Deep Learning For RADAR Signal Processing

A Thesis

Presented in Partial Fulfillment of the Requirements for the Degree
Master of Science in the Graduate School of The Ohio State
University

By

Michael Wharton, B.S.

Graduate Program in Electrical and Computer Engineering

The Ohio State University

2021

Master's Examination Committee:

Dr. Philip Schniter, Advisor

Dr. Emre Ertin

Abstract

We address the current approaches to radar signal processing, which model radar signals with several assumptions (e.g., sparse or synchronized signals) that limit their performance and use in practical applications. We propose deep learning approaches to radar signal processing which do not make such assumptions. We present well-designed deep networks, detailed training procedures, and numerical results which show our deep networks outperform current approaches.

In the first part of this thesis, we consider synthetic aperture radar (SAR) image recovery and classification from sub-Nyquist samples, i.e., compressive SAR. Our approach is to first apply back-projection and then use a deep convolutional neural network (CNN) to de-alias the result. Importantly, our CNN is trained to be agnostic to the subsampling pattern. Relative to the basis pursuit (i.e., sparsity-based) approach to compressive SAR recovery, our CNN-based approach is faster and more accurate, in terms of both image recovery MSE and downstream classification accuracy, on the MSTAR dataset.

In the second part of this thesis, we consider the problem of classifying multiple overlapping phase-modulated radar waveforms given raw signal data. To do this, we design a complex-valued residual deep neural network and apply data augmentations during training to make our network robust to time synchronization, pulse width, and

SNR. We demonstrate that our optimized network significantly outperforms the current state-of-the-art in terms of classification accuracy, especially in the asynchronous setting.

Dedicated to my parents, Kevin and Bernadette Wharton, for their unconditional love, support, and encouragement.

Acknowledgements

First and foremost I would like to thank my supervisor, Prof. Phil Schniter, without whom this thesis would not have been possible. I will always be grateful for how much time and effort he contributed to mentoring me the past 3 years. His consistently high expectations and work ethic challenged me to become the best researcher, collaborator, and student I could; likewise, his door was always open to provide any guidance, insights, and discussion that would help me succeed. Prof. Schniter as well as many others including Prof. Lee Potter, Prof. Kiryung Lee, and Prof. Emre Ertin, have my thanks for teaching challenging and useful courses, providing research feedback, and interesting discussions. I also wish to thank Prof. Ertin for serving on my committee.

Within the IPS lab, Ted Reehorst, Subrata Sarkar, and Saurav Shastri welcomed me to the group, provided helpful insights, and fun conversations. Ted's friendship and willingness to always give advice was invaluable. From the Air Force Research Lab, Dr. Anne Pavy has wrangled datasets, provided helpful guidance, and her help on the job hunt has been more than generous; I am grateful for all of her support. Dr. Matt Scherreik and Chris Ebersole were interested in my work and provided useful discussions. I greatly enjoy our virtual "coffee breaks," especially during these difficult times.

I have many other friends to thank for making the past few years so great. Countless long nights of studying and fun weekends were enjoyed with Anthony Cistone, Brennon Pinion, Riley Sommers, and my brother, Dan Wharton. Thanks to Cj Busic, Logan McCarthy, Cole Holubeck, Andrew Guazzerotti, Megan Caroll, Joe Castrilla, Joe Wachob, and Brian Dillon for some great summers; see y'all at the next hog roast! Last, I want to thank Jessica Worsham, for bringing so much joy and happiness to my life and for making this final year the most memorable.

My studies were funded by the Center for Surveillance Research.

Vita

2020	B.S. Electrical and Computer Engineering, The Ohio State University
2019-present	Graduate Research Assistant, The Ohio State University

Publications

Journal Publications

M. Wharton, A. M. Pavy, and P. Schniter, "Phase-Modulated Radar Waveform Classification Using Deep Networks, " *IEEE Signal Processing Letters, submitted*

Conference Publications

M. Wharton, E. T. Reehorst, and P. Schniter, "Compressive SAR Image Recovery and Classification via CNNs," *Proc. Asilomar Conf. on Signals, Systems, and Computers*, Nov. 2019.

Fields of Study

Major Field: Electrical and Computer Engineering

Studies in:

Machine Learning	Prof. Philip Schniter
Radar Systems	Prof. Emre Ertin
Signal Processing	Prof. Kiryung Lee

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

In this thesis we investigate how deep neural networks (DNN), which currently provide state-of-the-art results in many domains, can be used to improve radar signal processing. There are two types of radar systems considered. The first is "actively sensing" radars, in which a radar system transmits a known signal, listens for the reflections, and analyzes those reflections in order to learn information about the surrounding environment. The second is "passively sensing" radar signals, in which a sensor is "listening" to the electromagnetic spectrum in order to identify signals of interest.

A synthetic aperture radar (SAR) operates as an active sensor, moving a radar through space while transmitting radar pulses at a given target and creating an image from the reflections, or echos. If these echos are measured at the Nyquist rate or higher, constructing the SAR image is straightforward and classification algorithms can be trained to recognize the target. However, for SAR imaging, we often work with sub-Nyquist sampled scenes, which is known as *compressive SAR*. The challenges now are, given compressive SAR measurements, can the true scene be well-estimated? If so, can the target still be recognized from our estimated scene? In Chapter 2 we develop a deep learning approach to address both of these challenges and show our approach outperforms the standard in compressed sensing.

In the passive sensing case, a sensor begins sampling incoming signals over time and attempts to classify the radar waveforms present in the collected samples. However, these samples will be corrupted with additive white Gaussian noise (low SNR), the received radar waveforms will have been transmitted at an unknown carrier frequency and will arrive at an unknown time, and there may be several radar systems simultaneously transmitting. Thus, detecting and classifying each waveform from just the collected samples is quite challenging.

Our experiments assume the transmitted radar waveforms are phase-modulated and on the same carrier frequency; we do not make assumptions on the time of arrival of the waveforms and consider classification of up to 4 simultaneously received waveforms. In Chapter 3 we design and extensively optimize a deep network, showing state-of-the-art classification accuracy.

We present concluding remarks in Chapter 4. First, we summarize the main contributions of this thesis and then identify several suggestions for future work.

Chapter 2: Compressive SAR Image Recovery and Classification via CNNs

2.1 Introduction

Synthetic aperture radar (SAR) uses a moving radar platform to transmit electromagnetic pulses and then uses the received echoes to estimate the scene reflectivity. We focus on spotlight SAR [1], where the radar continuously points at a given ground patch while transmitting and receiving pulses.

After demodulation, the sampled radar returns $\boldsymbol{r} \in \mathbb{C}^M$ can be expressed as [1]

$$\boldsymbol{r} = \boldsymbol{A}\boldsymbol{g} + \boldsymbol{w},$$

where $\boldsymbol{g} \in \mathbb{C}^N$ is a vector of 2D scene reflectivity samples that we aim to recover, $\boldsymbol{A} \in \mathbb{C}^{M \times N}$ is a linear operator, and $\boldsymbol{w}$ contains additive noise and clutter. With linear FM chirps, a uniform pulse repetition interval, and uniform sampling, $\boldsymbol{A}$ generates uniformly spaced samples along equi-spaced radial lines in 2D Fourier space, i.e., "polar format" samples [1].

