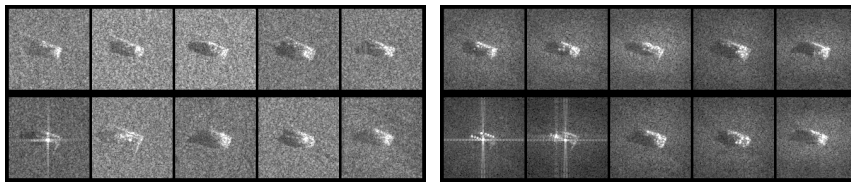
resolution of approximately 0.2 m in each direction. While most measured images were already 128×128 pixels in dimension, the rare larger exceptions were cropped to this smaller dimension. Other image formation parameters are listed in Table 1.2.

Each measured image was aligned with its corresponding synthetic image to ensure the best match of all parameters between the two images. As the position of the vehicle within the simulated image is well-defined, the measured images were aligned with the synthetic images rather than vice versa. For each image, a mask around the target was formed using a non-parametric variance-based joint sparsity method [9]. Outlier pixels were removed with a RANSAC-based [10] algorithm. All pixels within each mask were then quantized into seven quantization levels, of which the lowest two levels were set to zero. This produced a pair of quantized images with reasonably similar shapes and magnitudes. The pixel offset between the pair of images was computed using two-dimensional cross-correlation. The measured image was circularly shifted by the pixel offset amount to align with the synthetic image.

For the purposes of the machine learning techniques described in this chapter, the complex images were converted into magnitude-only images. After converting each image to magnitude by using the absolute value function, the image was normalized to values between zero and one. An image normalization function was then applied by taking the fourth root of each pixel (e.g., $|x|^{1/4}$), which is a common SAR image normalization technique that helps to make the details of the target stand out more. Each normalized image was saved as a loseless image file for later use. One image of each measured MSTAR vehicle and the corresponding synthetic SAR image are shown in Figure 1.3.

1.3 Deep Learning Evaluation and Baseline

Due to the combinatorially large number of possible views of a given target, automatic target recognition in SAR imagery is a difficult task. It is simple enough to build a classifier for a subset of the space, such as the MSTAR dataset. Several papers have reported 99% classification accuracy on MSTAR classification tasks, where the training dataset consists of images collected at 15° elevation and test imagery is from 17° elevation [4, 5, 6]. However, the problem of domain adaptation quickly becomes



(a) (b)

Figure 1.3: A set of measured MSTAR images (a) and corresponding synthetic SAR images (b) as seen from 17° elevation and 160° azimuth. Each small image within a larger figure represents one of the ten targets from the SAMPLE dataset, with targets presented in the same order in subfigures (a) and (b).

relevant when the training and testing data are not well-matched. The difference between synthetic and measured data domains is a prime example of this mismatch.

To illustrate this issue, the t-Distributed Stochastic Neighbor Embedding (t-SNE) [11] was calculated on the measured and synthetic portions of the SAMPLE dataset, which was introduced in Section 1.2. The t-SNE algorithm is designed to help visualize the clustering of data in high-dimensional datasets in a comprehensible two-dimensional plane. Points that are close together in the full-dimensional space tend to cluster together in the t-SNE projection, which makes t-SNE a convenient way to visualize the approximate separability of a dataset. The class label of individual points is not used when computing the t-SNE, but it often beneficial to add the class labels after t-SNE is computed to aid in graph comprehension.

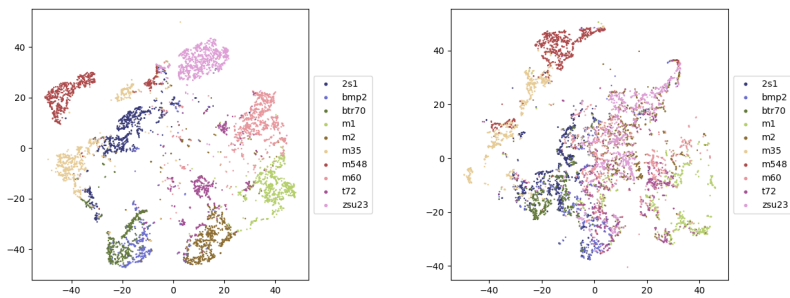
To visualize the measured and synthetic portions of the SAMPLE dataset (which, as the reader should recall, was designed from the beginning to be as well-matched as possible), a DenseNet [12] classifier was trained on the measured SAMPLE images. Using the trained network, multi-dimensional features were extracted from the network at the layer before the fully-connected classification layer. A joint dataset of features was computed for both the measured and synthetic portions of the SAMPLE dataset. All features were then used as inputs to t-SNE, which computed the two-dimensional embedding for each feature point. The embeddings were then separated according to the source of the image (measured or synthetic) and class labels were assigned. The measured t-SNE embeddings were plotted in the graph shown in Figure 1.4a, while the synthetic embeddings were plotted in Figure 1.4b.

As can be seen, the points representing each class cluster fairly well for the real images represented in Figure 1.4a. However, this separability decays for the synthetic points in Figure 1.4b. Because the two embeddings for the two sets of points are the output of the same network, this difference between the graphs suggests that even with great effort to match the collected imagery, the gap between the two domains is nontrivial.

To further illustrate the point, a DenseNet network implemented in PyTorch was used to classify measured SAR images. The DenseNet was modified to accomodate monochromatic images. The first MaxPool layer was also replaced with an adaptive MaxPool layer to allow for smaller input sizes. In the rest of the chapter, these

same modifications were made to accommodate the SAR images from the SAMPLE dataset.

Two experiments were performed. In the first, the classifier was trained on measured data, while in the second, the classifier was trained on synthetic data. Measured images were then classified with these trained networks. In each experiment, the DenseNet was trained on randomly sampled batches of 64 images. The Adam optimizer [13] was used with learning rate α=0.001 and beta parameters β_1=0.9 and β_2=0.999. Training was carried out for 50 epochs, several epochs after the validation accuracy has reached near-perfect levels, in accordance with theoretical results in the machine learning literature [14]. Training data consisted of SAR images at 17° elevation, while test data was the measured data collected at 15° elevation.



(a) (b)

Figure 1.4: A comparison of the synthetic and measured images from the SAMPLE database, as represented by the t-Distributed Stochastic Neighbor Embedding (t-SNE). In (a), the t-SNE embeddings for the measured images are plotted, while (b) shows the embeddings for the synthetic data.

In the first experiment, a near-perfect test classification accuracy of 97.5% was achieved, as shown in Figure 1.5a. This accuracy is well in line with results reported in the literature [4, 5, 6]. The success of this network is due to the extreme similarities of the train and test dataset, in that both the train and the test data comes from the same vehicle and the same collect. Because of this, the only major difference between the two datasets is the elevation at which it was collected. Due to the nature of the MSTAR dataset, operating conditions such as weather, target articulation, radar characteristics, ground type, and many more were essentially the same for the train and test sets of images.

In the second experiment, where the DenseNet was trained on synthetic data and tested on measured data, the strong performance of the first experiment was significantly degraded. The confusion matrix in Figure 1.5b shows that the accuracy of the network dropped to 51.5%. This illustrates the same point as the t-SNE embeddings in Figure 1.4 - namely, that the two domains are distinct and not easily bridged.

While more sophisticated and SAR-specific networks may provide better classification performance on this problem, the point is that even well-matched synthetic and measured data can have significant differences that hurt the generalization of

(a) (b)

Figure 1.5: A comparison of using synthetic and measured data to train a classifier. In (a), a DenseNet was trained using measured data, while in (b) the same network was trained with synthetic data. The confusion matrix shows the performance when classifying measured data with each network.

ATR algorithms. Despite great effort devoted to matching CAD models to collected truth, there will always be a mismatch between the two domains due to judgment calls, approximations made by computational electromagnetic software, and inaccurate truthing information. These imperfections motivate the use of machine learning to try and bridge the measured/synthetic gap.

In the rest of the chapter, several techniques will be presented, each of which aims to make measured and simulated data more interchangeable. As experimental results are presented, DenseNet classification results will be presented and compared to the cross-domain performance presented in Figure 1.5b where appropriate. DenseNet classification can appropriately be applied to techniques in which the synthetic data is altered to look more like measured data in some way. For example, layer retraining (Subsection 1.4.3) autoencoder-related techniques (Subsection 1.4.4) and generative adversarial networks (Subsection 1.4.5) all well-suited to direct classification methods (although the results presented with layer retraining do not use the DenseNet architecture). In contrast, Siamese and Triplet networks (Subsection 1.4.6) do not directly adjust the data, but rather work to learn appropriate data representations for both domains of data. As such, DenseNet results will not be directly presented for these latter methods. Instead, ways to leverage Siamese and Triplet networks as classifiers will be discussed.

1.4 Addressing the Synthetic/Measurement Gap with Deep Neural Networks

For the reasons described in the sections above, the problem of augmenting a SAR dataset with synthetic data is non-trivial, and no claim is made that the work in this

chapter fully solves the problem. In essence, bridging the difference between the two domains is a type of transfer learning problem.

Over the past few years, the authors have investigated several methods to address the measured/synthetic SAR data gap. Before discussing machine learning algorithms, a few data pre-processing routines are examined. These include a discussion of data augmentation measures that are productive and sensible for SAR imagery (Subsection 1.4.1) as well as a brief foray into SAR image despeckling techniques (Subsection 1.4.2). Removing the speckle and background from SAR images is an attractive pre-processing step that removes much of the non-target information from SAR images; in theory, this information removal allows the network to learn target signatures without undue influence from non-target portions of the image. On the matter of data augmentation, some typical data augmentation schemes do not make sense in the context of the physics of SAR images and will be advised against.

