FAST COMPRESSED AUTOMATIC TARGET RECOGNITION FOR A COMPRESSIVE INFRARED IMAGER

by

BRIAN MILLIKAN
B.S. University of Central Florida, 1997
M.S. University of Central Florida, 2012

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the Department of Electrical Engineering and Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Spring Term
2018

Major Professor: Hassan Foroosh

ABSTRACT

Many military systems utilize infrared sensors which allow an operator to see targets at night. Several of these are either mid-wave or long-wave high resolution infrared sensors, which are expensive to manufacture. But compressive sensing, which has primarily been demonstrated in medical applications, can be used to minimize the number of measurements needed to represent a high-resolution image. Using these techniques, a relatively low cost mid-wave infrared sensor can be realized which has a high effective resolution. In traditional military infrared sensing applications, like targeting systems, automatic targeting recognition algorithms are employed to locate and identify targets of interest to reduce the burden on the operator. The resolution of the sensor can increase the accuracy and operational range of a targeting system. When using a compressive sensing infrared sensor, traditional decompression techniques can be applied to form a spatial-domain infrared image, but most are iterative and not ideal for real-time environments. A more efficient method is to adapt the target recognition algorithms to operate directly on the compressed samples. In this work, we will present a target recognition algorithm which utilizes a compressed target detection method to identify potential target areas and then a specialized target recognition technique that operates directly on the same compressed samples. We will demonstrate our method on the U.S. Army Night Vision and Electronic Sensors Directorate *ATR Algorithm Development Image Database* which has been made available by the Sensing Information Analysis Center.

To my wife, Jennifer, and kids, Christopher and Lauren

ACKNOWLEDGMENTS

There are many whom have contributed to this work. First and foremost, I would like to thank my advisor Dr. Hassan Foroosh for his guidance through the years and excellent leadership. I would like to thank Dr. Wasfy Mikhael for helping me start my academic experience. I also would like to thank Dr. Marshall Tappen for taking me as a student in the Computer Vision lab at UCF. I learned a great deal under his supervision. I would like to thank Dr. Michael Georgiopoulos for his support through the years. His neural networks course has proven to be very useful! I would like to thank Dr. Robert Muise for his advisement and support. I would like to thank Dr. Abhijit Mahalanobis for mentoring and advising me both in academics and my career. Many thanks to my committee members not previously mentioned, Dr. Nazanin Rahnavard and Dr. George Atia for their guidance and support. I would also like to thank Dr. Qiyu Sun and Dr. Aritra Dutta for their support for my research and collaboration on several papers. I would like to thank my wife and kids for their support for countless hours of research that had to be done on “off hours” at home. Last, but not least, I would like to thank my Lord and Savior Jesus Christ for bringing me to and through this.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ACRONYMS (or) ABBREVIATIONS

ATR – Automatic Target Recognition

BP – Basis Pursuit

CNN – Convolutional Neural Network

CQCF – Compressed Quadratic Correlation Filter

DCT – Discrete Cosine Transform

DFT – Discrete Fourier Transform

DMD – Digital Micromirror Device (a type of spatial light modulator)

DSP – Digital Signal Processing

DWT – Discrete Wavelet Transform

EO/IR – Electro-Optical/Infrared

FLIR – Forward Looking Infrared

FKT – Fukunaga-Koontz Transform

FPA – Focal Plane Array

GB – Gigabytes

GHz – Gigahertz

HOG – Histogram of Oriented Gradients

IIRLS – Initialized Iterative Reweighted Least Squares

IR – Infrared

JPEG – Joint Photographic Experts Group

KLT - Karhunen-Loève Transform

LWIR – Long-wave Infrared

MACH – Maximum Average Correlation Height

MLPNN – Multilayer Perceptron Neural Network

MMCF – Maximum Margin Correlation Filter

MPEG – Moving Pictures Experts Group

MWIR – Mid-wave Infrared

NN – Neural Network

NP-Hard – At least as hard as any NP-problem

NP-Problem – Non-deterministic polynomial time problem

NVESD – Night Vision and Electronic Sensors Directorate

OMP – Orthogonal Matching Pursuit

QCF – Quadratic Correlation Filter

QCFCNN – Quadratic Correlation Filter Convolutional Neural Network

QMLPNN – Quadratic Multilayer Perceptron Neural Network

QMMCF – Quadratic Maximum Margin Correlation Filter

SENSIAC – Sensing Information Analysis Center

SLM – Spatial Light Modulator

SLP – Single-layered Perceptron

STLS – Stochastically Trained Least Squares

SVM – Support Vector Machine

WLS – Weighted Least Squares

CHAPTER 1 - INTRODUCTION

Automatic Target Recognition (ATR) is a general field of study pertaining to the detection and identification of targets of interest in sensor data. These sensors are commonly either forward looking infrared (FLIR) or visible spectrum cameras, but there are others like synthetic aperture radar (SAR) and laser radar [1]. The phrase "ATR" dates back to the early nineteen-eighties with aircraft-mounted targeting pods utilizing a FLIR sensor to detect ground targets of interest [2, 3]. Despite the age of this problem however, ATR is still a critical element of military applications in use today [1, 4].

Many modern ATR electro-optical/infrared (EO/IR) sensors operate in the mid-wavelength infrared (MWIR) or long-wavelength infrared (LWIR) frequency band. MWIR has wavelengths in the range of 3-5 μm and LWIR has wavelengths in the range of 8-12 μm [5]. Both MWIR and LWIR sensors require optics with materials such as germanium (Ge) and sapphire (Al_2O_3) which are costly [5]. In addition to the cost of the optics in MWIR and LWIR sensors, the infrared (IR) focal plane array (FPA) is also constructed from costly materials such as indium gallium arsenide (InGaAs) or germanium (Ge) [6]. Therefore, MWIR and LWIR sensor systems are extremely expensive and cost prohibitive for most applications.

Rice University demonstrated with the single-pixel camera that a cost-effective focal plane can be realized using a single photodetector with a spatial light modulator (SLM) such as the Texas Instruments digital micromirror device (DMD) [7]. This camera uses the principles of compressive sensing to generate a spatial domain image. In compressive sensing, the sparsity in naturally occurring signals in a basis set is used to reduce the sampling requirement to sub-Nyquist levels [8-12]. Using these same principles, an MWIR compressive imaging sensor was developed to reduce the cost of a high resolution FPA by using a lower resolution MWIR FPA and a dense DMD, which is necessary to resolve targets at long ranges without optical assistance [5, 13]. It may seem counterintuitive in compressive sensing to have extended range

by allocating more measurements on the target, but compressive sensing is typically accomplished in a transform domain where the target image has a sparse representation. If we capture enough of the spatial domain signal to reconstruct the sparse representation, then we are still able to resolve targets at long range.

Since a compressive imager does not produce a spatial domain image, a recovery process such as basis pursuit (BP), orthogonal matching pursuit or initialized iterative reweighted least squares (IIRLS) is typically needed before post-processing can be performed [14-16]. This extra processing is time consuming and adds complexity to the system. In addition, many of these methods are iterative which means they have an indefinite convergence time. This is not ideal for a deterministic real-time system which has a set amount of processing time allocated for each new frame of data. However, there are closed-form methods that can be used for recovery which typically require some prior knowledge about the signal being recovered [17, 18].

ATR systems are generally broken up into two stages, detection and recognition, which can be performed as post-processing on a compressive infrared imaging system [1]. Target detection is done to reduce the burden on the more processing intensive recognition stage. Target detection is a two-class discriminator which identifies potential target areas from background clutter. Once the potential target areas are identified, then the target recognition multi-class discriminator can determine the type of target (with the background as a potential target class) [1]. Generally the detected target areas are much larger than the targets under consideration, so the target recognition classifier must also localize where the identified target is in the scene [19, 20].

Quadratic correlation filters (QCFs) have been shown to be an effective tool to identify targets in background clutter for target detection applications [21]. However, extending this to a multi-class discriminator for target recognition is not trivial. One proposed solution is the quadratic maximum margin

correlation filter (QMMCF), which is a QCF paired with a support vector machine (SVM). Each trained SVM represents a target identification class and the response of the SVM is used to determine the identity and the location of the target [20].

Convolutional neural networks (CNNs) have been used recently for deep learning image and video object classification applications [22, 23]. CNNs are classification models that are known to have invariance to some types of transformations on the input signals. Typical image object classifiers have a feature extraction stage, but CNNs learn the feature extraction necessary to classify the data automatically [22]. Since target recognition is analogous to object recognition, this makes CNNs attractive to ATR applications.

As mentioned previously, compressive sensing is used to minimize the cost of a high resolution MWIR sensor but it comes at a higher computational cost. This higher computational cost, if not mitigated, can make this compressive imager prohibitive for real-time applications. In order to maximize performance in a compressive ATR system, an ATR algorithm that processes compressive samples directly bypassing a non-linear, iterative decompression step is proposed. Two methods are proposed, one is a modified quadratic multi-layer perceptron neural network (QMLPNN) and the other is a modified quadratic correlation filter convolutional neural network (QCFCNN). Both are augmented with deconvolution layers designed to process compressed samples. These methods are demonstrated using the SENSIAC *ATR Algorithm Image Database* [24].

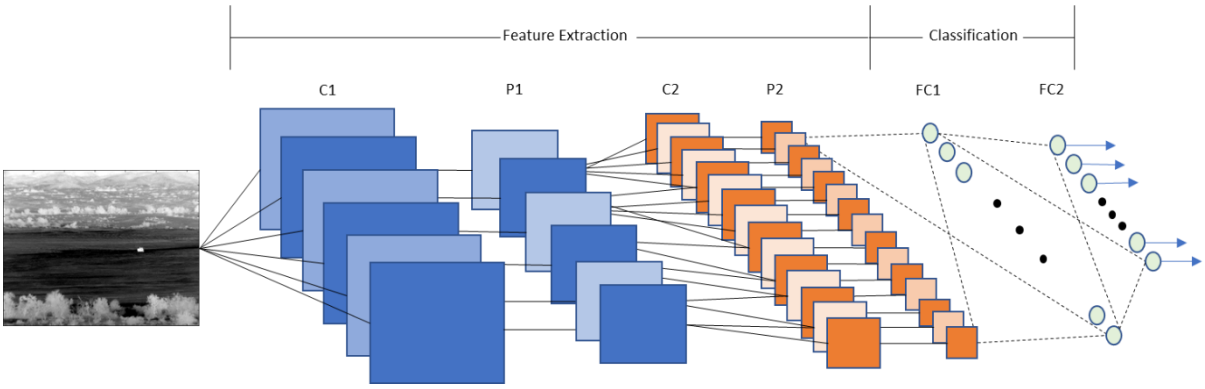


Figure 1 - Sample convolutional neural network. Convolution layers are denoted by C1 and C2 whereas pooling layers are denoted by P1 and P2. These layers can be used for feature extraction. The fully connected layers are denoted by FC1 and FC1. These layers are typically used for classification.