When the samples $\boldsymbol{r}$ are taken at the Nyquist rate or above, $\boldsymbol{A}$ has full column rank, and thus $\boldsymbol{g}$ can be recovered using the least-squares approach

$$\widehat{\boldsymbol{g}} = (\boldsymbol{A}^{\mathsf{H}}\boldsymbol{A})^{-1}\boldsymbol{A}^{\mathsf{H}}\boldsymbol{r}. \tag{2.1}$$

In fact, $\widehat{\boldsymbol{g}}$ from (2.1) would perfectly estimate $\boldsymbol{g}$ in the absence of noise. In practice, it is common to approximate (2.1) by interpolating the polar-format samples $\boldsymbol{r}$ onto a Cartesian grid and then applying a 2D IFFT to the result.

Recently, it has been proposed to sample $\boldsymbol{r}$ below the Nyquist rate (i.e., use $M < N$), which is known as *compressive SAR* [2, 3]. For example, one may choose to transmit and receive only a (possibly random) subset of the usual pulses, known as "slow-time subsampling." There are several motivations for doing this. For one, there is less information to store and/or transmit back to the ground station. Another is that the radar could simultaneously image multiple targets. Several other applications, include increased robustness to jamming, are discussed in [2].

In the compressive case, $\boldsymbol{A}$ is a fat matrix, in which case SAR image recovery is more challenging. In particular, $\boldsymbol{A}$ has a non-trivial nullspace, and all components of $\boldsymbol{g}$ in that nullspace are lost when collecting the measurements $\boldsymbol{r}$. Likewise, the inverse in (2.1) does not exist.

The traditional approach to compressive SAR recovery exploits image sparsity in an appropriate basis [2,3], i.e., applies compressive sensing [4,5]. For example, if $\boldsymbol{g}$ is sparse in the canonical basis (i.e., $\boldsymbol{g}$ is itself sparse), then one may attempt to recover $\boldsymbol{g}$ from $\boldsymbol{r}$ by solving the convex problem

$$\widehat{\boldsymbol{g}} = \arg\min_{\boldsymbol{g}} \|\boldsymbol{g}\|_1 \text{ s.t. } \|\boldsymbol{r} - \boldsymbol{A}\boldsymbol{g}\|^2 \leq M\sigma^2, \tag{2.2}$$

which is known as basis pursuit (BP) denoising [6]. This optimization problem (2.2) is convex and first-order algorithms like SPGL1 [7] can efficiently solve it. Still, these methods are computationally intensive for practical image sizes, and the sparsity model on which they are based may not fully exploit the structure of SAR images.

Thus, one may wonder whether compressive SAR recovery can be performed using methods that are faster and/or more accurate.

2.2 CNN-Based SAR Image Recovery

We propose a convolutional neural network (CNN)-based approach to compressive SAR image recovery. In particular, we propose to first back-project the radar returns, yielding

$$\boldsymbol{z} \triangleq \boldsymbol{A}^{\mathsf{H}}\boldsymbol{r}. \tag{2.3}$$

With sub-Nyquist sampling, the back-projected scene $\boldsymbol{z}$ will be heavily aliased. We propose to then de-alias $\boldsymbol{z}$ using a CNN. Among the plethora of CNN architectures, we chose a U-Net [8], because of its excellent performance in other image-recovery tasks [9].

For simplicity, we input only the magnitudes of the elements in $\boldsymbol{z}$ to the CNN. We do this because, in our experience, image phase information in $\boldsymbol{z}$ does not improve classification accuracy, at least with the MSTAR dataset that we used for our experiments. Altogether, our image-recovery approach can be summarized as

$$\widehat{\boldsymbol{g}} = \boldsymbol{f}(|\boldsymbol{z}|; \widehat{\boldsymbol{\theta}}), \tag{2.4}$$

where $\boldsymbol{f}(\cdot; \widehat{\boldsymbol{\theta}})$ is the CNN, $\widehat{\boldsymbol{\theta}}$ is a vector of trained CNN parameters, and $|\boldsymbol{z}|$ denotes the vector composed of the element-wise magnitudes of $\boldsymbol{z}$. The output $\widehat{\boldsymbol{g}}$ of our CNN is also non-negative, and thus should be considered as estimate of $|\boldsymbol{g}|$ rather than of complex-valued $\boldsymbol{g}$.

A similar approach was proposed for compressive magnetic resonance imaging (MRI) in [10], but—to the best of our knowledge—no CNN-based methods have been proposed for compressive SAR image recovery. However, CNNs have previously been proposed for other SAR tasks, such as image segmentation [11], image de-speckling [12], and automatic target recognition (ATR) [13, 14].

By training our CNN to de-alias the results of many *different* slow-time sub-sampling patterns (for a given sampling rate $\delta = M/N$), it learns to be agnostic to the specific choice of the sub-sampling pattern. This way, we do not need to retrain the CNN when the sub-sampling pattern changes. We did train a different CNN for each sampling rate $\delta \in \{1/2, 1/3, 1/4, 1/5, 1/10\}$, however. To learn the CNN parameters $\widehat{\boldsymbol{\theta}}$, we minimized ℓ_1 loss in the image space, i.e.,

$$\widehat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \sum_{t_1=1}^{T_1} \sum_{t_2=1}^{T_2} \left\| \boldsymbol{f}\left(|\boldsymbol{A}_{t_1}^{\mathsf{H}} \boldsymbol{A}_{t_1} \boldsymbol{g}_{t_2}|; \boldsymbol{\theta}\right) - |\boldsymbol{g}_{t_2}| \right\|_1, \tag{2.5}$$

using stochastic gradient descent. In (2.5), $\{\boldsymbol{g}_t\}$ are scene reflectivities from a training database and $\{\boldsymbol{A}_t\}$ are slow-time randomly sub-sampled Fourier matrices. We used the ℓ_1 loss, as opposed to the ℓ_2 loss, because it is a more typical choice when training CNNs to perform image recovery tasks [15].

2.3 CNN-Based Automatic Target Recognition

In the previous section, our goal was to recover the SAR image $\boldsymbol{g}$. Often, the recovered image is subsequently fed to an image classifier for automatic target recognition (ATR). In this case, the resulting classification performance is the primary metric of interest.

To evaluate our compressive SAR image recovery method from the perspective of ATR, we trained a ResNet-18 image classifier [16] to perform classification. Our choice of ResNet-18 was inspired by the excellent performance previously reported in [14]. For example, we found that a ResNet-18 gave 99.06 % classification accuracy with noiseless, fully sampled MSTAR images.

We experimented with two different approaches to training the ResNet classifier. We used either

1. noiseless, fully sampled MSTAR images, or
2. reconstructed MSTAR images produced by either BP or the U-Net, as described in Section 2.2, at a given value of $\delta = M/N$.