The remainder of the section discusses machine learning approaches to bridging the synthetic/measured gap. While space does not permit more than a cursory overview of each method, the authors have published conference papers on each topic, which are referenced in the appropriate section. These approaches include layer-wise training of networks (subsection 1.4.3), autoencoders (subsection 1.4.4), generative adversarial networks (subsection 1.4.5), and Siamese and Triplet networks (subsection 1.4.6). Each subsection discusses the architectural concepts of the network, including any major modifications made to support radar data, and presents results using the network to solve the synthetic/measurement gap.

While several approaches have been taken by the authors, much work remains to effectively use synthetic data when training machine learning classification algorithms. Future developments will be published by the authors as they become viable.

1.4.1 SAR Data Augmentation

In the machine learning community, a number of data augmentation techniques are generally prescribed as a way to help expand the variety of the dataset in a computationally inexpensive manner. Some of these methods are ineffective with SAR data, because the image formation process ensures that the structure of the image data is somewhat constrained.

Flipping, rotating, and scaling transformations, in particular, are inappropriate for use with SAR data. SAR images may be formed such that the illumination from the radar is seen from a constant direction in the image. As all images can be formed this way, flipping or rotating images merely creates additional views of the target that may be avoided in the image formation algorithm. Likewise, the measured spacing between pixels is a defined parameter when forming images. Thus, scaling images up or down does not add much value, as a scaled version of a given vehicle is unlikely to be seen in operation.

Other traditional data preprocessing techniques are of more use with SAR data. The position of the target within the image and the size of the image are not essential characteristics, so cropping images around the center or translating the target within the image are valid ways to augment the data. Image cropping tends to work well when classifying images, because much of the background speckle noise is no longer

available to provide non-target information to the network. Target translation can also aid networks in learning position invariance. While there are many ways to fill in data that is removed in translation, a viable way to do so with SAR imagery is to circularly shift the image such that the edge that is removed from one side is added to the other side. This is acceptable because the background contains no significant information about the target, is often noisy enough to mask the discontinuity between the two edges, and should not produce a measurable effect on the classification of the image.

Data augmentations that modify the pixel values in images are also appropriate for use with SAR imagery. Adding random noise to images can help with classification robustness as well as aid convergence in generative techniques. Transforming the range of pixel values to lie between -1 and 1 is also good practice, as it ensures that all pixels are in the same order of magnitude and prevents extreme gradients within the network.

1.4.2 Mathematical Despeckling

When viewing a SAR image, it is apparent that the brightness distribution of the image is not smooth, but rather composed of complicated granular patterns of bright and dark spots. This salt- and pepper-like noise, which is called speckle, is the result of the radar illuminating a surface that contains features that are rough when compared to the length of the illuminating wavelength. For example, it is common in SAR for the resolution cell to be at least an order of magnitude larger than the transmitted wavelength. The coherent summation of returns within the imaging cell from objects much smaller than the cell create a return pattern of varying intensity. Speckle noise varies based on the environment being imaged. From one image to the next, even if the same target is being imaged, speckle noise is independent.

Despite the independence of the speckle noise, machine learning systems have a huge capacity to fit a model to data. The nature of convolutional networks also means that these networks don't inherently possess the ability to focus on one area of an image in particular. As such, machine learning networks can easily learn to categorize images based on the background rather than the actual target, which is an obvious impediment to generalization.

One way to mitigate this problem is to eliminate the background clutter and speckle. This is an important task, as the clutter found in measured data is independent of the the simulated clutter found in synthetic data.

Understanding speckle requires a detailed examination of the properties of electromagnetic waves after they have been reflected or scattered from rough objects [15]. Because of the difficulty in understanding such details of the small scale structures of the complex wavefronts leaving the objects, it is common to instead generate a statistical speckle model. Such models predict the statistical properties of intensity over an ensemble of different rough surfaces with the same macroscopic properties, but different in microscopic detail. There are many examples of speckle models in the literature[9, 16, 17, 18, 19, 20, 21, 22].

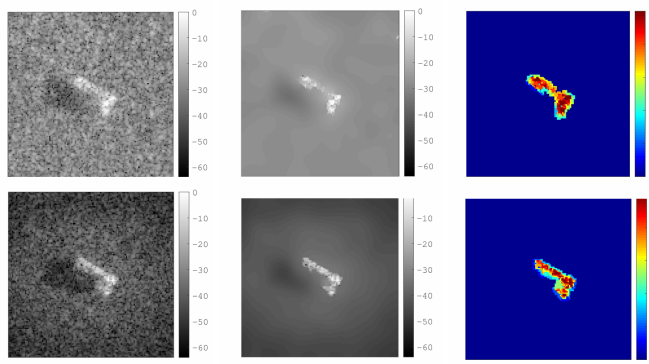
A statistical model of speckle can be derived from first principles. Speckle appears in images as a multiplicative noise term that affects the radar cross section of

each pixel [23]. The intensity of the multiplicative speckle term is characterized on a per-pixel basis by a negative exponential distribution. If the radar cross sections are known, then the speckle can be fully characterized using first order statistics [23]. These statistical characteristics can then be exploited by speckle reduction techniques.

Speckle reduction methods can broadly be categorized as image formation techniques or postprocessing methods. Reducing speckle during image formation is typically more flexible, as the techniques are applied to unprocessed data. Even so, postprocessing is also an important signal processing technique that can help filter irrelevant information from images.

When raw data is available, image formation despeckling techniques, including regularized inversion [22], variational methods [19, 21], and sparsity-based methods [9], are best able to preserve the information in the SAR image while removing speckle. These algorithms leverage the data to inform how the speckle noise should be reduced, all prior to introducing assumptions that affect the final image. Statistical speckle models and other prior knowledge about the scene may also be included when forming the image.

After the SAR image has been formed, postprocessing methods, such as filtering [17], multi-look averaging [16] and quantization [24, 25, 26], affect the individual pixels of the image in advantageous ways. These methods typically are much faster computationally, but tend to reduce the amount of relevant information within an image. Often, the target in a postprocessed image loses its structure and edges become blurred, which diminishes the information.



(a) (b) (c)

Figure 1.6: (a) Speckled images. (b) VBSJ IFT. (c) VBJS IFT with the quantization PPM. Here $n = 10$. (top) Measured SAR data. (bottom) Matching synthetic SAR data.

One useful image formation technique is high order regularization, which was posed in [27]. In [28], speckle reduction via high order regularization was explicitly studied and it was shown qualitatively and quantitatively that this technique successfully reduces speckle in SAR imagery while maintaining target fidelity.

Given SAR phase history data $\hat{f} \in \mathbb{C}^M$ and model $\mathcal{F} \in \mathbb{C}^{N \times M}$ that maps the data to the image, this high order regularization technique is defined as

$$f^* = \underset{f \in \mathbb{C}^N}{\operatorname{argmin}} \left\{ ||\mathcal{L}^m f||_1 + \frac{\lambda}{2} ||\mathcal{F} f - \hat{f}||_2^2 \right\}. \tag{1.1}$$

Here, $||\mathcal{L}^m f||_1$ is the regularization term that encodes prior knowledge about the solution into the system. The mth order sparsity operator $\mathcal{L}^m \in \mathbb{R}^{N \times N}$ projects the solution into a sparse subspace. Minimizing (1.1) with respect to the ℓ_1 norm enforces a solution that is mostly zeros in the sparse subspace defined by $\mathcal{L}^m$. In SAR, it is beneficial to allow $\mathcal{L}^m$ to map f into the space of sparse edges [29]. The data fidelity term, $||\mathcal{F} f - \hat{f}||_2^2$, enforces the solution to obey the physics-based data model and $\lambda > 0$ is the regularization parameter that measures the trade off between the prior knowledge and the physics-based model. Typically λ must be hand-tuned, a drawback of most regularization-based IFTs.

Variance Based Joint Sparsity (VBJS)[9, 30] image formation was adapted for speckle reduction of SAR imagery. This method eliminates the need to hand-tune regularization parameters by exploiting the overlapping support of multiple measurement vectors (images) in the domain where the signals are assumed to be sparse. Figure 1.6b displays the result of applying the VBJS technique to measured and synthetic (top and bottom images of Figure 1.6a, respectively) SAR data.

An example of a postprocessing technique is quantization. This has been explored in many template-based classification algorithms [24, 25, 26]. There are two main steps in the quantization procedure. First, a mask is placed over the area of interest (AoI). In SAR images, the AoI is typically considered the target region, but at times the shadow region is also included in the AoI. While it is a trivial exercise in bookkeeping to generate such a mask for synthetic images, AoI masks for measured data must be hand-drawn or discovered algorithmically. Algorithmic approaches, in particular, are crucial when handling large datasets. After the AoI mask is given, the pixels within the mask area are organized into n bins. It has been shown that for SAR images $n = 4$ [26] and $n = 10$ [24, 25] work well for template-based classification algorithms. However, similar research has not been done for machine learning classification algorithms.

Quantization can be applied after speckle reduction during image formation. Figure 1.6c displays the result of applying quantization with $n = 10$ bins after despeckling the SAR images via the VBJS. Here, the AoI masks are an intermediate output of the VBJS algorithm.

To illustrate the merit of despeckling SAR images for classification, a DenseNet [12] network was trained on four types of image types (Figure 1.7). The experiment was run with images that were i) unaltered, ii) despeckled via VBJS, iii) despeckled via VBJS and quantized with $n = 4$, and iv) despeckled via VBJS and quantized with $n = 10$. The experiment was designed to show the performance of the classification algorithm as a function of the amount of measured data available during training. Various combinations of measured and synthetic data were used to train the network, which was then used to classify measured data. Each combination of data is defined by the fraction of the measured data that was included in the training set.