1.1 Background

1.1.1 Convolutional Neural Network

Fully connected neural networks for an image provides a weight connection for each pixel of the input image or feature vector. These weights are then propagated throughout the network to the output layer. A convolutional neural network (CNN) takes advantage of the statistical stationarity of naturally occurring image signals [25]. A CNN typically has convolutional layers and pooling layers followed by fully-connected layers, see Figure 1. Backpropagation of a cost objective function is used to train the network.

A CNN will take an image, $\mathbf{x}_n$ where $n \in [1, N]$ is an index in the set of N training images, and provide an estimated label d_n from the set of classes $\mathcal{C}$. If the input to a convolution layer is $\mathbf{z}_{n,s}$, or a sub-image of the image $\mathbf{z}$, then the output of a convolutional layer is

$$\mathbf{z}_{n,s}^{(\ell+1)} = \sigma\left(\mathbf{H}^{(\ell)}\mathbf{z}_{n,s}^{(\ell)} + \mathbf{b}^{(\ell)}\right), \qquad 1 \le \ell \le L_F - 1 \tag{1}$$

where σ is a non-linear activation function, L_F is layer number for the first fully-connected layer and $\mathbf{z}_n^{(1)} = \mathbf{x}_n$. The filter can be efficiently applied to the entire image through a valid 2D cross-correlation of the form

$$\mathbf{z}_n^{(\ell+1)} = \left(\sigma\left(\sum_{i \in K_{\ell-1}} \mathbf{h}_{i,k}^{(\ell)} \otimes \mathbf{z}_{n,i}^{(\ell)} + b_k^{(\ell)} \right) \right)_{k \in K_\ell}, 1 \leq \ell \leq L_F - 1, \quad (2)$$

where $K_{\ell-1}$ represents the number of input maps and K_ℓ represents the number of output maps that connect the convolutional layers.

CNNs typically have a pooling layer that follows a convolution layer. This layer downsamples the convolution layer by either taking the maximum or average of a local area. Pooling is done to promote translation invariance in a CNN [25].

A CNN also typically consists of fully-connected layers following the convolutional and pooling layers. A fully connected layer is represented by

$$\mathbf{z}_n^{(\ell+1)} = \sigma\left(\mathbf{H}^{(\ell)} \mathbf{z}_n^{(\ell)} + \mathbf{b}^{(\ell)} \right), \quad L_F \leq \ell \leq L - 1, \quad (3)$$

where L_F is the first fully-connected layer in the CNN. The output cost function of a CNN can be any one of a number of functions, but it is commonly a softmax regression function such as

$$J(\mathbf{H}, \mathbf{b}) = -\left[\sum_{n=1}^{N} \sum_{c \in \mathcal{C}} [d_n = c] \log\left(\frac{\exp\left(\langle \mathbf{h}_c^{(L)}, \mathbf{z}_n^{(L)} \rangle + b_c^{(L)} \right)}{\sum_{j \in \mathcal{C}} \exp\left(\langle \mathbf{h}_j^{(L)}, \mathbf{z}_n^{(L)} \rangle + b_j^{(L)} \right)} \right) \right], \quad (4)$$

where $[x = y]$ is Iverson bracket notation for the Kronecker delta function with $\mathbf{H} = \left\{ \mathbf{H}^{(1)}, \mathbf{H}^{(2)}, \ldots, \mathbf{H}^{(L)} \right\}$ and $\mathbf{b} = \left\{ \mathbf{b}^{(1)}, \mathbf{b}^{(2)}, \ldots, \mathbf{b}^{(L)} \right\}$.

Like most neural networks, the CNN can be trained using backpropagation (see Section 3.2.1.1). The output layer error term for the softmax regression function is

$$\boldsymbol{\delta}^{(L+1)} = -\left(\sum_{n=1}^{N} \left([d_n = c] - \mathrm{P}\left[d_n = c \mid \mathbf{z}_n^{(L)}; \mathbf{H}^{(L)}, \mathbf{b}^{(L)} \right] \right) \right)_{c \in \mathcal{C}} \quad (5)$$

(see APPENDIX B). The error function for a fully connected layer is

$$\boldsymbol{\delta}^{(\ell)} = \left(\left(\mathbf{H}^{(\ell)}\right)^T \boldsymbol{\delta}^{(\ell+1)}\right) \circ \sigma'\left(\mathbf{H}^{(\ell)}\mathbf{z}_n^{(\ell)} + \mathbf{b}^{(\ell)}\right), \qquad L_F \leq \ell \leq L \tag{6}$$

where $\circ$ represents the Hadamard product of matrices. The error function for a transition layer between the fully connected layers and convolution and pooling layers is defined by

$$\boldsymbol{\delta}^{(\ell)} = \sigma'\left(\mathbf{H}^{(\ell)}\mathbf{z}^{(\ell)} + \mathbf{b}^{(\ell)}\right) \circ \text{upsample}\left(\boldsymbol{\delta}^{(\ell+1)}\right), \tag{7}$$

Where $\text{upsample}()$ propagates the errors from higher layer to the lower layer through the downsampling layer. Finally, the error function for a convolution layer is

$$\boldsymbol{\delta}^{(\ell)} = \left(\sigma'\left(\sum_{i \in K_{\ell-1}} \mathbf{h}_{\mathrm{i},k}^{(\ell)} \otimes \mathbf{z}_{n,i}^{(\ell)} + b_k^{(\ell)}\right) \circ \text{upsample}\left(\boldsymbol{\delta}_k^{(\ell+1)} \otimes_f \mathbf{h}_k^{(\ell+1)}\right)\right)_{k \in K_\ell}, \tag{8}$$

where $1 \leq \ell \leq L_F - 1$, the operator, $\otimes_f$, represents a "full" 2D cross-correlation between the operands and the $\text{upsample}()$ function typically performs a Kronecker product with a filter of all ones to propagate the error through the pooling layer.

Finally, the gradients are updated using gradient descent. The weight matrix (or tensor) $\mathbf{H}$ with the objective function $J(\mathbf{H}, \mathbf{b})$ can be found using

$$\mathbf{H}_{t+1} = \mathbf{H}_t - \eta \nabla_{\mathbf{H}} J(\mathbf{H}, \mathbf{b}), \tag{9}$$

where $t \geq 0$ and η is the learning rate. Similarly, the bias vector $\mathbf{b}$ can be determined using

$$\mathbf{b}_{t+1} = \mathbf{b}_t - \eta \nabla_{\mathbf{b}} J(\mathbf{H}, \mathbf{b}). \tag{10}$$

The weights for each layer will be calculated in the same manner. The gradients for the fully connected layers for both the weight and the bias are given by

$$\nabla_{\mathbf{H}} J(\mathbf{H}, \mathbf{b}) = \boldsymbol{\delta}^{(\ell+1)}\left(\mathbf{z}_n^{(\ell)}\right)^T \text{ and} \tag{11}$$

$$\nabla_{\mathbf{b}} J(\mathbf{H}, \mathbf{b}) = \boldsymbol{\delta}^{(\ell+1)}, \tag{12}$$

where $L_F \leq \ell \leq L - 1$. The gradients for the convolution layer bias and weight are given by

$$\nabla_{\mathbf{H}} J(\mathbf{H},\mathbf{b}) = \left(\sum_{i \in K_{\ell-1}} \mathbf{z}_{n,i}^{(\ell)} \otimes \boldsymbol{\delta}_k^{(\ell+1)} \right)_{\mathrm{k} \in K_\ell} \quad \text{and} \tag{13}$$

$$\nabla_{\mathbf{b}} J(\mathbf{H},\mathbf{b}) = \left(\sum_{x,y} \left(\boldsymbol{\delta}_k^{\ell+1}\right)_{x,y} \right)_{j \in K_\ell}, \tag{14}$$

where $1 \leq \ell \leq L_F - 1$.

CNN design is an active area of research. Most pattern recognition algorithms, including other neural networks, require some type of feature extraction of the input data. However, an advantage of CNNs is that the convolution layer will learn the necessary features needed to properly classify the dataset [22, 25, 26].

1.1.2 Compressive Sensing

Traditional image sensing is based on standard digital signal processing (DSP) principles. The number of samples from the focal plane array (FPA) equals the number of discretized elements on the FPA. For simplicity, we will assume that the only discretization occurs due to the discretized grid of the FPA and that no quantization of the signal, through digitization, is made. According to the Nyquist-Shannon sampling theorem, the continuous signal can be accurately represented if the sampling rate is two times that of the highest frequency in the signal [27, 28]. However, natural signals have properties that may have an effect on the number of samples needed to accurately represent the continuous signal. Compressive sensors take advantage of these underlying properties of naturally occurring signals to reduce the required number of samples needed to accurately represent the continuous signal thereby incurring a sub-Nyquist rate [8-11, 29].