In both cases, we used the standard cross-entropy loss when training the classifier. In the latter case, the images used for training are reconstructed after randomly subsampling a fully sampled image, thus, many training examples for the classifier are generated from one MSTAR image; i.e., many unique recoveries, $\widehat{\boldsymbol{g}}$, may be generated from a single image, $\boldsymbol{g}$. This substantially increases the number of training examples relative to the former approach and likely contributes, as we will see in the next section, the classification from compressive samples is much more accurate when the classifier is trained on compressively recovered images.

2.4 Numerical Results

For our numerical results, we assumed noiseless measurements, i.e.,

$$\boldsymbol{r} = \boldsymbol{A}\boldsymbol{g}. \tag{2.6}$$

with images $\boldsymbol{g}$ taken from the 10-class MSTAR dataset [17]. We used the 17°-inclination subset for training, which had 3 671 images and the 15°-inclination subset for testing, which had 3 203 images. Because the images are of various sizes, we first center-cropped them to size 128×128. To implement the polar-format Fourier operator $\boldsymbol{A}$, we quantized each point on each radial line to the nearest point on the 128×128 Cartesian grid. This allows us to approximate $\boldsymbol{A} \approx \boldsymbol{M}\boldsymbol{F}$, where $\boldsymbol{M}$ is a random masking operator and $\boldsymbol{F}$ is the 2D FFT operator. Various sampling ratios

$$\delta \triangleq \frac{M}{N} \tag{2.7}$$

were tested. Figure 2.1 shows an example of a mask at $\delta = 1/3$. Note the random subset of radial lines, with dense sampling across each line.

2.4.1 Image Recovery

As a baseline, we compare the proposed CNN-based method to BP recovery (i.e., equation (2.2) with $\sigma^2 = 0$) implemented using the SPGL1 algorithm [7]. We used the public MATLAB implementation of SPGL1[1] with default parameters.

Figure 2.1 shows an example of a Fourier-domain sub-sampling mask at sampling ratio $\delta = 1/3$, and Figure 2.2 shows an example of an MSTAR image. Figure 2.3 shows the result of backprojection, Figure 2.4 shows the SPGL1 recovery, and Figure 2.5 shows the U-Net recovery. The SPGL1 recovery loses many details in the target that are visible in Figure 2.2, while the U-Net recovery preserves those details. It is interesting to observe that the U-Net recovery has suppressed most of the speckle artifacts that are present in the original MSTAR image Figure 2.2.

Table 2.1 shows the normalized mean squared error (NMSE) on the recovered image magnitudes, averaged over the test data $\{\boldsymbol{g}_t\}_{t=1}^{T}$, i.e.,

$$\text{NMSE} = \frac{1}{T}\sum_{t=1}^{T} \frac{\left\|\widehat{\boldsymbol{g}}_t - |\boldsymbol{g}_t|\right\|_2^2}{\|\boldsymbol{g}_t\|_2^2}, \tag{2.8}$$

[1]MATLAB implementation of SPGL1 was downloaded from https://www.cs.ubc.ca/~mpf/spgl1/index.html

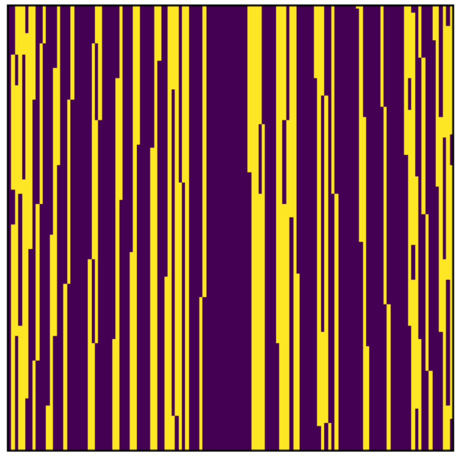

Figure 2.1: Fourier-domain sampling mask at $\delta = 1/3$.

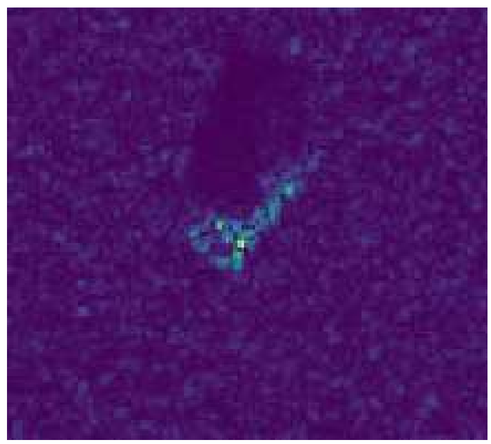

Figure 2.2: Example of original MSTAR image $|\boldsymbol{g}|$.

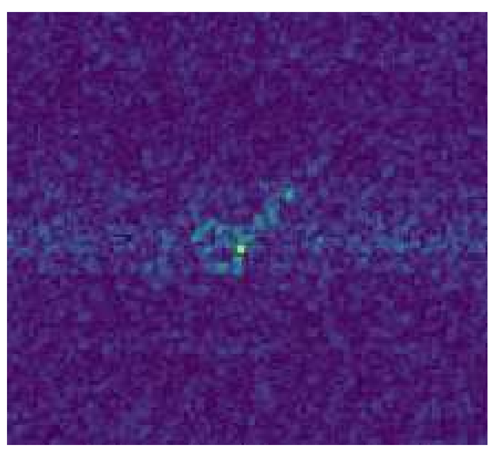

Figure 2.3: Back-projection image $\boldsymbol{z}$ at $\delta = 1/3$.

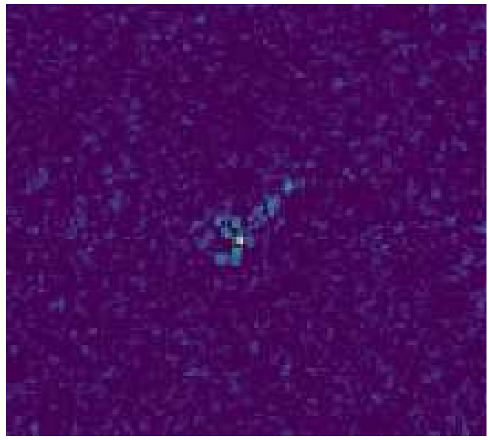

Figure 2.4: SPGL1 reconstruction $\widehat{\boldsymbol{g}}$ at $\delta = 1/3$. Note that a lot of detail in the target has been lost relative to Fig. 2.2.