For each experiment, the classification performance decreased as the percentage of measured training data decreased, and, simultaneously, the percentage of synthetic training data increased. For each image type, performance dramatically dropped when half of the training data were synthetic. However, Figure 1.7 shows that despeckling with the VBJS algorithm and the quantizing the result into $n = 10$ bins yields the highest classification performance when more than 35% of the training data are synthetic.

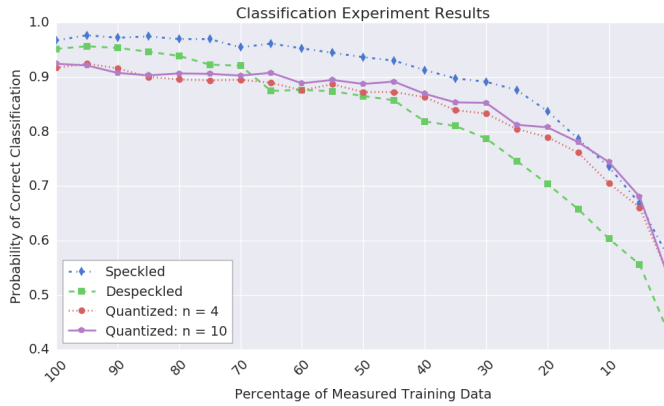


Figure 1.7: Classification performance as the percentage of measured data within the training set decreases and the percentage of synthetic data used for training increases for four different image types.

Note that it is unfair to compare the despeckled results to the results on unaltered images because, with the original data, the classification algorithm has the opportunity to learn not only the target signatures, but also the environment parameters that dictate the speckle noise. As all of the measured data was collected under similar conditions, there is a significant amount of shared information among these images. That shared information is attenuated by the despeckling process. Yet in the context of classification, this focused removal of information has the potential to broaden network generalization by allowing the network to solely focus on target signatures. Because the two types of images serve different purposes, the classification performance on the unaltered and despeckled images is not directly comparable. Even still, these results show that SAR image despeckling techniques are a promising preprocessing technique for deep neural networks[31].

1.4.3 Layer Freezing and Layerwise Training

Based on the classification performance of networks that are trained on synthetic data and tested on measured data, it is clear that there is a fundamental disagreement between the two domains and how each target is represented. The scattering in SAR is predominantly local and predominantly specular, which means that flat surfaces are generally not seen in radar returns[32]. However, at the rare viewing angles

where specular surfaces are visible, simulations tend to overpredict the amplitude of the return. These differences in the geometry of the model vs. the real target and the fact that the prediction codes are approximate create a gap between SAR model predictions and target measurements. Unless accommodations are taken as part of the data preparation and CNN design process, these difference between synthetic and measured data can degrade the performance of the classifier.

While CNNs have shown success on classification problems in multiple domains, limited data is still an issue. Transfer learning techniques [33], in which information from a network trained with large amounts of data is adapted to a problem with limited data, is one way of dealing with this limitation. One popular technique is to use layers of a network trained on a rich dataset, such as ImageNet[34], then retrain just the fully-connected classification layers on the new data while keeping the feature extraction layers frozen. Unfortunately, an analog to ImageNet is not available for SAR, so this particular method is ineffective for the problem at hand.

Despite the differences between measured and synthetic data, there are still many similarities between the two datasets, and neural networks certainly have the capacity to leverage that shared information. The experiment in this section leverages a simple CNN architecture based on LeNet [35] to perform joint domain training. The different layers of the network are presented with data from either the measured or the synthetic data in order to learn features from both domains. In general for these experiments, some or all layers of the network are trained primarily on synthetic data, while measured data is used to train the first layers. By so doing, the layers nearest the input learn information that is unique to the measured data while the later layers learn the overall structure of the targets.

Although more recent architectures such as ResNet[36] and DenseNet[12] have shown improved classification performance and faster training convergence, the separation of lower and higher layers in the network architecture is less clear. In the more advanced networks, the layers become mixed either through skip connections, such as in ResNet, or through the concatenation of early layers with later layers, as in DenseNet. LeNet's simple architecture is straightforward to analyze as the small number of layers can be more easily interpreted as corresponding to simple/local to more complex/global features. However, the LeNet derivative architecture used for this study is updated with advances since the LeNet architecture was developed including batch norm[37], dropout[38], and layer widening[39] to provide higher performance.

During training, the network (shown in Figure 1.8) was presented with several combinations of measured and synthetic data. The training began with a simple 50/50 split of both data domains into training and testing. During testing, the synthetic test data was not used. Furthermore, measured data for the T72 and ZSU23 were excluded entirely during training. By so doing, this experiment serves to test the ability of this transfer learning strategy to leverage synthetic information to correctly classify measured data.

Each experiment was defined by the set of layers that were trained only on measured data, the set trained on synthetic data, and the set of layers trained on both types of data. Each training batch was presented to the network in up to three parts

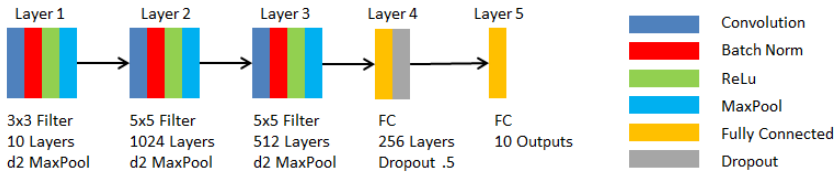


Figure 1.8: Illustration of the architecture used in this study including the composition and parameterization of each layer.

(namely measured, combined, or synthetic data). Notationally, this procedure is defined as a combination of $M^{set\_of\_layers}$, $X^{set\_of\_layers}$, and $S^{set\_of\_layers}$, respectively, where set_of_layers denotes the layers of the architecture that were trained with that data. The layers not indicated by the notation were frozen during that stage of training. For example, M^{12345} denotes that all layers of the network were trained and updated by measured data. $M^{12345}S^{45}$ denotes that the CNN was trained initially with measured data on all layers, then subsequently trained with synthetic data where only layers 4 and 5 were updated and the first three layers were frozen. $S^{12345}X^{1}S^{2345}$ denotes that the network was trained with synthetic data on all layers, then the first layer was trained with mixed measured and synthetic data, and finally the last four layers were trained with synthetic data while layer 1 was frozen.

For each training stage of the training sequence, stochastic gradient descent was used to optimize the weights, with an initial learning rate of .001 on a cosine weighted schedule [40]. The batch size was 21 and the number of epochs was 150. A cross-entropy loss function with soft targets[40] was used.

Thirty different training sequences were evaluated and are given in the table in Figure 1.9a. Along the top of table are the headings: Layer 1; Layer 1,2; and Layer 1,2,3. These denote the layers that are frozen for synthetic data and updated for the measured or mixed data, which is also represented in the notation in the table. The headings on the left side of the table describe the training sequences used for a particular training strategy.

For each experiment, the results are presented in the corresponding cell in Figure 1.9b. The first number in each cell indicates the classification performance across all ten measured targets, including the held-out T72 and ZSU23. The second number presents the accuracy on just the held-out classes. The results in the first row titled "Mea - All Targets" demonstrates that the CNN architecture is capable of getting near-perfect results given measured training data for all targets. The rest of the results correspond to networks that were trained on all classes of synthetic data and eight classes of measured data.

The table illustrates that the "Layer 1" column outlined in red gives the highest second percentage, or transfer learning performance, for the T72 and ZSU23. It also illustrates that the "Layer 1,2,3" column gives the best total measured performance but at the expense of the transfer learning performance. In addition, the "Syn, Meas, Syn" row outlined in blue gives the best overall performance of all the sequences tested. The results represented by the experiments at the intersections of the two

	All Layers	Layer 1	Layer 1,2	Layer 1,2,3
Mea – All Targets	M^{12345}			
Syn	S^{12345}			
Syn, Mea		$S^{12345}M^{1}$	$S^{12345}M^{12}$	$S^{12345}M^{123}$
Syn, Mea, Syn		$S^{12345}M^{1}S^{2345}$	$S^{12345}M^{12}S^{345}$	$S^{12345}M^{123}S^{45}$
Mea	M^{12345}			
Mea, Syn		$M^{12345}S^{2345}$	$M^{12345}S^{345}$	$M^{12345}S^{45}$
Mea, Syn, Mea		$M^{12345}S^{2345}M^{1}$	$M^{12345}S^{345}M^{12}$	$M^{12345}S^{45}M^{123}$
Mix	X^{12345}			
Mix, Syn		$X^{12345}S^{2345}$	$X^{12345}S^{345}$	$X^{12345}S^{45}$
Mix, Syn, Mea		$X^{12345}S^{2345}M^{1}$	$X^{12345}S^{345}M^{12}$	$X^{12345}S^{45}M^{123}$
Mix, Syn, Mix		$X^{12345}S^{2345}X^{1}$	$X^{12345}S^{345}X^{12}$	$X^{12345}S^{45}X^{123}$
Syn, Mix		$S^{12345}X^{2345}$	$S^{12345}X^{345}$	$S^{12345}X^{45}$
Syn, Mix, Syn		$S^{12345}X^{1}S^{2345}$	$S^{12345}X^{12}S^{345}$	$S^{12345}X^{123}S^{45}$

(a)

	All Layers	Layer 1	Layer 1,2	Layer 1,2,3
Mea – All Targets	99.0, 100			
Syn	69.4, 62.7			
Syn, Mea		74.7, 65.5	74.6, 4.35	80.22, 1.0
Syn, Mea, Syn		75.3, 73.4	79.1, 71.7	90.7, 65.9
Mea	79.2, 0.0			
Mea, Syn		86.1, 54.7	85.9, 53.7	90.2, 48.3
Mea, Syn, Mea		85.7, 45.7	80.7, 10.3	80.4, 4.6
Mix	86.6, 36.4			
Mix, Syn		91.2, 60.1	90.9, 59.4	91.6, 62.6
Mix, Syn, Mea		90.6, 57.6	87.7, 42.5	87.8, 40.3
Mix, Syn, Mix		90.8, 58.2	89.3, 50.7	87.6, 40.2
Syn, Mix		74.8, 66.9	80.6, 35.1	88.1, 42.2
Syn, Mix, Syn		74.9, 71.4	83.9, 70.4	90.1, 52.2

(b)

Figure 1.9: A description of the thirty different experiments performed (a) and the results (b). For the results, the first number gives the overall classification performance when tested on measured data, while the second number gives the performance on the withheld T72 and ZSU23 classes.