The sampling of a compressive imager can be modeled through a linear set of equations such as

$$\mathbf{y} = \mathbf{\Phi x} + \mathbf{\eta}, \tag{15}$$

where $\mathbf{y} \in \mathbb{R}^m$ are the measurements, $\mathbf{x} \in \mathbb{R}^N$ is the original signal in discretized form, $\mathbf{\Phi} \in \mathbb{R}^{m \times N}$ is called the measurement or sampling matrix, $\mathbf{\eta}$ represents additive noise and $m < N$. There are an infinite number of solutions to Equation (15) for $\mathbf{x}$ since it is an underdetermined system of equations. So, in compressive sensing, an assumption is made about the discretized signal, $\mathbf{x}$, in that it is sparse or contains many zeros which aides in finding a unique solution. If the original signal, $\mathbf{x}$, is not naturally sparse then is may be representable in another domain such as

$$\mathbf{x} = \mathbf{\Psi \alpha}, \tag{16}$$

where $\mathbf{\alpha}$ is a sparse signal and $\mathbf{\Psi}$ is a sparsifying transformation such as the discrete cosine transform (DCT), discrete wavelet transform (DWT) or discrete Fourier transform (DFT). Using (16) and assuming no additive noise, (15) can be solved using an optimization problem of the form

$$\min \|\mathbf{\alpha}\|_0 \text{ subject to } \mathbf{\Phi \Psi \alpha} = \mathbf{y}, \tag{17}$$

where the ℓ_0-norm is defined as

$$\text{supp}(\mathbf{x}) = \{j \in [1, \dots N] : x_j \neq 0\} \tag{18}$$

$$\|\mathbf{x}\|_0 = |\text{supp}(\mathbf{x})|. \tag{19}$$

The optimization problem (17) is NP-hard meaning that it cannot be solved in polynominal time. So, an ℓ_1-norm relaxation is typically used, which is commonly called basis pursuit (BP) [14]. There are many other proposed methods for finding the solution to (15) including Initialized Iterative Reweighted Least Squares (IIRLS) which is tailored to ATR applications [15], Orthogonal Matching Pursuit [16], Iterative Hard Thresholding [30], and Compressive Sampling Matching Pursuit [31].

Since a compressive imager provides compressed samples, $\mathbf{y}$, rather than the uncompressed image, $\mathbf{x}$, any post-processing must either be tailored to operate on the compressed samples or first use one of the

reconstruction methods to find the uncompressed image estimate, $\hat{\mathbf{x}}$, and then perform processing on the uncompressed estimate. Many of the reconstruction methods are iterative, meaning that they may have error if terminated before convergence. This causes additional processing time on a compressive sensing system that would not be present on a traditional sensing system, which means that compressing sensing systems are typically more computationally expensive [8-12, 29, 32].

1.2 Dataset

We chose the *ATR Algorithm Development Image Database* [24] from the Military Sensing Information Analysis Center (SENSIAC) for our dataset. This dataset contains ten target classes in MWIR video running at 30Hz. There are eight military and two civilian style vehicles, see Figure 2. These videos are 14-bit, 640 x 512 pixel images taken at various ranges, aspect angles and times of day. The ground truth provides the approximate location of the center of the target(s) in each frame where an appropriate-sized window can be formed to select the target based on range. We chose to use a 40 x 80 sized window for the targets and consider only targets within 1500 meters at any time of day.

Several of the videos contain a target that is driving in a 100 meter diameter circle at about 10 miles/hr. Because of this, many of the targets exhibit a full 360 degrees of rotation. The videos are 60 seconds at 30 Hz or 1800 frames long. Most scenarios contain a single target, but there are a few scenarios that contain multiple targets.

(a) 2S3 (b) BMP2 (c) BRDM2 (d) BTR70

(e) D20 (f) MTLB (g) T72 (h) ZSU23-4

(i) Pickup (j) SUV

Figure 2 - Target data types from the SENSIAC *ATR Algorithm Development Image Database* [24]. (a) Self-propelled howitzer, (b) Armoured personnel carrier, (c) Infantry scout vehicle, (d) Armoured personnel carrer, (e) Towed howitzer, (f) Armoured reconnaissance vehicle, (g) Main battle tank, (h) Anti-aircraft weapon, (i) Civilian pickup truck, and (j) Civilian sport uility vehicle.

1.3 Notation

Throughout this dissertation, we shall use the following notation:

- A bold lowercase letter represents a vector or vectorized ordering of a matrix, e.g. $\mathbf{x}$.
- A non-bold, italicized lowercase letter represents a scalar value, e.g. x.
- A subscripted, non-bold, italicized lowercase letter represents an element of a vector or matrix, e.g. x_k, m_{ij}.
- The symbol, $\circ$, represents the Hadamard operator.
- The symbol, $\otimes$, represents a valid 2D cross-correlation operation, e.g.

$$y_{jk} = \sum_{m=1}^{J-M+1} \sum_{n=1}^{K-N+1} h_{mn} x_{m+j,n+k} \text{ where } \mathbf{h} \in \mathbb{R}^{M \times N}, \mathbf{x} \in \mathbb{R}^{J \times K}. \quad (20)$$

- The symbol, $\bigotimes_f$, represents a full 2D cross-correlation operation, e.g.

$$y_{jk} = \sum_{m=1}^{J+M-1} \sum_{n=1}^{K+N-1} h_{mn} x_{m+j,n+k} \text{ where } \mathbf{h} \in \mathbb{R}^{M \times N}, \mathbf{x} \in \mathbb{R}^{J \times K}. \quad (21)$$

- The superscript, T, represents the transpose of a matrix or vector, e.g. $\mathbf{x}^T$.
- The use of brackets will be used to represent a set, e.g. $A \coloneqq \{1,2,3\}$
- The use of the parallel line operator, when used with a set, will be used to represent the cardinality of the set, e.g. $|A| = 3$. Otherwise, it will represent the absolute value.
- The norm operator of a vector will be defined as

$$\|\mathbf{x}\|_P \coloneqq \left(\sum_{j=1}^{N} |x_j|^P \right)^{1/P} \quad (22)$$

- For neural network definitions a superscript, such as ℓ, represents a network layer, e.g. $\mathbf{W}^{(\ell)}$. This would represent the weight matrix, $\mathbf{W}$, for the ℓ^{th} network layer.

1.4 Overview

Using the MWIR sensor proposed in [13], we will present compressed target detection methods that work directly with compressed sensor measurements in CHAPTER 2. We will also present a sampling matrix that will be applied to the DMD when acquiring samples. Our goal is to provide a method to identify and localize targets from the compressed samples. To do this, we will first present a maximum-margin correlation filter that can be used to identify and localize targets. Then, we will present a quadratic multi-layer perceptron which, we will show, enhances the discrimination capability of a multi-layer perceptron for automatic target recognition applications. Finally, we will present a CNN which has a QCF as an input

layer in CHAPTER 3 to take advantage of the discrimination properties of the QCF for target recognition. This allows us to use the multi-class classification and added localization properties of the CNN as an alternate method to the maximum-margin correlation filter. Then, without resampling, in CHAPTER 4 we will present target recognition methods that identify the target type from the compressed measurements. Also in CHAPTER 4, we will include experiments demonstrating our proposed methods. Finally, we will summarize our findings in CHAPTER 5.

CHAPTER 2 - COMPRESSED TARGET DETECTION

Target detection is used to reduce the processing requirement of the usually more intensive target recognition algorithm. Some popular approaches to target detection include anomaly detection and correlation. Anomaly detection tries to detect an anomalous target in a background setting whereas correlation uses target descriptions that are converted to templates which can be then be used to find matches in new data [4]. One particular correlation filter method is the quadratic correlation filter (QCF). With a correlation filter, we generate a classification for each pixel of the test image [21]. However, for a compressed image, we are only interested in the classification of the region that is compressed. Because of this, we only need to apply the filter once to the compressed region to generate a classification for the whole region.

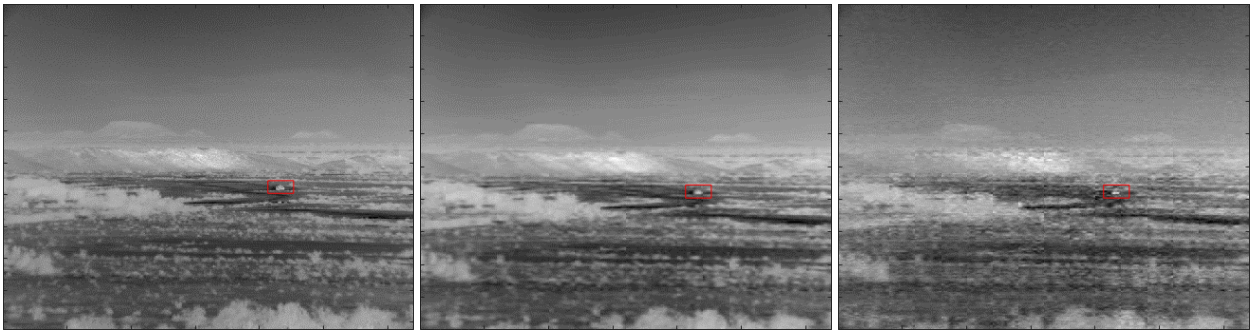
In order to use a compressed target detection method, we will require a special sampling matrix that will be applied to the DMD to capture the measurements. Most methods used to solve (17) are iterative and require several iterations for a reasonable solution. As mentioned, the ℓ_1 relaxation of this equation is a popular choice and is called basis pursuit. However, since perfect reconstruction is not our objective, we can relax this requirement and use ℓ_2-based reconstruction methods which have closed-form solutions, as long as target detection performance is not impacted [18].

In this chapter, we will review a couple of ℓ_2-based reconstruction methods. The first method is based on a weighted object-specific least-squares reconstruction method [33]. The second method is based on the probability distribution of the spectral coefficients from the training data [18]. Specifically, this method uses the distributions of the training data to determine the probability of having a large coefficient. This method is called stochastically trained least squares (STLS). Following this, we will explore how ℓ_2-based recovery methods can be paired with a target detection method, like QCF, to create

a compressed target detection method. Alternatively, we can eliminate the reconstruction ℓ_2-based recovery altogether by using a special sampling matrix for a faster solution [18].

2.1 ℓ_2 Reconstruction Techniques

It has been shown that iterative methods are significantly more time-consuming than closed-form solutions and will provide a non-optimal solution if terminated before convergence [18]. With a linear closed-form solution, optimized matrix algebra libraries and built-in parallel processing hardware can be employed to significantly accelerate execution. Furthermore, we find that target detection performance is not notably impacted by compression using an ℓ_2 reconstruction method [18]. In this section, we will recall the weighted object-specific least-squares reconstruction method presented in [33]. We will also review an ℓ_2-based recovery method presented in [18] called STLS which is based on the probability distributions of the training data spectral components.