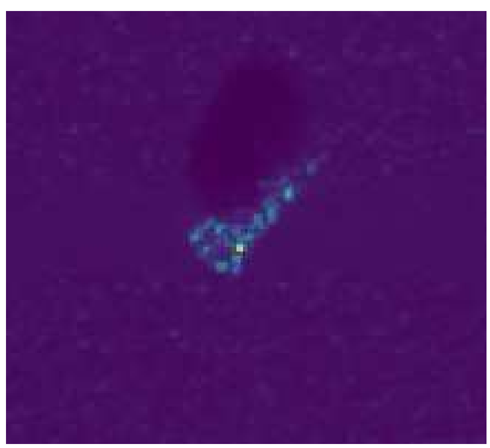

Figure 2.5: U-Net reconstruction $\widehat{\boldsymbol{g}}$ at $\delta = 1/3$. Note that target details have been preserved relative to Fig. 2.2 while speckle artifacts have been suppressed

Table 2.1: Average NMSE

δ	SPGL1	U-Net
1/2	-7.04 dB	**$-$10.63 dB**
1/3	-4.68 dB	**$-$10.12 dB**
1/4	-3.46 dB	**$-$8.43 dB**
1/5	-2.69 dB	**$-$8.11 dB**
1/10	-1.01 dB	**$-$6.92 dB**

Table 2.2: Reconstruction time

δ	SPGL1	U-Net
1/2	2.64 sec	**0.00451 sec**
1/3	2.76 sec	**0.00496 sec**
1/4	2.94 sec	**0.00460 sec**
1/5	2.96 sec	**0.00445 sec**
1/10	3.25 sec	**0.00472 sec**

where a different random subsampling mask was used for every test image. The table shows that the proposed U-Net recovery method greatly outperformed SPGL1 for all tested sub-sampling rates δ. We attribute the relatively poor performance of SPGL1 to the large amounts of speckle present in the original MSTAR images (see, e.g., Figure 2.2), which detract from the sparsity of the image.

Table 2.2 shows the reconstruction time on a Linux server with 24 Intel Xeon(R) Gold 5118 CPUs and a single Tesla V-100 GPU. The table shows that the proposed method ran more than 500 times faster than SPGL1.

Table 2.3: Average test classification accuracy using the ResNet classifier trained on fully sampled images

δ	SPGL1	Proposed
1/2	86.11 %	**87.42 %**
1/3	76.58 %	**88.29 %**
1/4	65.41 %	**86.73 %**
1/5	54.70 %	**85.76 %**
1/10	32.19 %	**74.37 %**

Table 2.4: Average test classification accuracy using the ResNet classifier trained on reconstructed images

δ	SPGL1	U-Net
1/2	96.85 %	**99.38 %**
1/3	94.01 %	**98.38 %**
1/4	90.67 %	**97.80 %**
1/5	86.58 %	**97.00 %**
1/10	70.12 %	**91.10 %**

2.4.2 Automatic Target Recognition

Table 2.3 shows test classification accuracy at different sub-sampling ratios δ for the ResNet classifier that was trained on noiseless, fully sampled MSTAR images. This table shows that the U-Net-reconstructed images lead to much better classification accuracy than the SPGL1-reconstructed images, but in both cases the classification accuracy is far from the 99% achieved by the ResNet on fully sampled test images (i.e., non-compressive SAR).

Table 2.4 shows classification accuracy at different sub-sampling ratios δ for the ResNet classifiers trained on reconstructed images. Note that a different classifier was trained for SPGL1 and for the U-Net at each sub-sampling rate δ. This table shows that, as before, U-Net image reconstruction leads to much better classification accuracy than SPGL1 image reconstruction. However, differently from before, Table 2.4 shows that the U-Net recoveries from compressive SAR lead to classification accuracies on par with fully sampled SAR. In fact, at a sampling rate of $\delta = 1/2$, U-Net recovery yields a classification accuracy of 99.38%, which is slightly better than that achieved in the fully sampled case.

2.5 Conclusion

In this chapter, we proposed a new approach to compressive SAR image recovery that used a convolutional neural network (CNNs) to de-alias sub-Nyquist sampled back-projection images. Importantly, by training with randomly subsampled images, our CNNs learned to become agnostic to the subsampling pattern. We showed for a given sampling rate, our CNN-based recoveries were at least 500 times faster and, in terms of normalize mean squared error, at least 3.59 dB better than standard BP. Furthermore, we showed CNN-based recoveries of compressive SAR measurements could be used to train a deep neural network (DNN) for automatic target recognition with accuracies similar to or outperforming a DNN trained with fully-sampled images.

Chapter 3: Phase-Modulated Radar Waveform Classification Using Deep Networks

3.1 Introduction

We consider the problem of classifying phase-modulated radar waveforms given raw signal data. Waveform classification is an important capability for cognitive radar. To train a classifier, we assume the availability of a dataset containing many examples of radar waveforms (i.e., sequences of complex-valued time-domain samples) with corresponding class labels in $\{1, \ldots, K\}$. In this work, we focus on deep neural network (DNN) classifiers, as they are state-of-the-art.

Several challenges are faced when designing DNN classifiers of raw radar waveforms, commonly referred to as "pulses." First, the pulse duration can span a very wide range (e.g., from hundreds to thousands of samples) even within a given class. Most DNNs, however, assume a fixed input dimension. Second, the pulses are complex-valued, whereas most DNNs are configured to support real-valued signals. Third, when applying the classifier, one cannot assume that the pulse will be time-synchronized, especially in passive sensing applications. Fourth, practical radar systems must operate over a wide range of signal-to-noise ratios (SNRs), including

SNRs far below 0 dB. To be practical, all of these challenges must be simultaneously addressed.

Early approaches to radar waveform classification first converted the raw waveforms to hand-crafted, low-dimensional features, to which classic machine-learning techniques could be applied. For example, [18] designed auto-correlation function (ACF) features of dimension 20 that were robust to time and frequency shifts, and trained a Fisher's linear discriminant classifier. These ACF features were later used in [19] with a support vector machine (SVM) classifier and in [20] with a shallow neural-network classifier.

It was recently demonstrated in [20] that significantly better classification accuracy could be obtained by training a DNN to operate on raw time-domain radar waveforms. This is not surprising, since the early DNN layers can learn a feature representation that outperforms hand-crafted ones, which is then classified by the final DNN layers.

Still, several assumptions were made in [20] that limited both the performance and practicality of the methodology. First, the DNN input dimension in [20] was chosen to be greater than the longest pulse in the training dataset. We will show that it is better to truncate or pad the pulses to an optimized input length. Second, the DNN in [20] ignored the quadrature (Q) input to avoid complex-valued operations. We show that a well-designed complex-valued DNN has advantages over a real-valued DNN. Third, the training and test waveforms in [20] were assumed to be time-synchronized, which is impractical. We train our DNN to be robust to time-asynchronous inputs. Fourth, the DNN architecture in [20] is far from optimal. We make an effort to optimize our DNN architecture (e.g., input dimension, depth, width, kernel size) and training procedure (e.g., batch size, learning rate). Additionally, we use data augmentation (i.e., random noise and delay realizations) to effectively increase the size of the training dataset. Fifth, [20] assumed a single radar pulse, whereas we also consider the problem of classifying multiple overlapping radar pulses.