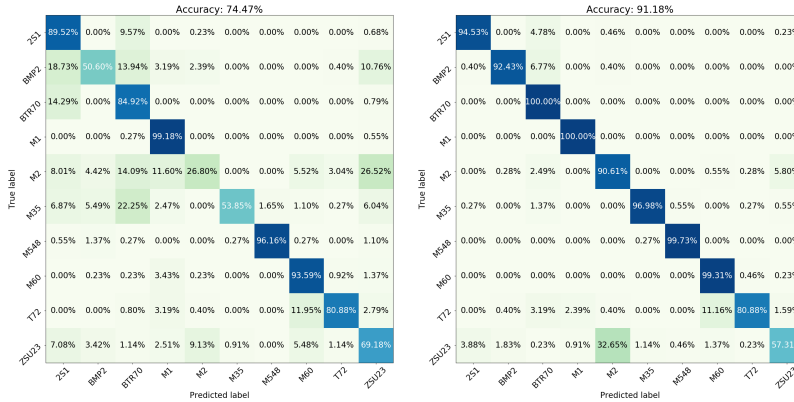
red columns and the blue row outlined in black are further detailed by the confusion matrices shown in Figure 1.10.

Based on the results in Figure 1.9, the first layer has the most generic and most transferable features. In contrast, when the third layer is included, the features become less transferable but more discriminative. These observations are consistent with layer complexity analysis performed using layer visualization for visible wavelength sensors[41]. Note that the M2 target was problematic. For the layer 1 experiment in Figure 1.10a, the M2 clearly had poor performance, and in the layer 1,2,3 experiment in Figure 1.10b, the M2 was a clear confuser for the ZSU23.

Although thirty different training strategies were compared and clear trends were observed, there are clearly many additional training strategies and networks that could be investigated using these training principles. The absolute numbers, though encouraging, are likely not reflective of current state of the art or future potential. Even so, care was taken in the experimentation so that relative performance numbers are informative. The lessons learned from building on the two trends of layer transferability sequence transferability (represented by the columns and rows of Figure 1.9b respectively) may provide fruitful directions for future research in transfer learning as applied to the synthetic/measured gap challenge.

1.4.4 Autoencoders

Autoencoders are one of the simplest deep learning algorithms. This type of network is designed to compress input data to a smaller representation, then decompress the representation back to a close reconstruction of the input data. Autoencoders are composed of a connected encoder and decoder network (see Figure 1.11), which are generally fully-connected or convolutional networks. The loss metric for the network penalizes reconstruction errors between the input and the reconstructed data.



(a) (b)

Figure 1.10: Confusion matrices corresponding to $S^{12345}M^{1}S^{2345}$ (a) and $M^{12345}M^{123}S^{45}$ (b). The weighted accuracy for the withheld T72 and ZSU23 classes in (a) is 73.4%, and the weighted accuracy in for the same two classes in (b) is 65.9%.

Autoencoder networks are useful because they extract the most salient features of data and represent them as simple numerical vectors.

In theory, this power to extract the most salient features of an image can help represent the two domains in a single, combined way. An autoencoder trained on measured and synthetic data should learn to represent both domains, although imperfectly. While the latent representation of each domain would not necessarily have any inherent adjacency for the two domains, the decompressed output of the autoencoder would represent, in some way, a unified image space. A classifier trained on the decompressed data would then enjoy a measure of agnosticism to the source of the data.

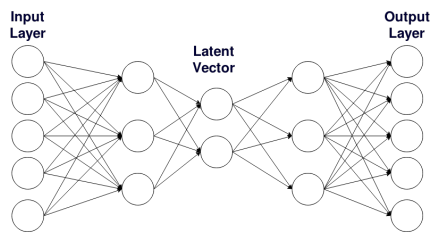


Figure 1.11: A graphical representation of an autoencoder. Autoencoders are composed of layers that compress the input to a one-dimensional latent vector, then decompress the latent vector back to a reconstruction of the input. The structure of the network is generally symmetric around the latent vector.

In practice, though, this is difficult to achieve. A simple autoencoder is not sufficient to bridge the synthetic/measurement data gap. In particular, missing data

(e.g., one target not being represented) in one of the two domains means that the autoencoder is unable to effectively represent that target in both domains.

To show this, a convolutional autoencoder was trained with a mix of synthetic and measured data, with the goal of learning an intermediate representation of the data. Alternating batches of the two types of data were presented to the network during training.An L2 loss was applied to the output of the network. This loss was computed against both the input image and the aligned equivalent image in the other domain, where available. By evaluating the loss against both images, the network was encouraged to learn domain independence in its representation. After training for 100 epochs, inference was performed on held-out data.

When the network was trained on all classes of data, the difference between autoencoded synthetic and measured data was minimal, as shown in Figure 1.12a. When measured data for two targets, the T72 and BTR70, was excluded from training, artifacts from the synthetic imagery were propagated during inference (see Figure 1.12b).

A DenseNet classifier was then trained on synthetic autoencoded data, then tested on measured autoencoded data. Although the two autoencoded manifolds should be very similar, the classification results show that this is not the case. In particular, the classification accuracy for withheld classes (Figure 1.12d) was unimpressive, and even as bad as chance guessing for the T72.

As a more advanced architecture is clearly needed to address missing data, a "manifold-joining autoencoder" was developed. This architecture, shown in Figure 1.13, builds on the idea of an autoencoder by adding a convolutional discriminator. The autoencoder portion of the network functions in a traditional way, i.e., by reconstructing input data. The addition of the discriminator adds an additional requirement that the autoencoded image look like measured data. Doing so encourages the network to break the symmetry between the input and the reconstructed image. When trained, the manifold-joining autoencoder serves to convert synthetic data into the measured realm, where it can more efficiently be used to train a classifier for measured data.

The manifold-joining autoencoder was trained in three repeated steps, each designed to make a part of the architecture perform in a certain way. In the first step, the discriminator was trained, using a binary cross-entropy loss, to distinguish between unaltered measured and simulated images. By so doing, the network learned the difference between the two domains, which can provide useful gradient information at later steps.

In the second step, the autoencoder was trained on synthetic images. A mean-squared error between the input $\mathcal{S}$ and its reconstruction $\hat{\mathcal{S}}$ was computed. The output $\hat{\mathcal{S}}$ was also classified by the discriminator as measured or synthetic, with a "synthetic" designation penalized and a "measured" label rewarded. These combined losses served as a gradient for the encoder and decoder portions of the autoencoder, with the discriminator loss given a few orders of magnitude less weight. This step is represented by the red arrows in the figure.

Finally, the synthetic image was passed through the updated autoencoder. $\hat{\mathcal{S}}$ was classified by the discriminator, and a binary cross-entropy loss was computed

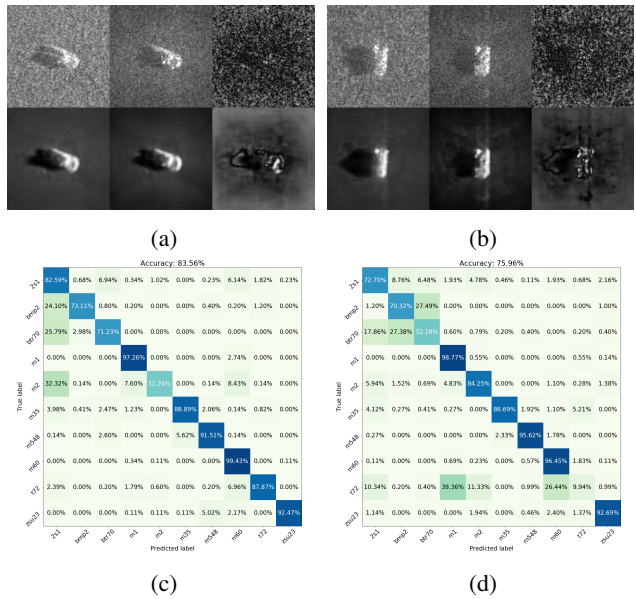


Figure 1.12: Output from an autoencoder designed to represent measured and synthetic images in an intermediate space. Blocks (a) and (b) show sample images of a T72. The top row shows the input images, while the bottom shows the autoencoded images. The left column shows real images, middle shows synthetic images, and the right shows the difference (autoscaled to emphasize differences; (b) has greater disparity). For (a), all measured and synthetic images were present during training. In (b), the T72 and BTR70 measured data was replaced by synthetic data. Blocks (c) and (d) show confusion matrices when classifying the output of the autoencoder in experiments in (a) and (b). Note the poor performance on the withheld targets in (d).

again. In contrast to the previous steps, this loss was only used to update the weights of the decoder, serving to enforce a measured-like style to the images. This step is represented by blue arrows in the figure.