(a) (b) (c)

Figure 3 - ℓ_2-based reconstruction examples: (a) original image with a single Anti-Aircraft Weapon (ZSU23-4) target, (b) STLS reconstructed image and (c) weighted ℓ_2-based reconstructed image from 50 samples per 20×40 block. The measurements for (b) are captured using the STLS sensing matrix in equation (53) and the measurements for (c) are captured using a random Bernoulli matrix.

2.1.1 Weighted Object-Specific Least-Squares Reconstruction

In target detection, infrared image patches, $\mathbf{x}$, have a compressible representation,

$$\mathbf{x} = \mathbf{\Psi\alpha} = \sum_{i=1}^{N} \alpha_i \boldsymbol{\psi}_i, \tag{23}$$

in some (DCT/DWT/KLT) basis $\mathbf{\Psi} = [\boldsymbol{\psi}_1 \quad \boldsymbol{\psi}_2 \quad \cdots \quad \boldsymbol{\psi}_N]$, where $\mathbf{\alpha} = [\alpha_1 \quad \alpha_2 \quad \cdots \quad \alpha_N]^{\boldsymbol{T}}$, is the sparse spectral representation of the infrared image patch in this basis. The measurement vector $\mathbf{y}$ of the infrared image patch $\mathbf{x}$ is

$$\mathbf{y} = \mathbf{\Phi x} = \hat{\mathbf{\Phi}}\mathbf{\alpha}, \tag{24}$$

where $\hat{\mathbf{\Phi}} = \mathbf{\Phi\Psi}$ and $\mathbf{\Phi}$ is an $m \times N$ random sensing matrix with $m < N$.

Since this is an underdetermined system, it requires additional information from the system to find a solution. The solution in (17) uses the sparsity in naturally occurring signals to find a unique solution. Although (17) provides an ideal method for finding the sparsity in a naturally occurring signal, it is categorized as an NP-Hard program since a large search space needs to be created to find the solution [8-12, 29]. As mentioned, a typical relaxation of (17) uses the $\ell_1$norm

$$\min\|\mathbf{\alpha}\|_1 \text{ subject to } \mathbf{\Phi\Psi\alpha} = \mathbf{y}, \tag{25}$$

and it is typically called basis pursuit which can be solved using linear programming [8-12, 14, 29].

However, the ℓ_2-norm optimization problem takes the form

$$\min\|\mathbf{\alpha}\|_2^2 \text{ subject to } \mathbf{\Phi\Psi\alpha} = \mathbf{y}, \tag{26}$$

where we can solve for $\mathbf{\alpha}$ using LaGrange multipliers. Using LaGrange multipliers $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \ldots, \lambda_m]^T$, we can rewrite the objective problem as

$$\mathrm{J}(\mathbf{\alpha}) = \mathbf{\alpha}^T\mathbf{\alpha} + \boldsymbol{\lambda}^T(\mathbf{\Phi\Psi\alpha} - \mathbf{y}), \tag{27}$$

perform the minimization by taking the derivative,

$$\frac{\partial \mathrm{J}(\mathbf{\alpha})}{\partial \mathbf{\alpha}} = 2\mathbf{\alpha} + \boldsymbol{\lambda}^T\mathbf{\Phi\Psi} = \mathbf{0}, \tag{28}$$

and then solve for $\boldsymbol{\alpha}$,

$$\hat{\boldsymbol{\alpha}} = -\frac{1}{2}\boldsymbol{\Psi}^T\boldsymbol{\Phi}^T\boldsymbol{\lambda}. \tag{29}$$

Using this solution, we can substitute it back into the original constraint equation in (26)

$$\boldsymbol{\Phi}\boldsymbol{\Psi}\,\hat{\boldsymbol{\alpha}} = -\frac{1}{2}\boldsymbol{\Phi}\boldsymbol{\Psi}\boldsymbol{\Psi}^T\boldsymbol{\Phi}^T\boldsymbol{\lambda} = \mathbf{y}. \tag{30}$$

If we assume that the transformation matrix $\boldsymbol{\Psi}$ is orthonormal or unitary, then $\boldsymbol{\Psi}\boldsymbol{\Psi}^{\boldsymbol{T}} = \mathbf{I}$ where $\mathbf{I}$ represents the identity matrix and we can solve for the LaGrange multipliers $\boldsymbol{\lambda}$

$$\boldsymbol{\lambda} = -2(\boldsymbol{\Phi}\boldsymbol{\Phi}^T)^{-1}\mathbf{y}. \tag{31}$$

We can now find the optimal $\hat{\boldsymbol{\alpha}}$ by substituting (31) back into (29)

$$\hat{\boldsymbol{\alpha}} = \boldsymbol{\Psi}^T\boldsymbol{\Phi}^T(\boldsymbol{\Phi}\boldsymbol{\Phi}^T)^{-1}\mathbf{y} \tag{32}$$

and to convert it back into the original space, we can substitute (32) into (16)

$$\hat{\mathbf{x}} = \boldsymbol{\Psi}\boldsymbol{\Psi}^T\boldsymbol{\Phi}^T(\boldsymbol{\Phi}\boldsymbol{\Phi}^T)^{-1}\mathbf{y} = \boldsymbol{\Phi}^T(\boldsymbol{\Phi}\boldsymbol{\Phi}^T)^{-1}\mathbf{y} \tag{33}$$

given that $\boldsymbol{\Psi}$ is orthonormal or unitary. This solution is known as the least squares solution [34]. We note that (33) does not depend on the transformation matrix $\boldsymbol{\Psi}$, but only on the random sensing matrix $\boldsymbol{\Phi}$ if $\boldsymbol{\Psi}$ is orthonormal or unitary and does not find the optimal sparse solution. Hence, it is not an effective reconstruction technique.

To force the spatial domain representation to be dependent on the transformation matrix $\boldsymbol{\Psi}$ for the ℓ_2 reconstruction, Mahalanobis et. al. [33] uses a weighted minimization problem to reconstruct to compressed data. The new optimization problem is

$$\min \boldsymbol{\alpha}^T\mathbf{W}\boldsymbol{\alpha} \text{ subject to } \boldsymbol{\Phi}\boldsymbol{\Psi}\boldsymbol{\alpha} = \mathbf{y}, \tag{34}$$

which can be rewritten using LaGrange multipliers

$$\mathrm{J}(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^T\mathbf{W}\boldsymbol{\alpha} + \boldsymbol{\lambda}^T(\boldsymbol{\Phi}\boldsymbol{\Psi}\boldsymbol{\alpha} - \mathbf{y}), \tag{35}$$

where we can again take the derivative with respect to $\boldsymbol{\alpha}$ to find the minimum

$$\frac{\partial \mathrm{J}(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} = 2\mathbf{W}\boldsymbol{\alpha} + \boldsymbol{\lambda}^T\boldsymbol{\Phi}\boldsymbol{\Psi} = \mathbf{0}. \tag{36}$$

We can now find the optimal $\hat{\boldsymbol{\alpha}}$

$$\hat{\boldsymbol{\alpha}} = -\frac{1}{2}\mathbf{W}^{-1}\boldsymbol{\Psi}^T\boldsymbol{\Phi}^T\boldsymbol{\lambda} \tag{37}$$

using an invertible diagonal matrix $\mathbf{W}$ and substitute this back into the original constraint equation (34)

$$\boldsymbol{\Phi}\boldsymbol{\Psi}\,\hat{\boldsymbol{\alpha}} = -\frac{1}{2}\boldsymbol{\Phi}\boldsymbol{\Psi}\mathbf{W}^{-1}\boldsymbol{\Psi}^T\boldsymbol{\Phi}^T\boldsymbol{\lambda} = \mathbf{y} \tag{38}$$

to find the LaGrange multipliers

$$\boldsymbol{\lambda} = -2(\boldsymbol{\Phi}\boldsymbol{\Psi}\mathbf{W}^{-1}\boldsymbol{\Psi}^T\boldsymbol{\Phi}^T)^{-1}\mathbf{y}. \tag{39}$$

To find the optimal $\hat{\boldsymbol{\alpha}}$, we substitute (39) into (37)

$$\hat{\boldsymbol{\alpha}} = \mathbf{W}^{-1}\boldsymbol{\Psi}^T\boldsymbol{\Phi}^T(\boldsymbol{\Phi}\boldsymbol{\Psi}\mathbf{W}^{-1}\boldsymbol{\Psi}^T\boldsymbol{\Phi}^T)^{-1}\mathbf{y} \tag{40}$$

and convert back to the spatial domain

$$\hat{\mathbf{x}} = \boldsymbol{\Psi}\mathbf{W}^{-1}\boldsymbol{\Psi}^T\boldsymbol{\Phi}^T(\boldsymbol{\Phi}\boldsymbol{\Psi}\mathbf{W}^{-1}\boldsymbol{\Psi}^T\boldsymbol{\Phi}^T)^{-1}\mathbf{y}. \tag{41}$$

This spatial domain recovery equation not only has a dependency on the new weight matrix $\mathbf{W}$, but it also preserves the dependency on the sparsifying transform $\boldsymbol{\Psi}$.

The weight matrix is defined as

$$\mathbf{W} = \begin{pmatrix} w_1^2 & 0 & 0 & \cdots & 0 \\ 0 & w_2^2 & 0 & \cdots & 0 \\ 0 & 0 & w_3^2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & w_N^2 \end{pmatrix} \tag{42}$$

with entries based on the training data. Most methods for finding the diagonal entries in $\mathbf{W}$ utilize the autocorrelation matrix $\mathbf{R}_\alpha = E[\boldsymbol{\alpha}\boldsymbol{\alpha}^T]$ of the spectral components to describe the desired object. The expected value is over the class of the target we wish to detect.