3.2 Network Architecture

Like in [20], we focus on feed-forward convolutional DNNs. (Although we experimented with recursive networks, our initial results were not competitive.) In particular, we focus on deep residual networks (ResNets) [16] due to their excellent performance on related tasks. Although ResNets were originally proposed for classification of images, they can be easily adapted to one-dimensional signals by changing the two-dimensional convolutions to one-dimensional convolutions and appropriately modifying the kernel sizes and numbers-of-channels.

For single-pulse classification, we train with the standard cross-entropy (CE) loss. For multi-pulse classification, we use the same network architecture, but train using binary cross-entropy (BCE) loss on each of the K outputs.

3.2.1 Complex Arithmetic

Most DNNs support only real-valued arithmetic, which is sufficient when the data is real-valued. But our radar waveforms are complex-valued. The question is then how to best handle these complex waveforms.

Many approaches have been proposed to handle complex-valued signals with real-valued DNNs. Whether or not they are successful or not depends largely on the application. One of the simplest approaches is to feed the magnitude of the complex-valued signal to the DNN. Although this was successfully used in [21] to classify synthetic aperture radar (SAR) images, we found that it did not work well for classification of phase-modulated radar waveforms. Another simple approach is to feed only the real part of the complex-valued signal to the DNN. Although was successfully used in [20] to classify phase-modulated radar waveforms, we show in the sequel that it can be improved upon. Another approach is to stack the real and imaginary parts of the waveform into a real-valued 2-row "image" and feed it to a real-valued DNN with 2-dimensional convolutions [22]. Yet another approach is to feed the real and imaginary components into two separate input channels of a real-valued DNN, similar to how RGB image data is typically fed into three separate input channels.

An alternative is to design a complex-valued DNN. Such networks have led to improved performance in, e.g., audio classification [23] and magnetic resonance image (MRI) reconstruction [24] tasks. To understand what we mean by a "complex-valued DNN," consider the multiplication of a learnable parameter $k = k_r + ik_i \in \mathbb{C}$ with a feature $x = x_r + ix_i \in \mathbb{C}$, where $k_r, x_r \in \mathbb{R}$ represent real parts, $k_i, x_i \in \mathbb{R}$ represent imaginary parts, and $i \triangleq \sqrt{-1}$. Such multiplications arise in DNNs when implementing convolution and when linearly combining the convolution outputs. The complex-valued multiplication of k and x can be written using four real-valued multiplications as

$$kx = (k_r x_r - k_i x_i) + i(k_i x_r + k_r x_i). \tag{3.1}$$

The key point is that (3.1) is a two-input/two-output operation with only two learnable parameters: $k_r, k_i \in \mathbb{R}$. By contrast, a 2-channel real-valued DNN would implement

$$y_1 = k_{11}x_1 + k_{12}x_2 \tag{3.2}$$

$$y_2 = k_{21}x_1 + k_{22}x_2, \tag{3.3}$$

with $x_1 \triangleq x_r$ and $x_2 \triangleq x_i$, which involves four learnable parameters: $k_{11}, k_{12}, k_{21}, k_{22} \in \mathbb{R}$. Reducing the number of learnable parameters helps to mitigate overfitting.

With regards to the construction of complex-valued activation functions (which are essentially two-input/two-output memoryless nonlinearities), there are many options, as discussed in [23, 24]. Since both papers suggest that

$$\mathbb{C}\text{ReLU}(x) \triangleq \max\{0, x_r\} + i \max\{0, x_i\} \tag{3.4}$$

$$\text{for } x = x_r + ix_i$$

outperforms other complex-valued activations in many applications, we use (3.4) in our network.

Complex-valued implementations of batch-norm have also been developed (see, e.g., [24]). In our implementation, for simplicity, we apply standard batch-norm separately to the real and imaginary outputs.

3.3 Data Augmentation: Noise Padding, Truncation, and Delay

As mentioned earlier, our raw radar waveforms differ in duration from hundreds to thousands of samples, even within a single class. However, the ResNet DNN forces us to choose a fixed input dimension for minibatch training. The usual approach (e.g., routinely used in image classification) is to set the input dimension equal to the longest samples (e.g., largest images) and zero-pad the shorter samples as needed.

With noisy radar waveforms, it would be more appropriate to noise-pad rather than zero-pad. Thus, in [20], the waveforms were symmetrically padded with white Gaussian noise whose variance was selected to match the noise content of the original sample. An example is shown in Fig. 3.1.

There are several issues with the approach from [20]. First, as a consequence of symmetric padding, the padded waveforms will all be time-centered, i.e., synchronized. But since time synchronization is not expected in practice, it is not advantageous to train on time-synchronized waveforms. Second, noise-padding reduces SNR. In particular, if a sample is $P\times$ longer after noise-padding, then its SNR will change by the factor $1/P$. Third, the approach in [20] was to pad each training sample with a *fixed* noise waveform. In this case, the DNN could attempt to classify based on this noise realization, which leads to overfitting. This latter problem is exacerbated by the very low SNRs encountered in radar.

Our approach is to noise-pad *or* truncate the training waveforms as needed to obtain a fixed input length of D, where D is optimized. This approach recognizes that noise-padding decreases SNR, while truncation discards discriminative features, and so the optimal approach must balance between these extremes. To avoid injecting time-synchronization bias into the training procedure, we randomly delay the training waveform whenever noise-padding or truncating that waveform. We can describe this more precisely using N to denote the duration of the original training waveform and U to denote a random integer uniformly distributed from 0 and $|N - D|$. When $N < D$, we noise-pad the front of the waveform using U samples and noise-pad the back of the waveform using $D - N - U$ samples. (See Fig. 3.2.) When $N > D$, we keep D consecutive samples of the training waveform starting at index U. To avoid (over)fitting the DNN to particular training noise waveforms or training delays U, we draw new realizations of these quantities in every training minibatch (e.g., in the DataLoader of PyTorch [25]). This approach can be recognized as a form of "data augmentation" [26] that increases the effective number of the training samples. Finally, we optimized the input dimension D over a grid of possibilities, as described in the sequel.

3.4 Experimental Results

For our experiments, we use the SIDLE dataset, which was also used in [18–20]. This dataset contains 23 classes of phase-modulated radar pulses with 10000 examples of single pulses from each class. The details of each modulation type are given in Table 3.1. Furthermore, each pulse in the dataset was generated with a random initial phase and random intermediate frequency between 120 and 200 megahertz, although the individual pulses are not labeled with this information. For a fair comparison to [20], we omitted classes 6 and 19–23 in the original dataset and used only the remaining $K = 17$ classes to train and test our DNN. For these 17 classes, the dataset contains complex-valued waveforms with lengths from between 131 and 8925 samples and SNRs between –12 and +12 dB. For each experiment, we used a random subset of 80% of the dataset for training and the remaining 20% for testing. We used the Adam optimizer with default parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. In what follows, we discuss classification of single pulses in Sections 3.4.1 and 3.5 and multiple pulses in Section 3.6.