After the model was trained, it was used to convert synthetic data to the measured domain. These outputs are shown in Figure 1.14, which shows a similar experiment to that performed with the simple autoencoder. It can be seen that the manifold-joining autoencoder does not blur the image nearly as much as the simple autoencoder. A DenseNet classifier was trained on the adjusted synthetic images and tested on unaltered measured images (in contrast to the simple autoencoder, where train and test images were both autoencoded). The performance of this classifier is given in the confusion matrices in Figures 1.14c and 1.14d for cases in which all ten measured classes were present (1.14c) and in which the T72 and BTR70 were removed (1.14d). While the classification results in the withheld data case are not impressive, there is potential for this architecture due to its ability to make synthetic images look more realistic.

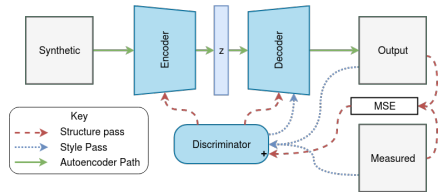


Figure 1.13: A manifold-joining autoencoder. This architecture extends the traditional autoencoder by adding a discriminator network. The discriminator works to distinguish between an autoencoded synthetic image and measured images, which allows the network to adjust the style of the autoencoded image.

In summary, autoencoders show some promise in bringing the two domains together, but suffer from an inability to represent missing data. Adding a discriminator to the autoencoder encourages greater variation in the encoded output, but this is also unable to fully solve the problem. More details about the manifold-joining autoencoder are available in reference 42.

1.4.5 Generative Adversarial Networks

Generative adversarial networks (GANs) are a relatively new method in the field of machine learning. These networks, which were introduced in 2014 by Ian Goodfellow and his collaborators[43], are designed to create new data that in some form mimics the statistical properties of a given set of training data. Given a target dataset, such as celebrity faces[44] or categories from the ImageNet[34] dataset, a GAN can be trained that generates new, unseen data that (ideally) fit comfortably and indistinguishably in the dataset. Since the introduction of GANs, several variations of the architecture[45] and many theories to help train these inherently unstable networks[46] have been developed.

In general, GANs are composed of generator and discriminator neural networks (Figure 1.15) which, for imagery data, are typically convolutional networks. Training is accomplished by repeatedly presenting the networks with data from a target dataset. The generator is tasked with learning to convert random n-dimensional vectors to data matching the dataset. The discriminator, in turn, is tasked with distinguishing between data from the dataset and the generator's output. In a descriptive analogy offered by Goodfellow *et al*, the generator can be likened to an art forger, whose goal is to create undetectable forgeries of the world's great artists. The discriminator plays the role of a detective, trying to discover which pieces are real and which are fakes.

The loss function for a GAN is given by

$$\begin{aligned}\min_G \max_D L(D,G) &= \mathbb{E}_{x\sim p_r(x)}\left[\log(D(x))\right] + \mathbb{E}_{z\sim p_z(z)}\left[\log(1-D(G(z)))\right] \\ &= \mathbb{E}_{x\sim p_r(x)}\left[\log(D(x))\right] + \mathbb{E}_{x\sim p_g(x)}\left[\log(1-D(x))\right]\end{aligned} \tag{1.2}$$

where $G(z)$ is the output of generator network, $D(x)$ is the output the discriminator network, z is a multi-dimensional random input to the generator, p_z is the distribution

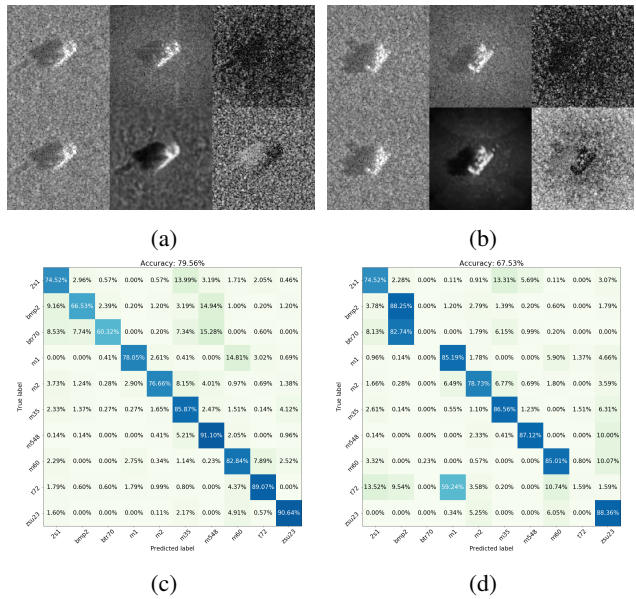


Figure 1.14: Representative output from a manifold joining autoencoder. The figures are set up in a similar manner as in Figure 1.12. Although the encoded difference is greater than those in Figure 1.12, the stylistic attention to detail is encouraging and the output of the autoencoder looks more like a measured image.

of z (usually uniform), and $p_g(x)$ and $p_r(x)$ are the probability manifold distributions for the generated data and the target dataset, respectively. Via backpropagation, these objectives direct the generator to create data that fits well with the dataset, while simultaneously increasing the distinguishing power of the discriminator. It can be shown[47] that, for a GAN with sufficient capacity, this training objective minimizes the Kullback-Leibler divergence between $p_g(x)$ and $p_r(x)$. This divergence metric describes how similar two probability distributions are, with low values denoting greater similarity. In other words, training a GAN creates a generator that is able to mimic the distribution of data in the given dataset at some level.

SAR images generated entirely from scratch are likely to be physically unrealistic in comparison to data generated by computational electromagnetic software. However, there are a few varieties of GANs that are well-suited to altering existing synthetic data in useful ways. In particular, the class of image-to-image translation GANs - such as Pix2Pix[48] and its extensions[49, 50, 51] - are formulated to translate images between domains, such as from grayscale images to color. This type of network can be used to translate synthetic SAR images to the measured domain and are designed to preserve the structure of the original image through the transformation. In a similar vein, methods such as SimGAN[52] aim to add realism to synthetic imagery by making small alterations to the image but do not explicitly translate im-

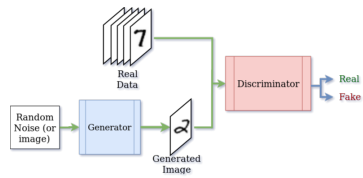


Figure 1.15: A diagram of a generic generative adversarial network. The network shown here is designed to produce new images of handwritten MNIST digits. The generator converts random noise into images that attempt to match the data from the target dataset. The discriminator distinguishes between real and generated data.

ages between domains. SimGAN has been used by other researchers for similar SAR image realism work[53].

As a brief example, this section explores DualGAN[49], which is based on Pix2Pix, as a way to make synthetic SAR data more realistic by translating between the two domains. Details of this experiment can be found in reference 54.

DualGAN[49], which is depicted in Figure 1.16, is designed to learn a mapping between two domains of imagery (e.g., real and synthetic SAR images). The architecture consists of two identical GANs, each assigned to one of the two data domains. The generators are composed of mirrored downsampling and upsampling convolutional layers with skip connections, in the style of U-Net[55]. In contrast to an autoencoder, the skip connections preserve information about the original structure of the image by concatenating pre-downsampled information to the upsampling layers, allowing the reproduction of important image details. The discriminators in each GAN learn to distinguish between data from each of the domains, and provide a gradient for the generators to generate data of the appropriate style. Importantly, DualGAN does not require paired data samples during training, as some image-to-image networks do. Instead, the generators learn the differences between the two domains from the dataset as a whole.

Let G_M be a generator mapping from the synthetic to the measured domain, D_M be a discriminator that determines whether an image belongs to the measured domain or not, and G_S and D_S be matching complementary networks operating on synthetic data.

DualGAN training is accomplished by converting a batch of synthetic images, S, to measured images with the measured generator, i.e., $\hat{M}_S = G_M(S)$. The discriminator D_M compares the translated images, $\hat{M}_S$, with samples of measured images, M, and computes the domain-specific loss accordingly. The translated images $\hat{M}_S$ are then translated back to the synthetic domain, $\hat{S} = G_S(\hat{M}_S)$. The difference between S and $\hat{S}$ is computed and weighted, creating the cycle-consistent loss metric. A similar path, $M \rightarrow G_S(M) \rightarrow \hat{S}_M \rightarrow G_M(\hat{S}_M) \rightarrow \hat{M}$, is followed for measured images. As both paths through the networks are repeated, the network learns a symmetric mapping between domains. Once trained, the generator G_M can translate synthetic images to measured images.

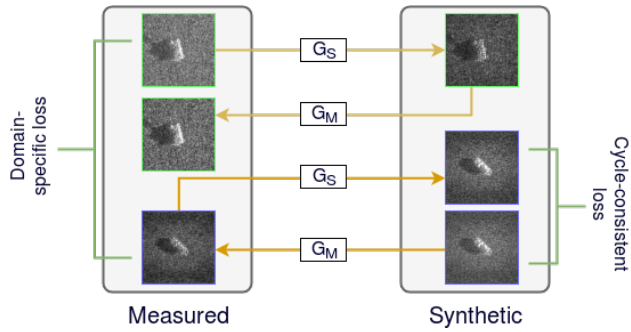


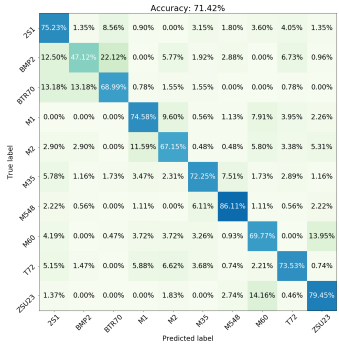
Figure 1.16: A schematic of DualGAN. The boxes denote the measured and synthetic domains. Translation operations in each direction are shown; the label on the arrow signifies a pass through a given generator. The first image in each chain is a non-altered image. The next image in the chain shows the same image translated into the other domain. Finally, the image is translated back into the original domain, completing the cycle. Loss metrics are defined between the reconstructed and original images as well as via the discriminators in the GANs.