An intuitive approach to finding the weights would be to emphasize large coefficients in $\boldsymbol{\alpha}$ and suppress small (or zero) coefficients. To do this, Mahalanobis et. al. [33] proposes using

$$J(\mathbf{w}) = \frac{\mathbf{w}^T\mathbf{R}_\alpha\mathbf{w}}{\mathbf{w}^T\mathbf{D}\mathbf{w}} \tag{43}$$

where $\mathbf{D}$ is a diagonal matrix equal to the diagonal of $\mathbf{R}_\alpha$. To maximize this quantity, we take the derivative with respect to $\mathbf{w}$ and set it equal to zero

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \frac{[\mathbf{w}^T\mathbf{D}\mathbf{w}]\frac{\partial[\mathbf{w}^T\mathbf{R}_\alpha\mathbf{w}]}{\partial\mathbf{w}} - [\mathbf{w}^T\mathbf{R}_\alpha\mathbf{w}]\frac{\partial[\mathbf{w}^T\mathbf{D}\mathbf{w}]}{\partial\mathbf{w}}}{(\mathbf{w}^T\mathbf{D}\mathbf{w})^2} = 0. \tag{44}$$

Since we are interested when this quantity is equal to zero, we only need to consider the numerator

$$[\mathbf{w}^T\mathbf{D}\mathbf{w}]2\mathbf{R}_\alpha\mathbf{w} - [\mathbf{w}^T\mathbf{R}_\alpha\mathbf{w}]2\mathbf{D}\mathbf{w} = 0. \tag{45}$$

We can rearrange this equation and collect terms

$$\mathbf{R}_\alpha\mathbf{w} = \frac{\mathbf{w}^T\mathbf{R}_\alpha\mathbf{w}}{\mathbf{w}^T\mathbf{D}\mathbf{w}}\mathbf{D}\mathbf{w} \tag{46}$$

which can be rewritten as

$$\mathbf{R}_\alpha\mathbf{w} = J(\mathbf{w})\mathbf{D}\mathbf{w}. \tag{47}$$

In this equation, $J(\mathbf{w})$ is a scalar value, so we can multiply both sides of the equation by $\mathbf{w}^T\mathbf{D}^{-1}$ given that $\mathbf{D}$ is invertible and $\mathbf{w}$ is orthonormal. The resulting equation is

$$J(\mathbf{w}) = \mathbf{w}^T\mathbf{D}^{-1}\mathbf{R}_\alpha\mathbf{w}. \tag{48}$$

It can be seen from this equation that this is the eigendecomposition of the matrix $\mathbf{D}^{-1}\mathbf{R}_\alpha$ with $\mathbf{w}$ as the eigenvector and $J(\mathbf{w})$ as the eigenvalue. So, in order for this quantity to be maximized, we will choose

the eigenvector that corresponds to the largest eigenvalue of $\mathbf{D}^{-1}\mathbf{R}_{\alpha}$. Another choice for the weights involves solving

$$\mathbf{R}_{\alpha} = \boldsymbol{\Psi}\mathbf{W}^{-1}\boldsymbol{\Psi}^{T} \tag{49}$$

where $\mathbf{R}_{\alpha}$ is the autocorrelation matrix and $\boldsymbol{\Psi}$ contains the eigenvectors of $\mathbf{R}_{\alpha}$. Given this, the diagonal matrix $\mathbf{W}$ can be determined using the eigenvalues λ_i

$$\lambda_i = \frac{1}{w_i^2} \tag{50}$$

or

$$w_i = \pm\lambda_i^{-1/2}. \tag{51}$$

The quantity will be maximized by allowing $\mathbf{w}$ to be the eigenvector for the largest eigenvalue of $\mathbf{D}^{-1}\mathbf{R}_{\alpha}$. An example of an image that has been reconstructed using weighted least squares is shown in Figure 3c.

2.1.2 Stochastically Trained Least Squares

Since the image $\mathbf{x}$ is compressible, it can be approximated by a sparse image $\hat{\mathbf{x}} = \boldsymbol{\Psi}\boldsymbol{\alpha}^{*}$ with small $\|\boldsymbol{\alpha}\|_0$. The support of the recovered signal $\hat{\mathbf{x}}$ is denoted by the set S which has cardinality s. In most applications like target detection, the true support set S is unknown. But, the training data can be used to identify the most probable locations T of the largest coefficients. In [18], this was defined as

$$T = \{i \in [1, N] \mid P[|\alpha_i| \geq \rho] \geq \kappa\}, \tag{52}$$

where P is the coefficient probability distribution, ρ is the coefficient discrimination threshold and κ is the probability threshold for the set T.

In [18], we assumed that formation of the measurement matrix $\boldsymbol{\Phi}$ depends on the statistical properties of the signal $\mathbf{x}$ such that

$$\mathbf{\Phi} = \mathbf{\Psi}_T, \tag{53}$$

where $\mathbf{\Psi}$ is some basis (DCT/DWT/KLT) matrix and $\mathbf{\Psi}_T$ is created by selecting the rows in $\mathbf{\Psi}$ specified by T. This matrix is used to capture the measurements $\mathbf{y}$. We note that most sensing matrices for compressive imaging are random, thereby sampling the signal in the spatial domain. This has been shown [8-10, 29] to be optimal for recovering a signal from a sparsifying domain. However in the case of STLS, we find that a fixed deterministic sensing matrix is sufficient for capturing dominant spectral coefficients and recover essential data for target detection and recognition applications. This sampling matrix is put on the DMD providing samples that are in the transformed domain.

Since we have some statistical information about the support set S of the target image, the probability of the set S with cardinality s to be contained in the set T with cardinality $t \geq s$ is high. Therefore, we can use the least squares solution $\boldsymbol{\alpha}_T$ of the linear system

$$\mathbf{y} = \widehat{\mathbf{\Phi}}_T \boldsymbol{\alpha}_T, \tag{54}$$

to approximate the compressible coefficient vector $\boldsymbol{\alpha}^*$. The matrix $\widehat{\mathbf{\Phi}}_T$ is a submatrix of $\widehat{\mathbf{\Phi}}$ and is formed by selecting the columns of $\widehat{\mathbf{\Phi}}$ in T**.** Using STLS, equation (54) has the solution

$$\boldsymbol{\alpha}_T = \widehat{\mathbf{\Phi}}_T^{+} \mathbf{y} \tag{55}$$

where $\widehat{\mathbf{\Phi}}_T^{+}$ is the Moore-Penrose pseudoinverse of the matrix $\widehat{\mathbf{\Phi}}_T$.

In [18], by using DCT coefficients α_i can be assumed to be independent with a Laplace distribution, cf. [35] and have a corresponding population mean and variance,

$$\alpha_i \sim \text{LaPlace}(\mu_i, b_i) \text{ for } i = 1, \dots, N, \tag{56}$$

which can be approximated by the sample mean and variance. The i-th sample mean $(\bar{\alpha}_i)$ and sample variance $\left(2\tilde{b}_i^2\right)$ are evaluated by

$$\bar{\alpha}_i = \frac{1}{N_S}\sum_{j=1}^{N_S}\alpha_i(j) \tag{57}$$

and

$$2\tilde{b}_i^2 = \frac{1}{N_S - 1}\sum_{j=1}^{N_S}(\alpha_i(j) - \bar{\alpha}_i)^2, \;\; 1 \le i \le N_S, \tag{58}$$

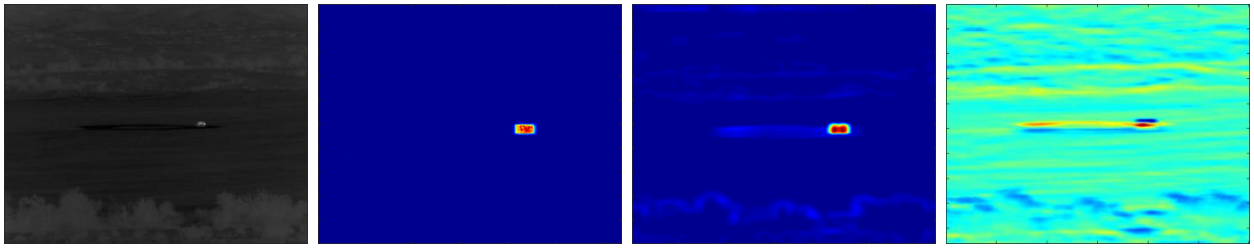
where N_S is the total number of target training images. Therefore, the probability of having a large coefficient at the i-th position can be calculated as

$$P[|\alpha_i \ge \rho|] = 1 - \frac{1}{2\tilde{b}_i^2}\int_{-\rho}^{\rho} e^{-\frac{|\alpha_i - \bar{\alpha}_i|}{\tilde{b}_i}}\,\mathrm{d}\alpha_i, \tag{59}$$

with the measurement matrix defined as $\mathbf{\Phi} \in \mathbb{R}^{t\times N}$ as in (53) for STLS. An example of an image that has been reconstructed using STLS is shown in Figure 3b.

2.2 The Quadratic Correlation Filter

The quadratic correlation filter is a fast and efficient way to find targets in a cluttered scene. The original quadratic correlation filter (QCF), introduced by Mahalanobis et. al. [21] utilized the Fukunaga-Koontz Transform (FKT) to separate target image areas from background clutter [36]. In their same paper, an alternative filter creation method was also presented which finds the filter that maximizes the separation between the expected values of the target and background statistics. In [18], another filter was introduced that minimizes the sum of the squared error between the output statistic and the true label. We will review these methods in this section.



(a) (b) (c) (d)

Figure 4 – Target detection using QCF with three different filter selection methods : (a) An MWIR image containing a single Sport Utility Vehicle (SUV) target type. Three different methods of generating the coefficient matrix for the quadratic correlation filter are applied to this image: (b) the Fukunaga-Koontz Transform method, (c) the max distance between expected values method and (d) the single-layer perceptron method.

2.2.1 Fukunaga-Koontz Transform Method

In Mahalanobis et. al. [21], the first filter introduced was derived from the Fukunaga-Koontz Transform (FKT). In this method, we denote a target pattern as $\mathbf{x}_{tgt}$ and a background pattern as $\mathbf{x}_{bkg}$. These patterns have correlation matrices

$$\mathbf{R}_{tgt} = \mathbb{E}\left[\mathbf{x}_{tgt}\mathbf{x}_{tgt}^T\right] \text{ and } \mathbf{R}_{bkg} = \mathbb{E}\left[\mathbf{x}_{bkg}\mathbf{x}_{bkg}^T\right], \tag{60}$$

respectively. These correlation matrices are positive semidefinite. The sum of the target and background correlation matrices can be decomposed into the form

$$\mathbf{R}_{tgt} + \mathbf{R}_{bkg} = \mathbf{V}\boldsymbol{\Delta}\mathbf{V}^T, \tag{61}$$

where the columns of the orthogonal matrix $\mathbf{V}$ are the eigenvectors and the diagonal matrix $\boldsymbol{\Delta}$ contains the corresponding eigenvalues of $\left(\mathbf{R}_{tgt} + \mathbf{R}_{bkg}\right)$. We can then define a transform matrix

$$\mathbf{P} = \mathbf{V}\boldsymbol{\Delta}^{-1/2} \tag{62}$$

and rewrite (61) as

$$\mathbf{P}^T\left(\mathbf{R}_{tgt} + \mathbf{R}_{bkg}\right)\mathbf{P} = \mathbf{I}, \tag{63}$$

where $\mathbf{I}$ is the identity matrix.