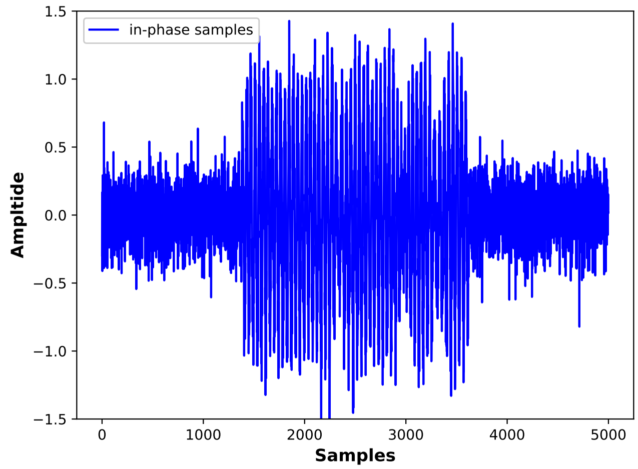

Figure 3.1: Example synchronous noise-padded waveform, at SNR=10.8dB, waveform length $N = 2261$, and DNN input length $D = 5000$.

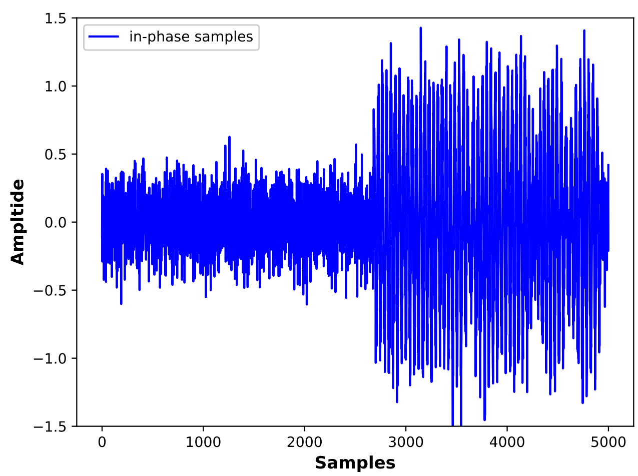

Figure 3.2: Example asynchronous noise-padded waveform, at SNR=10.8dB, waveform length $N = 2261$, DNN input length $D = 5000$, delay $U = 2700$.

Table 3.1: Phase modulation types, code lengths, and pulse width range for each class in the SIDLE dataset

Class #	**Modulation Type**	**Code Length**	**Pulse Width Ranges (μsec)**
1	Barker	7	$\in [0.875, 7]$
2	Barker	11	$\in [1.375, 11]$
3	Barker	13	$\in [1.625, 13]$
4	Combined Barker	16	$\in [1, 8]$
5	Combined Barker	49	$\in [3.08, 21.1]$
6	Combined Barker	169	$\in [10.58, 84.6]$
7	Max. Length Pseudo Random	15	$\in [1, 4.5]$
8	Max. Length Pseudo Random	31	$\in [0.235, 10.5]$
9	Max. Length Pseudo Random	63	$\in [4.221, 18.9]$
10	Min. Peak Sidelobe	7	$\in [1.05, 4.2]$
11	Min. Peak Sidelobe	25	$\in [1.25, 10]$
12	Min. Peak Sidelobe	48	$\in [2.4, 19.2]$
13	T1	N/A	$\in [2, 16]$
14	T2	N/A	$\in [1.5, 12]$
15	T3	N/A	$\in [1, 8]$
16	Polyphase Barker	7	$\in [0.875, 7]$
17	Polyphase Barker	20	$\in [1, 8]$
18	Polyphase Barker	40	$\in [2, 16]$
19	P1	N/A	$\in [5, 20]$
20	P2	N/A	$\in [3.2, 25.6]$
21	P3	N/A	$\in [3.2, 25.6]$
22	P4	N/A	$\in [5, 20]$
23	Minimum Shift Key	63	$\in [2, 18.9]$

3.4.1 Baseline

As a baseline, we first investigate the performance of the 9-layer real-valued convolutional DNN from [20] on the task of single-pulse classification. As described earlier, this network uses an input dimension of 11000, discards the imaginary part of the input, and noise-pads each training pulse with a fixed noise waveform. To train the network, we used the synchronous noise-padding approach described in Section 3.3

and illustrated in Fig. 3.1, and saw that the network converged to 0% training error. We then tested the network using synchronous noise-padded data from the test set, and observed 3.57% test error, similar to what was reported in [20].

Next, to evaluate how well this DNN performs in the practical asynchronous setting, we re-trained it using the asynchronous noise-padding approach proposed in Section 3.3 and illustrated in Fig. 3.2. When testing with similarly processed (i.e., asynchronous) test data, we observed a test error of 18.29%. This relatively poor performance motivates the design of an improved DNN for asynchronous single-pulse classification.

3.5 Improvements

3.5.1 ResNet

As a first step, we swap the DNN from [20] with a 30-layer ResNet, but still use input dimension 11000 and only the real part of the radar waveform. We configured the ResNet using PyTorch's default parameters, but with a one-dimensional kernel rather than a two-dimensional kernel. Training and testing this ResNet-30 using the asynchronous noise-padding approach from Section 3.3 gave 2.14% test error, which significantly improves upon the 9-layer DNN from [20].

3.5.2 Optimized Input Dimension

To better understand how we might be able to improve the ResNet performance, we plot the classification outcome (i.e., correct or incorrect) versus pre-padded waveform length and SNR in Fig. 3.3. The figure shows that pulses with both short length and low SNR are most likely to be misclassified. This is consistent with the discussion in Section 3.3, which described how noise-padding by factor P causes the SNR

Table 3.2: ResNet-30 vs. Input Length

Input Length	1000	1821	3317	6040	11000
Test Error	8.50%	2.16%	**1.32%**	1.35%	2.14%

to change by the factor $1/P$. Thus, when the network input dimension is very large, short pulses will suffer a large SNR reduction that, when combined with an already low SNR, will likely result in misclassification.

To circumvent this problem, we allow smaller network input dimension D and either truncate or noise-pad each N-length raw waveform as needed (as described in Section 3.3). Table 3.2 reports test error rate for several values of D. The table shows that $D = 3317$ was best among the tested values for the ResNet-30. As a consequence of this D-optimization, the test error improved from 2.14% to 1.32%.

3.5.3 Complex-valued DNN

To improve the DNN further, we incorporate the complex-valued operations from Section 3.2.1 in the ResNet and refer to it as the "$\mathbb{C}$ResNet." We also show results for the real-valued DNN utilizing both in-phase and quadrature radar waveform data from (3.3), which we call the "IQ-ResNet." For a fair comparison, we adjusted the number of channels in the $\mathbb{C}$ResNet-30 and the IQ-ResNet-30 so that the number of trainable parameters is approximately equal to that in the (real-valued) ResNet-30. The resulting test error rates are shown in Table 3.3 for input length 3317, which shows the improvement brought by the $\mathbb{C}$ResNet.