Initial experiments with the network as formulated showed that it was unable to convincingly reproduce important details (e.g., the target) in the SAR images. After further experimentation, the loss function in (1.3) was added to the generator loss. This loss metric removes the $|\,x\,|^{1/4}$ normalization applied to the images, causing the bright parts of the image (namely the target) to pop out. This is then used as a mask on the L1 difference between the input image and the reconstructed image. By so doing, the network ensures that accurate reconstruction of the bright, target areas is prioritized by the network, which leads to better retention of target features.

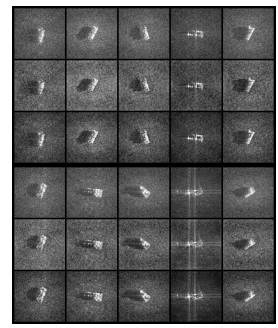
$$\lambda_k \sum_x |(u_x)^4 \times \left(u_x - G_V(u_x, z)\right) \mid \forall x \in u, u_x \in \{0,1\} \tag{1.3}$$

With a trained DualGAN network modified according to (1.3), the synthetic images can be transferred to the real domain using the trained generator. As an example of how well this method works, a DenseNet classifier was trained on the converted synthetic data and tested on non-altered measured images. Randomly-chosen images that were translated by the trained DualGAN, as well as the confusion matrix for the DenseNet classifier, are presented in Figure 1.17.

While DualGAN is not the only GAN method by which synthetic data can be made to look more like measured data, this example is representative of this method. Further research is currently underway with other networks, including SimGAN [52, 53], BicycleGAN [51], and the Auxiliary Classifier GAN [56]. While space does not allow for a detailed explanation of these algorithms, the approach is similar to that presented here. For more details on the application of these networks to this problem, please see references 54 and 42.



(a)



(b)

Figure 1.17: (a) A confusion matrix showing the results of classifying measured images with a DenseNet trained on synthetic images that were altered by the DualGAN network as described in this section. Compare these results to those in Figure 1.5b. All classes of measured data were used in training the DualGAN. 75% of the synthetic data were used to train the DualGAN, while the other 25% were altered using the trained DualGAN network and used to train the classifier. (b) A random sample of one image from each of ten classes. The first and fourth rows show original synthetic images, S. Rows two and five show the conversion to the measured domain, $\hat{M}_S$. The remaining rows show the conversion back to the synthetic domain, $\hat{S}$.

1.4.6 Siamese and Triplet Networks

While CNNs have shown great classification performance in general, their main weakness lies in their lack of ability to generalize [57].[3] Differences between the training and test data (such as the difference between measured and synthetic SAR data) lead to poor performance even with networks that have been carefully trained and exhibit good validation accuracy. CNNs simply are not well-suited for evaluating data outside of the domain on which they were trained. The confusion matrix in Figure 1.5b is a prime example of this problem, showing that a DenseNet trained on synthetic data is inadequate for classifying measured data.

Siamese and Triplet neural networks are classification networks designed to learn about data from two domains. These networks are presented with sets of images from each domain and learn to identify whether these sets represent the same or different class of item. The output of both types of network indicates whether the inputs belong to the same class or not. However, this binary decision can be transformed into a more traditional classification decision. Siamese [59, 60, 57, 61, 62] and Triplet [63, 64] networks consist of two and three identical neural networks, respectively, which share weights. By jointly sharing and updating the weights, the networks are able to form a shared representation of all the data. The classification layers at the end of the network enable it to identify data from matched and unmatched classes.

[3]Reference [58] shows a test of SNN generalizing but doesn't explicitly describe the CNN weakness.

These networks are an attractive way of solving the measured/synthetic gap because, rather than transforming the data before training a classifier (such as the case with GANs in Sec. 1.4.5 and autoencoders in Sec. 1.4.4), they are able to learn from both data manifolds and reach a decision despite the differences in the data.

1.4.6.1 Siamese Neural Network

A Siamese neural network (SNN) combines two neural networks (or twins) that have the same architecture. The network takes two inputs, one for each twin. The inputs represent either a true pair (for example, the images represent the same class) or a false pair (for example, the images represent different classes). Each twin's network has the same weights after training.

The loss function for the SNN is a similarity distance metric (such as the Euclidean distance between feature vectors), the value of which is "small" for the same type (class or model) of input and which is "large" for a different type of input. When training the network, each pair is classified by a value c that indicates whether the inputs are from the same class or not. Let $c = 1$ denote a true pair and $c = -1$ a false pair. Then for a given threshold τ for the margin of separation, the hinge-contrastive loss function (a common Siamese network loss function) [65] is calculated as

$$l = \left(\frac{1+c}{2}\right) x + \left(\frac{1-c}{2}\right) \max\{0, \tau - x\}, \tag{1.4}$$

where x is a distance metric between the embedded features produced by each twin network. It is evident that $l = x$ for a true pair and $l = \max\{0, \tau - x\}$ for a false pair. The parameter τ ensures that the network learns to separate true and false pairs by at least a given margin.

The number of pairs presented to the network is a function of the number of data points presented to each twin network, as well as the distribution of those data points into classes. Let there be N classes. Let m_i, $i = 1, \ldots, N$ be the number of images presented to twin 1, and let n_i, $i = 1, \ldots, N$ be the number of images presented to twin 2. Then the total set of images for twin 1 is $\mathcal{X}$, whose size is $|\mathcal{X}| = \sum_{i=1}^{N} m_i$, and the set presented to twin 2 is $\mathcal{Y}$, whose size is $|\mathcal{Y}| = \sum_{i=1}^{N} n_i$. Assume that any true-pair combination of $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ is not a duplication. Then the number of true pairs is

$$M_{\text{true}} = \sum_{i=1}^{N} m_i n_i, \tag{1.5}$$

and the number of false pairs is

$$M_{\text{false}} = \sum_{i=1}^{N} m_i \left(\sum_{j=1, j\neq i}^{N} n_j \right). \tag{1.6}$$

If the number of images for each target i in $\mathcal{X}$ and $\mathcal{Y}$ is balanced, then $m_i = n_i$ for all $i = 1, \ldots, N$. Then (1.5) and (1.6) become

$$M_{\text{true}} = \sum_{i=1}^{N} m_i^2 \tag{1.7}$$

and

$$M_{\text{false}} = \sum_{i=1}^{N} m_i \left(\sum_{j=1, j\neq i}^{N} m_j \right), \tag{1.8}$$

respectively. It is clear that the number of inputs $M_{\text{true}} + M_{\text{false}}$ for training a SNN grows approximately quadratically (depending on the distribution of images within classes) with $|\mathcal{X}| + |\mathcal{Y}|$.

Training the network involves generating pairs of SAR images for all targets; each pair is labeled as true (same class) or false (different classes). Ideally, all pairs would be composed of measured SAR data, as it is more self-consistent. Because this is generally not possible, one can augment the input data with synthetically generated SAR data. The full dataset need not be presented to both paired networks, which would be an exhaustive pairing of every image in the dataset with every other image. Doing so results in $\binom{n}{2}$ pairs, which can grow into the millions of pairs for more than a few thousand images and, in turn, increase the network training time dramatically. Instead, sets of data for each twin network should be chosen to present the data to the networks in some balanced fashion. For example, some pairs can be formed with one measured and one synthetic image of the same target, with a given ratio of measured and synthetic images, or to represent each class equally among all the pairs. After the pairs are constructed, the network is trained by presenting each pair to the twin networks and computing the forward pass through the convolutional layers and the similarity metric. Because the twins share network weights, the weight update is a function of the output of both networks.

Once the SNN is trained, there are two options for testing. The first, straightforward option involves matching an unknown image to one representative image from each trained class. Each of the N known images is paired with the unknown image, and each pair is presented to the network. The N scores from the SNN are compared with each other to determine which, if any, of the known targets is a good match for the unknown image. Zero, one, or many targets may be a good match for the unknown image, and further logic (for example, best score) could be used to make a class determination. Because the choice of representative image could affect the scoring, this mechanism could be repeated with a set of k representative images, producing $k \times N$ pairs and evaluations, and incorporate more complex decision logic. An example of such a classification system is shown in Figure 1.18.

The other option utilizes one of the twins (as each twin network has the same weights) and adds one or more fully-connected layers to the end. The pre-existing weights are then frozen and the newly-added fully-connected layers are trained on the existing data to create a traditional classifier.

As an example of how a Siamese network may be used, this architecture was trained on pairs of data from the SAMPLE dataset [58]. A modified AlexNet [66] CNN was used for each twin's architecture, with each convolution's weights initialized with a zero-mean Gaussian distribution with a small variance (0.01 or 0.05 in different experiments). This experiment consisted of data from five measured targets for training and testing and had a classification accuracy of 80.96% (see Figure 1.19). Experiments using synthetic data have not achieved classification accuracies

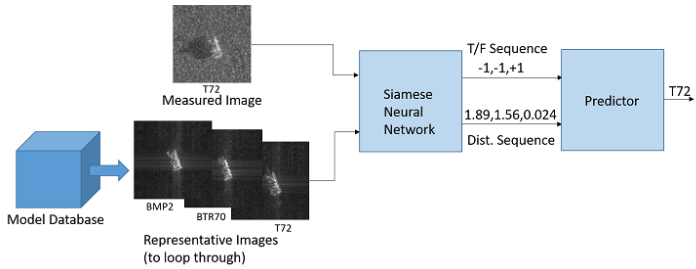


Figure 1.18: An example of a Siamese-based SAR ATR in which an example image of each vehicle type is compared to an unknown image. Those scores below a threshold of 1 indicate a match (that is, "+1"). The best match results from taking the minimum score from all the matches. Finally, the predictor identifies the class corresponding to the best match, in this case the T72. This figure is from reference 58.

that are better than chance. Experiments using all ten targets have had a similar lack of success.