Using the transform matrix, we can define target and background image patch correlation matrices

$$\widehat{\mathbf{R}}_{tgt} = \mathbf{P}^T\mathbf{R}_{tgt}\mathbf{P} \text{ and } \widehat{\mathbf{R}}_{bkg} = \mathbf{P}^T\mathbf{R}_{bkg}\mathbf{P} \tag{64}$$

in the new domain, which reduces equation (63) to

$$\widehat{\mathbf{R}}_{tgt} + \widehat{\mathbf{R}}_{bkg} = \mathbf{I}. \tag{65}$$

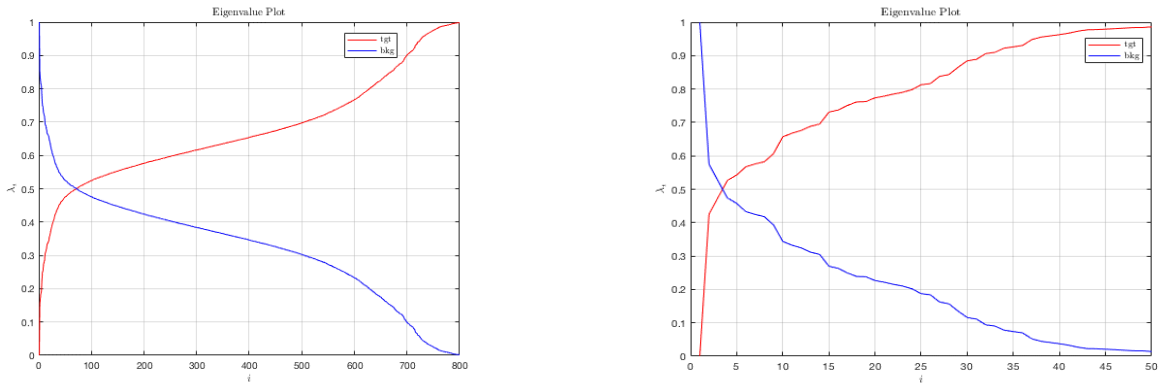
We can then perform another decomposition on the new domain correlation matrices which gives

$$\widehat{\mathbf{R}}_{tgt} = \mathbf{M}\mathbf{\Lambda}\mathbf{M}^T, \tag{66}$$

where the columns of the orthogonal matrix $\mathbf{M}$ are the eigenvectors and the diagonal matrix $\mathbf{\Lambda}$ contains the corresponding eigenvalues of $\widehat{\mathbf{R}}_{tgt}$. Using (65) and (66), we obtain

$$\widehat{\mathbf{R}}_{bkg} = \mathbf{M}(\mathbf{I} - \mathbf{\Lambda})\mathbf{M}^T, \tag{67}$$

where the eigenvalues of $\widehat{\mathbf{R}}_{tgt}$ and $\widehat{\mathbf{R}}_{bkg}$ lie between zero and one as demonstrated in Figure 5.



(a) (b)

Figure 5 - Eigenvalue plots : (a) uncompressed QCF and (b) compressed QCF, with $t = 50$. As mentioned in [21], for the QCF and CQCF to be effective in target detection, we need to choose a proper threshold ϵ so that the eigenvalues associated with the targets and background are sufficiently separated. We observe that, due to the compression, CQCF has less eigenvalues and the resulting discrimination statistic will be smaller in magnitude than QCF. These plots were generated using the sample correlation matrices calculated from the training set.

Given a threshold $\epsilon \in (0,1)$, we choose the largest $|\Omega_{tgt}|$ eigenvalues λ_i in $\mathbf{\Lambda}$ for the target (positive) class, where

$$\Omega_{tgt} \coloneqq \{i \in [1, N] \mid \lambda_i \geq 1 - \epsilon\}. \tag{68}$$

Similarly, we chose the $|\Omega_{bkg}|$ eigenvalues λ_i in $\mathbf{\Lambda}$ for the background (negative) class, where

$$\Omega_{bkg} \coloneqq \{i \in [1, N] \mid \lambda_i \leq \epsilon\}. \tag{69}$$

Using the indices of the set Ω_{tgt} to select the columns of the matrix $\mathbf{M}$, we create a target projection submatrix $\mathbf{M}_{\Omega_{tgt}}$. We can also create the background projection submatrix $\mathbf{M}_{\Omega_{bkg}}$ in a similar fashion using the set Ω_{bkg}.

Given a test image patch, $\mathbf{x}$, we can classify it as a target or background by projecting it into

$$\mathbf{m}_{tgt} = \mathbf{M}_{\Omega_{tgt}}^T \mathbf{P}^T \mathbf{x} \tag{70}$$

and

$$\mathbf{m}_{bkg} = \mathbf{M}_{\Omega_{bkg}}^T \mathbf{P}^T \mathbf{x} \tag{71}$$

We can then define a statistic

$$\varphi = \mathbf{m}_{tgt}^T \mathbf{m}_{tgt} - \mathbf{m}_{bkg}^T \mathbf{m}_{bkg}, \tag{72}$$

which can also be written as

$$\varphi(\mathbf{x}) = \mathbf{x}^T \mathbf{H} \mathbf{x}, \tag{73}$$

where $\mathbf{H} = \mathbf{F}\mathbf{F}^T - \mathbf{G}\mathbf{G}^T$, $\mathbf{F} = \mathbf{P}\mathbf{M}_{\Omega_{tgt}}$, $\mathbf{G} = \mathbf{P}\mathbf{M}_{\Omega_{bkg}}$. This statistic is used to discriminate between target and background areas in an image. The values for φ will be large and positive for target areas and small or negative for background areas, see Figure 4b.

An efficient way of applying the filter, $\mathbf{H}$, from (73) to a full image is by using a correlation. If we allow $\mathbf{f}_i$ to represent a column vector of $\mathbf{F}$ rearranged into an $r \times c$ mask and similarly $\mathbf{g}_i$ to represent a column vector of $\mathbf{G}$, then we can compute (73) using

$$\boldsymbol{\varphi} = \sum_{i=1}^{|\Omega_{tgt}|} |\mathbf{x} \otimes \mathbf{f}_i|^2 - \sum_{i=1}^{|\Omega_{bkg}|} |\mathbf{x} \otimes \mathbf{g}_i|^2 \tag{74}$$

for every element in the full-sized test image $\mathbf{x}$. The peaks in the resulting map $\boldsymbol{\varphi}$ from (74) represent potential target areas from the test image $\mathbf{x}$.

2.2.2 Maximum Distance Between Expected Values Method

Another method of generating the coefficient matrix, $\mathbf{H}$, for the QCF is to maximize the distance between the expected values of the target and background statistics. Denote the statistics of a target image patch $\mathbf{x}_{tgt}$ and background image patch $\mathbf{x}_{bkg}$ by

$$\varphi(\mathbf{x}_{tgt}) = \mathbf{x}_{tgt}^T \mathbf{H} \mathbf{x}_{tgt} \tag{75}$$

and

$$\varphi(\mathbf{x}_{bkg}) = \mathbf{x}_{bkg}^T \mathbf{H} \mathbf{x}_{bkg}, \tag{76}$$

respectively. The objective is the find the filter $\mathbf{H}^*$such that

$$\mathbf{H}^* = \underset{\mathbf{H}}{\operatorname{argmax}} \left(\mathbb{E}[\varphi(\mathbf{x}_{tgt})] - \mathbb{E}[\varphi(\mathbf{x}_{bkg})] \right), \tag{77}$$

which maximizes the distance between the expected values of the target and background statistics.

If we assume that the matrix $\mathbf{H}$ can be decomposed such that $\mathbf{H} = \mathbf{Q}\mathbf{Q}^T - \mathbf{P}\mathbf{P}^T$ then the target and background statistics are

$$\varphi(\mathbf{x}_{tgt}) = \mathbf{x}_{tgt}^T (\mathbf{Q}\mathbf{Q}^T - \mathbf{P}\mathbf{P}^T) \mathbf{x}_{tgt} \tag{78}$$

and

$$\varphi(\mathbf{x}_{bkg}) = \mathbf{x}_{bkg}^T (\mathbf{Q}\mathbf{Q}^T - \mathbf{P}\mathbf{P}^T) \mathbf{x}_{bkg}. \tag{79}$$

respectively. Using the identity $\mathbf{x}^T\mathbf{x} = \mathrm{tr}(\mathbf{x}\mathbf{x}^T)$, we can rewrite (77) as

$$\mathbf{H}^* = \underset{\mathbf{H}}{\operatorname{argmax}}\Big(\mathbb{E}\big[\operatorname{tr}\big(\mathbf{Q}^T\mathbf{x}_{tgt}\mathbf{x}_{tgt}^T\mathbf{Q} - \mathbf{P}^T\mathbf{x}_{tgt}\mathbf{x}_{tgt}^T\mathbf{P}\big)\big] - \mathbb{E}\big[\operatorname{tr}\big(\mathbf{Q}^T\mathbf{x}_{bkg}\mathbf{x}_{bkg}^T\mathbf{Q} - \mathbf{P}^T\mathbf{x}_{bkg}\mathbf{x}_{bkg}^T\mathbf{P}\big)\big]\Big), \tag{80}$$

or

$$\mathbf{H}^* = \underset{\mathbf{H}}{\operatorname{argmax}}\big(\operatorname{tr}\{\mathbf{Q}^T\big(\mathbf{R}_{tgt} - \mathbf{R}_{bkg}\big)\mathbf{Q}\} - \operatorname{tr}\{\mathbf{P}^T\big(\mathbf{R}_{tgt} - \mathbf{R}_{bkg}\big)\mathbf{P}\}\big), \tag{81}$$