Table 3.3: ResNet-30 vs. ℂResNet-30

Model	Test Error	Trainable Parameters
ResNet-30	1.52%	1782193
IQ-ResNet-30	0.39%	1782417
ℂResNet-30	**0.36%**	1690189

Table 3.4: Fine-tuned ℂResNet Results vs. Network Depth

Model	Test Error	# Parameters	Width	Kernel
ℂResNet-22	0.16%	7721041	32	11
ℂResNet-26	0.16%	1818161	16	7
ℂResNet-30	**0.14%**	659233	8	9
ℂResNet-34	0.15%	670945	8	9
ℂResNet-38	0.16%	2228913	16	7

3.5.4 Network Fine-Tuning

As a final step, we performed an extensive fine-tuning of the ℂResNet. This included a simultaneous search over network depth, network width, kernel width, batch size, and learning rate. For network and kernel width, we adjusted the 1st and 2nd layers respectively, and scaled the remaining layers in the same way as done in the default ResNets in PyTorch. The test error (averaged over 100 random delays and noise realizations) and # of trainable parameters for the optimized network and kernel widths of the fine-tuned ℂResNet are given in Table 3.4. A batch size of 512, learning rate of 0.001, and first-layer kernel width of 11 sufficed for all widths and depths. Each network was trained for 90 epochs, but stopped early if the test loss did not improve for 15 consecutive epochs. In the end, with fine-tuning, the test error rate dropped to 0.14%.

3.6 Classification of multiple overlapping pulses

We now use the fine-tuned network architecture to classify multiple overlapping pulses. For this purpose, we retrained the network to minimize binary cross-entropy (BCE) loss on each of the $K = 17$ outputs, rather than cross-entropy (CE) loss. The resulting network outputs a real-valued vector for which a positive entry in the kth location indicates that a pulse from the kth class is believed to be present, while a negative entry indicates that no pulse is believed to be present.

Our experiments consider $L \in \{1, 2, 3, 4\}$ simultaneously overlapping waveforms. For a given L, multi-pulse waveforms were generated by first creating one asynchronous noise-padded waveform (as described in Section 3.3) and then adding $L - 1$ additional asynchronous noiseless zero-padded waveforms. This way, the SNRs of the multi-pulse waveforms remained between –12 and +12 dB. A single network was trained by sampled L uniformly over $\{1, 2, 3, 4\}$.

There are two common ways to define the error-rate of multi-label classification: "absolute error" (E_{abs}) refers to the error on the binary predictions, whereas "subset error" (E_{sub}) refers to the error on the K-ary prediction vector as a whole. If the binary prediction errors are i.i.d., then $E_{\text{sub}} = 1 - (1 - E_{\text{abs}})^K \approx KE_{\text{abs}}$ using the binomial approximation, which is accurate for small E_{abs}.

Our test error-rate results for fixed $L \in \{1, 2, 3, 4\}$ overlapping pulses (using an 80/20% training/testing split) are presented in Fig. 3.4. There we see that both $\log E_{\text{abs}}$ and $\log E_{\text{sub}}$ grow approximately linearly with $\log L$. We also see that $E_{\text{sub}} \approx KE_{\text{abs}}$. Furthermore, we see that the single-pulse network (from Section 3.5) outperformed the multipulse network in the special case of $L = 1$ test pulses, but this is not surprising because it was trained for this special case. Still, with 4 overlapping pulses, the multi-L network achieves an absolute error of only 4.0%.

3.7 Conclusion

In this chapter we considered the classification of phase-modulated radar waveforms, focusing on the practical asynchronous scenario. We designed a complex-valued ResNet and extensively optimized the network architecture and training procedure. Altogether, our contributions improved the state-of-the-art test error on single-pulse asynchronous SIDLE pulses to 0.14% from the 18.29% achieved with the network designed in [20]. We then modified our complex-ResNet for multi-pulse classification, training the network to classify up to 4 pulses simultaneously and achieved only 4.0% absolute error with 4 overlapping pulses.

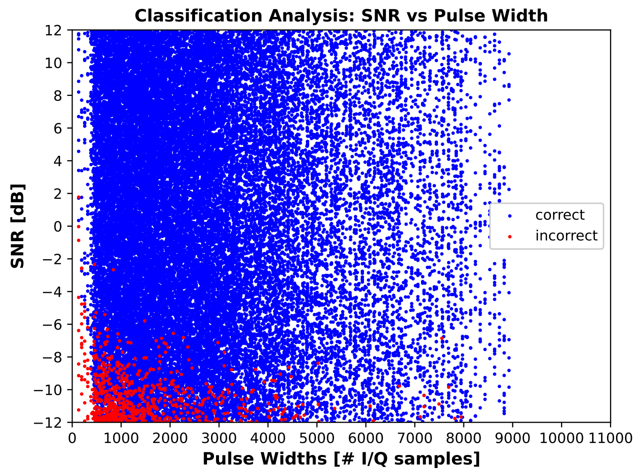

Figure 3.3: Classification outcome (true=blue, false=red) of 11000-input ResNet-30 vs. pre-padded SNR and pulse length.

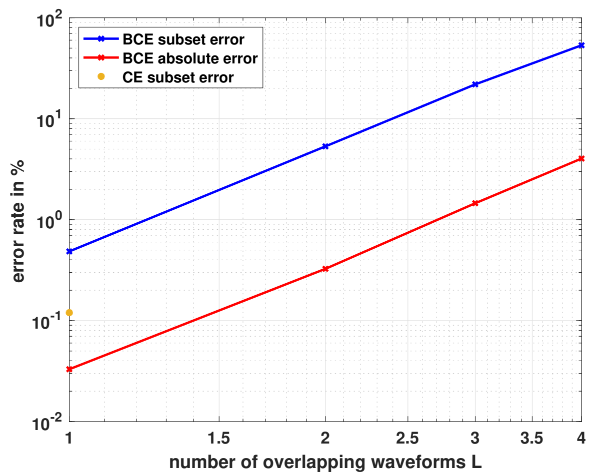

Figure 3.4: Subset and absolute error-rate versus number of simultaneously overlapping waveforms L for the BCE-trained and CE-trained networks.

Chapter 4: Conclusions

4.1 Summary of Original Work

In Chapter 2 of this thesis, we proposed a novel approach to compressive SAR image recovery that used a convolutional neural network (CNN) to de-alias back-projected images estimates. We showed the CNN-based recovery outperformed the standard BP approach in terms of both recovery speed and normalized mean squared error. The CNN-recovered images were also used to train ResNet classifiers that performed on-par or better than a classifier trained on fully sampled SAR images.

In Chapter 3, we considered both single-pulse and multi-pulse classification of phase-modulated radar waveforms from raw signal data, focused on the practical asynchronous setting. We designed a new complex-valued ResNet and data augmentations for this. For single-pulse classification, we showed our design improved the state-of-art error rate on asynchronous SIDLE pulses to 0.14% from the 18.29% of the network designed in [20]. We then modified our single-pulse network for multi-pulse detection of up to 4 pulses, and achieved an absolute error rate of only 4.% with 4 overlapping pulses.