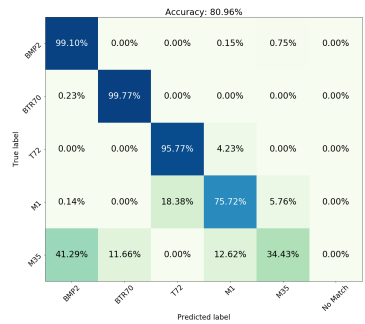


Figure 1.19: Siamese neural network classification results for a five-class problem. The network was trained on the first three targets and tested on the five targets shown in the confusion matrix. Note that this type of classifier features a "No Match" label.

1.4.6.2 Triplet Neural Network

A Triplet neural network (TNN) combines three neural networks (or triples) that have the same architecture. Triplet networks follow a similar training and data paradigm as Siamese networks—triples of data are constructed for training, and each network takes one of the images in the triple during training. As in Siamese networks, the three networks all share weights. The three images within the triple are designated the "anchor," "positive," and "negative" [67]. The triple combines the positive and negative pairs seen in Siamese networks into one group, where the anchor–positive pair is a true pair and the anchor–negative pair is a false pair. As is the case with Siamese networks, the three networks share the same weights. Both the anchor–

positive pair and the anchor–negative pair are combined by a distance metric, and these two intermediate values are then combined to create an overarching loss function. Just like for the SNN, the distance should be small for the anchor–positive pair and large for the anchor–negative pair. As described in [68], an advantage of a TNN over an SNN is not having to know from which class two images are; rather, the network just needs to have information on whether the classes are the same or not.

As opposed to a SNN loss function, a TNN loss function generally takes into account the Euclidean distances between the output of the anchor network and the positive and negative networks. That is, as in [64], suppose $\mathrm{Net}(X)$ represents the output of a Triplet network given one image in the triple $(X_\mathrm{a}, X_\mathrm{p}, X_\mathrm{n})$. The Euclidean distances for the anchor–positive network pair and the anchor–negative network pair are, respectively,

$$d_\mathrm{ap} = \|\mathrm{Net}(X_\mathrm{a}) - \mathrm{Net}(X_\mathrm{p})\|_2 \qquad \text{and} \tag{1.9a}$$

$$d_\mathrm{an} = \|\mathrm{Net}(X_\mathrm{a}) - \mathrm{Net}(X_\mathrm{n})\|_2. \tag{1.9b}$$

These Euclidean distances are then used to define a triplet loss function. One common iteration of this loss is

$$l = d_\mathrm{ap}^2 - d_\mathrm{an}^2 + \alpha \tag{1.10}$$

for some margin α [67]. This loss function encourages a large distance d_an between the X_a and X_n images, which are dissimilar, and a small distance d_ap between the X_a and X_p images from the same class. The α parameter drives a distance of $\sqrt{\alpha}$ between the two metrics.

A second common triplet loss function is

$$l = \|d_\mathrm{p}, d_\mathrm{n} - 1\|_2^2, \tag{1.11}$$

where

$$d_\mathrm{p} = \frac{e^{d_\mathrm{ap}}}{e^{d_\mathrm{ap}} + e^{d_\mathrm{an}}} \qquad \text{and} \tag{1.12a}$$

$$d_\mathrm{n} = \frac{e^{d_\mathrm{an}}}{e^{d_\mathrm{ap}} + e^{d_\mathrm{an}}} \tag{1.12b}$$

as provided in reference 64. The loss function in (1.11) has a similar aim as (1.10), pushing d_n toward a value of 1 and d_p closer to a value of 0. Like the first loss function, this encourages the metric between the anchor and positive images to be small and the metric between the anchor and negative images to be large. In contrast to SNN loss functions, the loss functions (1.10) and (1.11) do not require explicit positive and negative labels; rather, these labels are inherent to the network.

Additional loss functions for TNNs are defined and investigated in [68]. The authors describe two categories of loss functions as "difference losses" and "ratio losses," such that the positive and negative distances between the anchor–positive and anchor–negative networks, respectively, are compared by either differences or ratios, respectively.

At the time of writing, the authors have yet to perform significant, reportable experiments with a Triplet neural network, as they are still performing experiments

with Siamese networks. Nevertheless, a discussion of the theory of these networks is advantageous in this context for completeness. The number of possible triples (X_a, X_p, X_n) is also a limiting factor to experimentation. For N classes each with n images, the number of possible experimental triples is $N \times \frac{n!}{(n-2)!} \times (N-1) \times n$. For this reason, it is even more computationally intensive to experiment with this type of network than a Siamese network.

As with a SNN, there are two options for TNN testing. At the time of this writing, the writers have not fully investigated discrimination with Triplet networks, due to complexities in choosing anchor-positive and anchor-negative images and the size of the problem for non-trivial numbers of classes. However, classification may be performed much the same as with a Siamese network. For classification, a few fully-connected layers can be appended to one of the trained Triplet networks, and the fully-connected layers can be trained perform classification. In order to leverage the inherent functionality of the Triplet network, however, further exploration is required. Assuming that the unknown image is the anchor image, the prime difficulty lies in identifying a viable positive match to an unknown image. As anchor–negative–negative triples are far more probable than the ideal anchor–positive–negative triples, additional logic must be developed to effectively utilize the functionality of TNNs for classification.

1.5 Conclusion

The work of automatic target recognition for SAR is still a work in progress. The difficulties posed by this particular imaging method, including poor resolution, limited data, and the challenge of accurately creating simulated data create a challenging environment in which to work. In particular, the gap between synthetic and measured SAR data poses many challenges for deep learning classification systems, particularly due to the small size of available datasets.

Although the approaches in this chapter do not fully solve these challenges, they nevertheless represent an encouraging direction in which deep neural networks can help solve the measured/synthetic gap. This work with mathematical despeckling methods, autoencoders, layer retraining, generative adversarial networks, and Siamese networks represent a wide variety of possible approaches, and many more deep networks are available in the literature. Despite limited success to this point, these other approaches provide fertile ground on which to expand this work.

The authors acknowledge the hard work of many interns on these various projects, including (alphabetically) Julia Arnold, Michael Cline, James Cunningham, Omar DeGuchy, Devon Dollahon, Robert Friedlander, John Kaminski, Jennifer Liu, Brendan O'Connor, Joseph Sebastian, and Amy Wong.

References

[1] Sandia National Laboratory. MSTAR Overview; 1995. [Online; accessed 19-May-2017]. https://www.sdms.afrl.af.mil/index.php?collection=mstar.

[2] Han P, Wu R, Wang Y, et al. An Efficient SAR ATR Approach. In: IEEE International Conference on Acoustics, Speech, and Signal Processing. vol. 2; 2003. p. II–429.

[3] El-Darymli K, Gill EW, Mcguire P, et al. Automatic Target Recognition in Synthetic Aperture Radar Imagery: A State-of-the-Art Review. IEEE Access. 2016;4:6014–6058.

[4] Chen S, Wang H. SAR Target Recognition Based on Deep Learning. International Conference on Data Science and Advanced Analytics. 2014;.

[5] Chen S, Wang H, Xu F, et al. Target Classification using the Deep Convolutional Networks for SAR Images. IEEE Transactions on Geoscience and Remote Sensing. 2016;.

[6] Bhanu B, Jones G. Object Recognition Results using MSTAR Synthetic Aperture Radar Data. In: Proc. IEEE Workshop on Computer Vision Beyond the Visible Spectrum: Methods and Applications; 2000. p. 55–62.

[7] Lewis B, Scarnati T, Sudkamp E, et al. A SAR Database for ATR Development: Synthetic and Measured Paired Labeled Experiment (SAMPLE). In: Proc. SPIE Algorithms for Synthetic Aperture Radar XXVI. Baltimore; 2019. .

[8] Rosencrantz S, Nehrbass J, Zelnio E, et al. "AFacet": A Geometry Based Format and Visualizer to Support SAR and Multisensor Signature Generation. In: Proc. SPIE Algorithms for Synthetic Aperture Radar XXV. Orlando; 2018. .

[9] Scarnati T, Gelb A. Variance Based Joint Sparsity Reconstruction of Synthetic Aperture Radar Data for Speckle Reduction. In: Proc. SPIE Algorithms for Synthetic Aperture Radar Imagery XXV. vol. 10647. Orlando; 2018. p. 106470R.

[10] Fischler MA, Bolles RC. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Communications of the Association for Computing Machinery. 1981 Jun;24(6):381–395.

[11] Van Der Maaten L, Hinton G. Visualizing Data using t-SNE. Journal of Machine Learning Research. 2008;.

[12] Huang G, Liu Z, Van Der Maaten L, et al. Densely Connected Convolutional Networks. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition. Honolulu; 2017. .

[13] Kingma DP, Ba J. Adam: A Method for Stochastic Optimization. arXiv preprint arXiv:14126980. 2014;.

[14] Shwartz-Ziv R, Tishby N. Opening the Black Box of Deep Neural Networks via Information. arXiv preprint arXiv:170300810. 2017;.

[15] Pinel N, Boulier C. Electromagnetic Wave Scattering from Random Rough Surfaces: Asymptotic Models. John Wiley & Sons; 2013.