where $\mathbf{R}_{tgt} = \mathbb{E}\big[\mathbf{x}_{tgt}\mathbf{x}_{tgt}^T\big]$ and $\mathbf{R}_{bkg} = \mathbb{E}\big[\mathbf{x}_{bkg}\mathbf{x}_{bkg}^T\big]$. If we assume that the matrix $\big(\mathbf{R}_{tgt} - \mathbf{R}_{bkg}\big)$ can be decomposed such that

$$\mathbf{Q}^T\big(\mathbf{R}_{tgt} - \mathbf{R}_{bkg}\big)\mathbf{Q} = \mathbf{\Lambda} \tag{82}$$

and

$$\mathbf{P}^T\big(\mathbf{R}_{tgt} - \mathbf{R}_{bkg}\big)\mathbf{P} = \mathbf{\Gamma}, \tag{83}$$

where $\mathbf{Q}$ and $\mathbf{P}$ contain the eigenvectors of $\big(\mathbf{R}_{tgt} - \mathbf{R}_{bkg}\big)$ and $\mathbf{\Lambda}$ and $\mathbf{\Gamma}$ contain the corresponding eigenvalues, then (81) can be maximized by choosing the eigenvectors in $\mathbf{Q}$ corresponding to the positive eigenvalues in $\mathbf{\Lambda}$ and the eigenvectors in $\mathbf{P}$ corresponding to the negative eigenvalues in $\mathbf{\Gamma}$. The filter $\mathbf{H}^*$ that is made up of these two matrices can be applied to a new image patch $\mathbf{x}$ to generate the discriminator statistic

$$\varphi(\mathbf{z}) = \mathbf{x}^T\mathbf{H}^*\mathbf{x}, \tag{84}$$

which will be positive and large for target image patches and small or negative for background image patches, see Figure 4c.

2.2.3 Single-Layer Perceptron Method

A third method of generating the coefficient matrix $\mathbf{H}$ that was introduced in [18] minimizes the sum of the squared error between the output of the statistic φ and the true label d for all of the training

samples. This method is based on the backpropagation training algorithm for kernel neurons to generate the filter [37]. Denote the statistic of an image patch $\mathbf{x}_i$ by

$$\varphi(\mathbf{x}_i) = \mathbf{x}_i^T \mathbf{H} \mathbf{x}_i, 1 \leq i \leq N_e, \tag{85}$$

where $N_e = 2N_S$ is the total number of training images. The image patch $\mathbf{x}_i$ has the true label $d_i \in \{-1,1\}$ where -1 represents a background image patch and 1 represents a target image patch. As with the previous methods, the statistic φ in (85) is large and positive when image patch $\mathbf{x}_i$ is a target and small or negative if it is background. To force the output of the SLP to be between -1 and 1, we use a differentiable squashing function $\sigma\big(\varphi(\mathbf{x}_i)\big) = \tanh\big(\varphi(\mathbf{x}_i)\big)$. The objective function is

$$J(\mathbf{H}) = \frac{1}{2}\sum_{i=1}^{N_e} \big|d_i - \sigma\big(\varphi(\mathbf{x}_i)\big)\big|^2 = \frac{1}{2}\sum_{i=1}^{N_e} \big|d_i - \sigma\big(\mathbf{x}_i^T \mathbf{H} \mathbf{x}_i\big)\big|^2 \tag{86}$$

which has gradient

$$\nabla_H J(\mathbf{H}) = -\sum_{i=1}^{N_e} \big|d_i - \sigma\big(\varphi(\mathbf{x}_i)\big)\big| \Big(1 - \sigma^2\big(\varphi(\mathbf{x}_i)\big)\Big) \mathbf{x}_i \mathbf{x}_i^{\mathbf{T}}. \tag{87}$$

We can iteratively solve for the filter $\mathbf{H}$ using gradient descent

$$\mathbf{H}_{n+1} = \mathbf{H}_n - \eta \nabla_H J(\mathbf{H}), \tag{88}$$

where $n \geq 0$ is the iteration number, η is the learning rate and $\mathbf{H}_0 = \mathbf{0}_{r \times c}$ assuming that r and c denote the number of rows and columns in $\mathbf{H}$, respectively.

The best approximate solution to (88) can be denoted by $\mathbf{H}^*$ and occurs after a maximum number of iterations or a convergence criterion has been met. So now, (85) can be written in terms of the approximate solution

$$\varphi(\mathbf{x}_i) = \mathbf{x}_i^T \mathbf{H}^* \mathbf{x}_i, 1 \leq i \leq N_e. \tag{89}$$

This statistic is what will be used to distinguish target image patches from background, see Figure 4d.

2.3 Compressed Quadratic Correlation Filters

The compressed quadratic correlation filter (CQCF) was introduced in [18] as a modification of the QCF algorithm for compressive imagers so that its data could be classified without decompression of the sensed image. In this section, we will discuss how the STLS or weighted least squares and QCF algorithms can be combined to form a target detection algorithm on compressed samples. We will then show how the QCF algorithm can be modified to incorporate the compressed measurements directly using the sampling strategy from STLS.

2.3.1 Quadratic Correlation Filter Using STLS or Weighted Least Squares

The STLS and WLS recovery methods are alternatives to OMP, IIRLS or BP. The main advantage of the STLS or WLS as compared to these, and many other reconstruction methods is that they have closed-form solutions, which typically requires much less execution time than other iterative methods. This method takes a compressively sampled infrared image patch, $\mathbf{y}$, and reconstructs it using (55) together with (24) giving

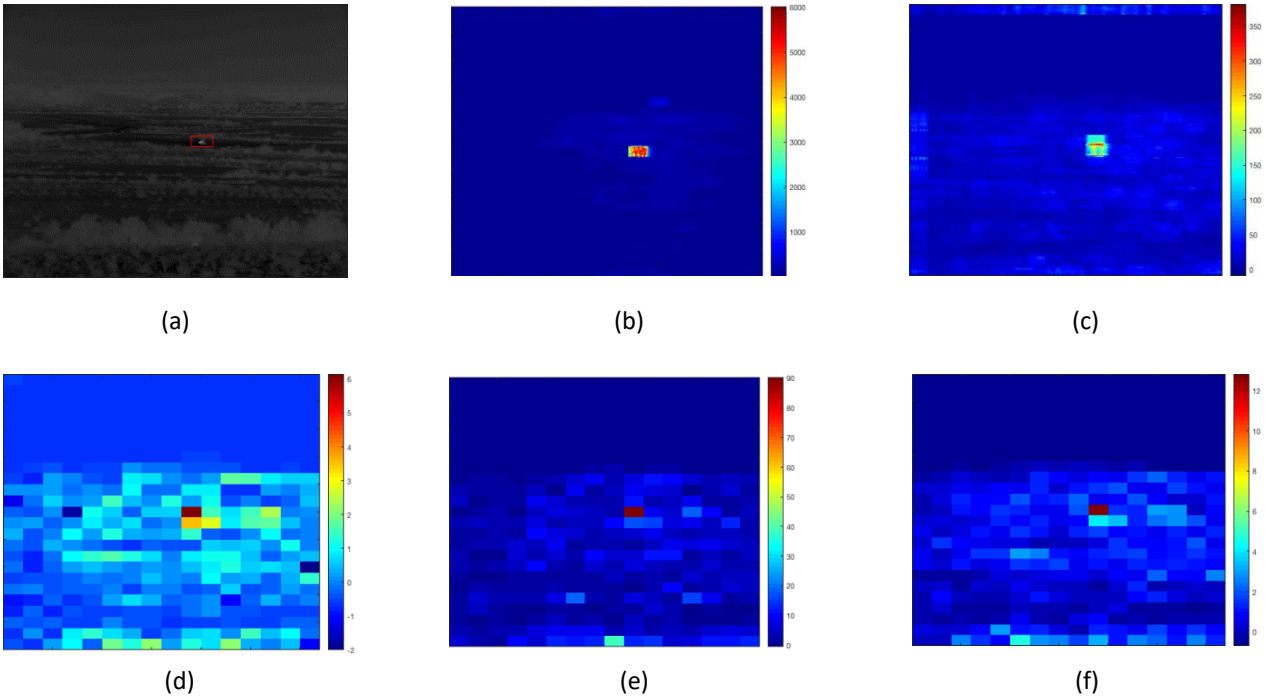
$$\hat{\mathbf{x}} = \boldsymbol{\Psi}\widehat{\boldsymbol{\Phi}}_T^{+}\mathbf{y} \tag{90}$$

for STLS reconstruction and (41) for weighted least squares reconstruction. Using (73), (84) or (89) and the recovered image patch from (90) or (41), we have

$$\varphi(\hat{\mathbf{x}}) = \hat{\mathbf{x}}^T\mathbf{H}\hat{\mathbf{x}} = \mathbf{y}^T\widetilde{\mathbf{H}}\mathbf{y}, \tag{91}$$

where $\widetilde{\mathbf{H}} = \mathbf{A}^T\mathbf{H}\mathbf{A}$ and $\mathbf{A} = \boldsymbol{\Psi}\widehat{\boldsymbol{\Phi}}_T^{+}$ for STLS where $\boldsymbol{\Phi}$ is the matrix define in (53) and $\mathbf{A} = \boldsymbol{\Psi}\mathbf{W}^{-1}\boldsymbol{\Psi}^T\boldsymbol{\Phi}^T(\boldsymbol{\Phi}\boldsymbol{\Psi}\mathbf{W}^{-1}\boldsymbol{\Psi}^T\boldsymbol{\Phi}^T)^{-1}$ for WLS where $\boldsymbol{\Phi}$ is a random Bernoulli matrix. In (91), the compressed infrared image patch, $\mathbf{y}$, is captured using the corresponding sensing matrix for each method. Then it is uncompressed using STLS or WLS and applied to a standard QCF filter. This can be implemented using optimized linear algebra libraries resulting in very fast execution speeds due to its closed-form. In

addition, because of the linear relationship, if the filter operation can be expressed in terms of the recovered image, then it can also be expressed in terms of the compressed image $\mathbf{y}$, giving a compressed target recognition method. This is one main advantage of ℓ_2 recovery methods to iterative methods. In this method, the filter $\mathbf{H}$ is $\mathbb{R}^{N\times N}$ dimensional space where $\mathbf{y}$ is in $\mathbb{R}^m$ dimensional space and $m < N$. The STLS or WLS operation performs a mapping from $\mathbb{R}^m$ to $\mathbb{R}^N$.