4.2 Potential Future Work

Future experiments of the SAR image recovery and classification approaches developed in Chapter 2 include training one CNN to be agnostic to the degree of subsampling. We could attempt this by training with back-projected scenes of images sampled at different rates. A second possible extension of this work would be to jointly optimize the recovery and classification networks as a single CNN. A third experiment could address that in practice, the imaged target may not belong to one of the classes used to train the CNNs. In this case, we still wish to estimate the image from compressive measurements. We can test the recovery network's performance in this scenario by training a CNN with a random subset of the training classes (e.g., 8 of 10 MSTAR classes). Then we can generate compressive SAR measurements from the classes that were unused during training, input these examples to our trained CNN, and measuring the NMSE of the recovery as done previously. The final step of this experiment is to repeat this procudure several times, randomly leaving out the same number of classes during training time, and averaging the results.

In Chapter 3, our experiments assumed that each pulse was transmitted on similar carrier frequencies. This assumption is unrealistic in a practical application of our CNN, especially in the mulit-pulse case. If there are several transmitters operating simultaneously, then it is likely each will be operating on a unique carrier frequency and often in different frequency bands. A more realistic experiment could be created by assuming knowledge of the carrier frequency of *one* pulse. This pulse could be down-converted to complex-baseband, while the remaining pulses will be located at some random and unknown frequencies. The CNN would now be trained to classify only the complex-baseband pulse, considering the others as interference.

Another approach to multi-pulse classification is to reformulate the CNN to act as a "waveform detection" network, analogous to "object detection" in computer vision literature. Using the raw signal data of radar pulses, one could approach this by attempting to training a deep neural network which predicts the class of each pulse while also "localizing", i.e., predicting the time-of-arrival and time-of-departure each pulse.

Bibliography

[1] D. C. Munson, J. D. O'Brien, and W. K. Jenkins. A tomographic formulation of spotlight-mode synthetic aperture radar. In *Proc. IEEE*, volume 71, pages 917–925, August 1983.

[2] Vishal M. Patel, Glenn R. Easley, Jr. Dennis M. Healy, and Rama Chellappa. Compressed synthetic aperture radar. *IEEE J. Sel. Topics Signal Process.*, 4(2):244–254, April 2010.

[3] Lee C. Potter, Emre Ertin, Jason Parker, and Mujdat Cetin. Sparsity and compressed sensing in radar imaging. *Proc. IEEE*, 98(6):1006–1020, June 2010.

[4] E. J. Candès and M. B. Wakin. An introduction to compressive sampling. *IEEE Signal Process. Mag.*, 25(2):21–30, March 2008.

[5] M. A. Davenport, M. F. Duarte, Y. C. Eldar, and G. Kutyniok. Introduction to compressed sensing. In Y. C. Eldar and G. Kutyniok, editors, *Compressed Sensing: Theory and Applications*. Cambridge Univ. Press, 2012.

[6] Scott S. Chen, David L. Donoho, and Michael A. Saunders. Atomic decomposition by basis pursuit. *SIAM J. Scientific Comput.*, 20:33–61, 1998.

[7] E. van den Berg and M. P. Friedlander. Probing the Pareto frontier for basis pursuit solutions. *SIAM J. Scientific Comput.*, 31(2):890–912, 2008.

[8] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Intl. Conf. Med. Image Comput. & Computer-Assisted Intervention*, pages 234–241, 2015.

[9] Jure Zbontar, Florian Knoll, Anuroop Sriram, Matthew J. Muckley, Mary Bruno, Aaron Defazio, Marc Parente, Krzysztof J. Geras, Joe Katsnelson, Hersh Chandarana, Zizhao Zhang, Michal Drozdzal, Adriana Romero, Michael Rabbat, Pascal Vincent, James Pinkerton, Duo Wang, Nafissa Yakubova, Erich Owens, C. Lawrence Zitnick, Michael P. Recht, Daniel K. Sodickson, and Yvonne W. Lui. fastMRI: An open dataset and benchmarks for accelerated MRI. *arXiv:1811.08839*, 2018.

[10] C. M. Hyun, H. P. Kim, S. M. Lee, S. Lee, and J. K. Sleu. Deep learning for undersampled MRI reconstruction. *Physics in Medicine and Biology*, 63, June 2018.

[11] Z. Zhang, H. Wang, F. Xu, and Y. Q. Jin. Complex-valued convolutional neural network and its application in polarimetric SAR image classification. *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, 55(12):7177–7188, December 2017.

[12] P. Wang, H. Zhang, and V. M. Patel. SAR image despeckling using a convolutional neural network. *IEEE Signal Process. Lett.*, 24(12):1763–1767, May 2017.

[13] D. A. E. Morgan. Deep convolutional neural networks for ATR from SAR imagery. In *Proc. SPIE Defense and Security*, May 2015.

[14] H. Furukawa. Deep learning for target classification from SAR imagery: Data augmentation and translation invariance. *arXiv:1708.07920*, August 2017.

[15] Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz. Loss functions for image restoration with neural networks. *IEEE Trans. Comp. Imag.*, 3(1):47–57, 2016.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv:1512.03385*, 2015.

[17] T. D. Ross, S. W. Worrell, V. J. Velten, J. C. Mossing, and M. L. Bryant. Standard SAR ATR evaluation experiments using the MSTAR public release data set. In *Proc. SPIE*, volume 3370, September 1998.

[18] Brian Rigling and Craig Roush. ACF-Based classification of phase modulated waveforms. In *Proc. IEEE Radar Conf.*, pages 287–291, 2010.

[19] Anne Pavy and Brian Rigling. Phase modulated radar waveform classification using quantile one-class SVMs. In *Proc. IEEE Radar Conf.*, pages 745–750, 2015.

[20] R. V. Chakravarthy, H. Liu, and A. M. Pavy. Open-set radar waveform classification: Comparison of different features and classifiers. In *Proc. IEEE Radar Conf.*, pages 542–547, 2020.

[21] M. Wharton, E. T. Reehorst, and P. Schniter. Compressive SAR image recovery and classification via CNNs. In *Proc. Asilomar Conf. Signals Syst. Comput.*, pages 1081–1085, 2019.

[22] Timothy J O'Shea, Johnathan Corgan, and T Charles Clancy. Convolutional radio modulation recognition networks. In *Prof. Intl. Conf. Eng. Appl. Neural Netw.*, pages 213–226, 2016.

[23] Chiheb Trabelsi, Olexa Bilaniuk, Ying Zhang, Dmitriy Serdyuk, Sandeep Subramanian, João Felipe Santos, Soroush Mehri, Negar Rostamzadeh, Yoshua Bengio, and Christopher J Pal. Deep complex networks. *arXiv:1705.09792*, 2017.

[24] Elizabeth K. Cole, Joseph Y. Cheng, John M. Pauly, and Shreyas S. Vasanawala. Analysis of deep complex-valued convolutional neural networks for MRI reconstruction. *arXiv:2004.01738*, 2020.

[25] E. Stevens, L. Antiga, and T. Viehmann. *Deep Learning with PyTorch*. Manning, Shelter Island, NY, 2020.

[26] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.