[16] Argenti F, Lapini A, Bianchi T, et al. A Tutorial on Speckle Reduction in Synthetic Aperture Radar Images. IEEE Geoscience and Remote Sensing Magazine. 2013;1(3):6–35.

[17] Mansourpour M, Rajabi M, Blais R. Effects and Performance of Speckle Noise Reduction Filters on Active Radar and SAR Images. In: Proc. International Society for Photogrammetry and Remote Sensing. vol. 36; 2006. p. W41.

[18] Chen DQ, Cheng LZ. Spatially Adapted Total Variation Model to Remove Multiplicative Noise. IEEE Transactions on Image Processing. 2012;21(4):1650–1662.

[19] Aubert G, Aujol JF. A Variational Approach to Removing Multiplicative Noise. SIAM Journal on Applied Mathematics. 2008;68(4):925–946.

[20] Lu J, Shen L, Xu C, et al. Multiplicative Noise Removal in Imaging: An Exp-model and its Fixed-point Proximity Algorithm. Applied and Computational Harmonic Analysis. 2016;41(2):518–539.

[21] Aubert G, Aujol JF. A Nonconvex Model to Remove Multiplicative Noise. In: International Conference on Scale Space and Variational Methods in Computer Vision. Springer; 2007. p. 68–79.

[22] Rudin L, Lions PL, Osher S. Multiplicative Denoising and Deblurring: Theory and Algorithms. In: Geometric Level Set Methods in Imaging, Vision, and Graphics. Springer; 2003. p. 103–119.

[23] Goodman JW. Statistical Optics. John Wiley & Sons; 2015.

[24] Irving WW, Ettinger GJ. Classification of Targets in Synthetic Aperture Radar Imagery via Quantized Grayscale Matching. In: AeroSense '99; 1999. p. 320–331.

[25] Paulson C, Wilson J, Lewis T. Synthetic Aperture Radar Quantized Grayscale Reference Automatic Target Recognition Algorithm. In: Proc. SPIE Algorithms for Synthetic Aperture Radar Imagery XXV. vol. 10647. Orlando; 2018. p. 106470P.

[26] Horvath M, Rigling B. Performance Prediction of Quantized SAR ATR Algorithms. IEEE Transactions on Aerospace and Electronic Systems. 2016;52(1):189–204.

[27] Sanders T, Gelb A, Platte RB. Composite SAR Imaging using Sequential Joint Sparsity. Journal of Computational Physics. 2017;338:357–370.

[28] Scarnati T, Zelnio E, Paulson C. Exploiting the Sparsity of Edge Information in Synthetic Aperture Radar Imagery for Speckle Reduction. In: Proc. SPIE Algorithms for Synthetic Aperture Radar Imagery XXIV. vol. 10201. Anaheim; 2017. p. 102010C.

[29] Archibald R, Gelb A, Platte RB. Image Reconstruction from Undersampled Fourier Data using the Polynomial Annihilation Transform. Journal of Scientific Computing. 2016;67(2):432–452.

[30] Gelb A, Scarnati T. Reducing Effects of Bad Data using Variance Based Joint Sparsity Recovery. Journal of Scientific Computing. 2019;78(1):94–120.

[31] Scarnati T, Lewis B. A Deep Learning Approach to the Synthetic and Measured Paired Labeled Experiment (SAMPLE) Challenge Problem. In: SPIE Algorithms for Synthetic Aperture Radar. Baltimore; 2019. .

[32] Knott EF, Schaeffer JF, Tulley MT. Radar Cross Section. SciTech Publishing; 2004.

[33] Tan C, Sun F, Kong T, et al. A Survey on Deep Transfer Learning. In: International Conference on Artificial Neural Networks. Springer; 2018. p. 270–279.

[34] Russakovsky O, Deng J, Su H, et al. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision. 2015;115(3):211–252.

[35] LeCun Y, et al.. LeNet-5, Convolutional Neural Networks; 2015. http://yann.lecun.com/exdb/lenet.

[36] He K, Zhang X, Ren S, et al. Deep Residual Learning for Image Recognition. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition; 2016. p. 770–778.

[37] Ioffe S, Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv preprint arXiv:150203167. 2015;.

[38] Srivastava N, Hinton G, Krizhevsky A, et al. Dropout: a Simple Way to Prevent Neural Networks from Overfitting. The Journal of Machine Learning Research. 2014;15(1):1929–1958.

[39] Zagoruyko S, Komodakis N. Wide Residual Networks. arXiv preprint arXiv:160507146. 2016;.

[40] Xie J, He T, Zhang Z, et al. Bag of Tricks for Image Classification with Convolutional Neural Networks. arXiv preprint arXiv:181201187. 2018;.

[41] Zeiler MD, Fergus R. Visualizing and Understanding Convolutional Networks. In: European Conference on Computer Vision. Springer; 2014. p. 818–833.

[42] Lewis B, DeGuchy O, Sebastian J, et al. Realistic SAR Data Augmentation using Machine Learning Techniques. In: Proc. SPIE Algorithms for Synthetic Aperture Radar XXVI. Baltimore; 2019. .

[43] Goodfellow I, Pouget-Abadie J, Mirza M, et al. Generative Adversarial Nets. In: Advances in Neural Information Processing Systems; 2014. p. 2672–2680.

[44] Liu Z, Luo P, Wang X, et al. Deep Learning Face Attributes in the Wild. In: Proc. of the IEEE International Conference on Computer Vision; 2015. .

[45] Hindupur A. The GAN Zoo; 2018. Online; accessed 03-Oct-2018. https://deephunt.in/the-gan-zoo-79597dc8c347.

[46] Creswell A, White T, Dumoulin V, et al.. Generative Adversarial Networks: An Overview; 2018.

[47] Weng L. From GAN to WGAN; 2017. Online; accessed 23-Aug-2018. https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html.

[48] Isola P, Zhu JY, Zhou T, et al. Image-to-Image Translation with Conditional Adversarial Networks. arXiv preprint arXiv:161107004. 2016;.

[49] Yi Z, Zhang H, Tan P, et al. DualGAN: Unsupervised Dual Learning for Image-to-Image Translation. arXiv preprint arXiv:170402510. 2017;.

[50] Zhu JY, Park T, Isola P, et al. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. arXiv preprint arXiv:170310593. 2017;.

[51] Zhu JY, Zhang R, Pathak D, et al. Toward Multimodal Image-to-Image Translation. In: Advances in Neural Information Processing Systems; 2017. p. 465–476.

[52] Shrivastava A, Pfister T, Tuzel O, et al. Learning from Simulated and Unsupervised Images through Adversarial Training. Proc IEEE Conference on Computer Vision and Pattern Recognition. 2017;p. 2242–2251.

[53] Cha M, Majumdar A, Kung HT, et al. Improving SAR Automatic Target Recognition using Simulated Images Under Deep Residual Refinements. In: IEEE International Conference on Acoustics, Speech, and Signal Processing. Calgary, Alberta, Canada; 2018. .

[54] Lewis B, Liu J, Wong A. Generative Adversarial Networks for SAR Image Realism. In: Proc. SPIE Algorithms for Synthetic Aperture Radar XXV. Orlando; 2018. .

[55] Ronneberger O, Fischer P, Brox T. U-Net: Convolutional Networks for Biomedical Image Segmentation. arXiv preprint arXiv:150504597. 2015;.

[56] Odena A, Olah C, Shlens J. Conditional Image Synthesis With Auxiliary Classifier GANs. arXiv preprint arXiv:161009585. 2016;.

[57] Koch G. Siamese Neural Networks for One-Shot Image Recognition; 2015.

[58] Friedlander RD, Levy M, Sudkamp E, et al. Deep Learning Model-based Algorithm for SAR ATR. In: Proc. SPIE. vol. 10647; 2018. p. 10647–10647–14.

[59] Bertinetto L, Valmadre J, Henriques JF, et al. Fully-Convolutional Siamese Networks for Object Tracking; 2016.

[60] Khalil-Hani M, Sung LS. A Convolutional Neural Network Approach for Face Verification. In: International Conference on High Performance Computing & Simulation; 2014. .

[61] Han X, Lueng T, Jia Y, et al. MatchNet: Unifying Feature and Metric Learning for Patch-based Matching. In: IEEE Conference on Computer Vision and Pattern Recognition; 2015. p. 3279–3286.

[62] Shaham U, Lederman RR. Common Variable Learning and Invariant Representation Learning using Siamese Neural Networks; 2016.

[63] Tzeng E, Devin C, Hoffman J, et al. Adapting Deep Visuomotor Representations with Weak Pairwise Constraints. In: Workshop on the Algorithmic Foundations of Robotics; 2016. .

[64] Hoffer E, Ailon N. Deep Metric Learning using Triplet Network. In: International Conference on Learning Representations; 2015. .

[65] Leal-Taixé L, Canton-Ferrer C, Schindler K. Learning by Tracking: Siamese CNN for Robust Target Association. In: IEEE International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). Deep Vision: Deep Learning for Computer Vision; 2016. .

[66] Krizhevsky A, Sutskever I, Hinton GE. ImageNet Classification with Deep Convolutional Neural Networks. In: Pereira F, Burges CJC, Bottou L, et al., editors. Advances in Neural Information Processing Systems. Curran Associates, Inc.; 2012. p. 1097–1105.

[67] Schroff F, Kalenichenko D, Philbin J. FaceNet: A Unified Embedding for Face Recognition and Clustering. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2015. p. 815–823.

[68] Liu Y, Huang C. Scene Classification via Triplet Networks. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing. 2018 Jan;11(1):220–237.