(a) (b) (c)

(d) (e) (f)

Figure 6 – Comparison of target detection methods on compressed data : (a) Original mid-wave infrared image containing a single Self-propelled howitzer (2S3) target. (b) Target detection statistic φ from QCF applied to the uncompressed original image. (c) QCF applied to a decompressed image using STLS. QCF applied to a 20×40 blockwise reconstruction of the image using (d) STLS and (e) WLS where $m = 50$, and (f) compressed quadratic correlation filter (CQCF) where $t = 50$. The sensing matrix for (e) is a random Bernoulli matrix and for (d) and (f) it is the matrix defined in (53).

2.3.2 The Compressed Quadratic Correlation Filter

Similar to the QCF, the CQCF can take multiple forms. In this section, we cover three methods that include the Fukunaga-Koontz transform method, the maximum distance between expected values method and the single-layer perceptron method first presented in [18].

2.3.2.1 Fukunaga-Koontz Transform Method

Section 2.3.1 showed how the STLS or WLS method could be used to create a pseudo-compressed quadratic correlation filter. It is not fully compressed because the filter is in the uncompressed domain. Due to the linear nature of QCF, we can create a compressed version of QCF and bypass decompression using just the sensing matrix in (53) for STLS which will further save processing time.

In this method, we denote a compressed target pattern as $\mathbf{y}_{tgt}$ and a compressed background pattern as $\mathbf{y}_{bkg}$ which are captured according to

$$\mathbf{y} = \mathbf{\Phi x}, \tag{92}$$

for STLS where $\mathbf{x}$ is an uncompressed target or background pattern and $\mathbf{\Phi}$ defined by (53). These patterns have correlation matrices

$$\widehat{\mathbf{R}}_{tgt} = \mathbb{E}\left[\mathbf{y}_{tgt}\mathbf{y}_{tgt}^T\right] \; and \; \widehat{\mathbf{R}}_{bkg} = \mathbb{E}\left[\mathbf{y}_{bkg}\mathbf{y}_{bkg}^T\right], \tag{93}$$

respectively. Following a derivation very similar to that presented in Section 2.2.1, we generate the statistic

$$\hat{\varphi} = \mathbf{y}^T\left(\widehat{\mathbf{F}}\widehat{\mathbf{F}}^T - \widehat{\mathbf{G}}\widehat{\mathbf{G}}\right)\mathbf{y} = \mathbf{y}^T\widehat{\mathbf{H}}\mathbf{y}, \tag{94}$$

where $\widehat{\mathbf{H}} \in \mathbb{R}^{m\times m}$.

The statistic in (94) has less eigenvalues than the method presented in the standard uncompressed QCF FKT method, see Figure 5b. As before, this statistic will have higher magnitude values for compressed target image patches and lower magnitude values for compressed background image patches. Therefore, we can use (94) to classify compressed image patches from a compressive MWIR sensor, see Figure 6f.

2.3.2.2 Maximum Distance Between Expected Values Method

Another method of generating a compressed coefficient matrix, $\widehat{\mathbf{H}}$, for the CQCF is to maximize the distance between the expected values of the compressed target and background image patch statistics. Denote the statistics of a compressed target image and background image patches as $\mathbf{y}_{tgt}$ and $\mathbf{y}_{bkg}$ by

$$\hat{\varphi}(\mathbf{y}_{tgt}) = \mathbf{y}_{tgt}^T \widehat{\mathbf{H}} \mathbf{y}_{tgt} \tag{95}$$

and

$$\hat{\varphi}(\mathbf{y}_{bkg}) = \mathbf{y}_{bkg}^T \widehat{\mathbf{H}} \mathbf{y}_{bkg}, \tag{96}$$

respectively. The objective is the find the filter $\widehat{\mathbf{H}}^*$such that

$$\widehat{\mathbf{H}}^* = \underset{\widehat{\mathbf{H}}}{\operatorname{argmax}}\left(\mathbb{E}\left[\hat{\varphi}(\mathbf{y}_{tgt})\right] - \mathbb{E}\left[\hat{\varphi}(\mathbf{y}_{bkg})\right]\right), \tag{97}$$

which maximizes the distance between the expected values of the target and background statistics. If we assume $\widehat{\mathbf{H}} = \widehat{\mathbf{Q}}\widehat{\mathbf{Q}}^T - \widehat{\mathbf{P}}\widehat{\mathbf{P}}^T$, then (97) can be written as

$$\widehat{\mathbf{H}}^* = \underset{\widehat{\mathbf{H}}}{\operatorname{argmax}}\left(\operatorname{tr}\left\{\widehat{\mathbf{Q}}^T\left(\widehat{\mathbf{R}}_{tgt} - \widehat{\mathbf{R}}_{bkg}\right)\widehat{\mathbf{Q}}\right\} - \operatorname{tr}\left\{\widehat{\mathbf{P}}^T\left(\widehat{\mathbf{R}}_{tgt} - \widehat{\mathbf{R}}_{bkg}\right)\widehat{\mathbf{P}}\right\}\right), \tag{98}$$

where $\widehat{\mathbf{R}}_{tgt} = \mathbb{E}\left[\mathbf{y}_{tgt}\mathbf{y}_{tgt}^T\right]$ and $\widehat{\mathbf{R}}_{bkg} = \mathbb{E}\left[\mathbf{y}_{bkg}\mathbf{y}_{bkg}^T\right]$. Just as before, $\widehat{\mathbf{Q}}$ will contain the vectors that correspond to the positive eigenvalues and $\widehat{\mathbf{P}}$ contain the vectors corresponding to the negative eigenvalues.

Given a new compressed infrared image patch, $\mathbf{y}$, we can classify it as a target or background using

$$\hat{\varphi}(\mathbf{y}) = \mathbf{y}^T \widehat{\mathbf{H}}^* \mathbf{y}, \tag{99}$$

where $\widehat{\mathbf{H}}^* \in \mathbb{R}^{m \times m}$. We observe that $\hat{\varphi}$ will be positive and large when the compressed image patch is a target and small or negative if it is background.

2.3.2.3 Single-Layer Perceptron Method

A third method of generating the coefficient matrix $\widehat{\mathbf{H}}$ minimizes the sum of the squared error between the output of the statistic $\hat{\varphi}$ and the true label d for all of the compressed training samples. Denote the statistic of a compressed image patch $\mathbf{y}_i$ by

$$\hat{\varphi}(\mathbf{y}_i) = \mathbf{y}_i^T \widehat{\mathbf{H}} \mathbf{y}_i, 1 \le i \le N_e, \tag{100}$$

where $N_e = 2N_S$ is the total number of compressed training samples. The compressed image patch $\mathbf{y}_i$ has the true label $d_i \in \{-1,1\}$ where -1 represents a background image patch and 1 represents a target compressed image patch. As with the previous compressed methods, the statistic $\hat{\varphi}$ in (100) is large and positive when image patch $\mathbf{y}_i$ is a target and small or negative if it is background. The objective function is

$$J(\widehat{\mathbf{H}}) = \frac{1}{2}\sum_{i=1}^{N_e} \left| d_i - \sigma\left(\hat{\varphi}(\mathbf{y}_i)\right)\right|^2 = \frac{1}{2}\sum_{i=1}^{N_e} \left| d_i - \sigma\left(\mathbf{y}_i^T \widehat{\mathbf{H}} \mathbf{y}_i\right)\right|^2 \tag{101}$$

which has gradient

$$\nabla_{\widehat{H}} J(\widehat{\mathbf{H}}) = -\sum_{i=1}^{N_e} \left| d_i - \sigma\left(\hat{\varphi}(\mathbf{y}_i)\right)\right| \left(1 - \sigma^2\left(\hat{\varphi}(\mathbf{y}_i)\right)\right) \mathbf{y}_i \mathbf{y}_i^{\mathbf{T}}. \tag{102}$$

We can iteratively solve for the filter $\widehat{\mathbf{H}}$ using gradient descent

$$\widehat{\mathbf{H}}_{n+1} = \widehat{\mathbf{H}}_n - \eta \nabla_H J(\widehat{\mathbf{H}}), \tag{103}$$

Given a test compressed image patch, $\mathbf{y}$, from the MWIR compressive sensor, we can classify it as target or background by examining $\hat{\varphi}(\mathbf{y}) = \mathbf{y}^T \widehat{\mathbf{H}}^* \mathbf{y}$ or $\sigma(\hat{\varphi}(\mathbf{y}))$ where the first will be positive if the compressed image patch is a target and negative otherwise and the second will be between 0 and +1 for a target and between -1 and 0 for background.

CHAPTER 3 - TARGET RECOGNITION

Automatic target recognition (ATR) includes two stages: detection and recognition [1, 4]. As shown in CHAPTER 2, detection involves separating targets from background clutter. The target identification step is typically more processing intensive than the target detection stage. This is the main reason why a fast and efficient target detection stage is employed [1, 4]. As with target detection, there are many target identification methods that can be employed for ATR. They range from model-based methods to learning classifiers [4]. Some examples of learning classifiers are support vector machines (SVMs) and neural networks (NNs).

Correlation filters are another method of classification. The correlation filter takes advantage of the linear nature of the correlation operation in another domain (like the Fourier domain). This provides a fast and efficient way to classify (and localize) targets in an image by simply looking for peaks on the correlation surface [38]. The MACH filter is one such correlation filter [39]. While correlation filters can be used for classification, they fail to perform as well as SVMs and NNs for target identification.

Support vector machines (SVMs) are a type of learning linear classifier that tries to find the line that maximally separates the closest samples from opposing classes [40]. The SVM can be augmented with a correlation filter to enhance the localization performance of the SVM. This proposed method is called the maximum margin correlation filter (MMCF) [19, 20].

Another type of learning classifier is a convolutional neural network. Convolutional neural networks (CNNs) were introduced by LeCun *et al* [41, 42] to build networks that are invariant to certain transformations of the inputs. Rather than performing the feature extraction manually, it is built into the network and learned through the training process [22]. A typical CNN has correlation and pooling layers followed by fully-connected layers for performing the feature extraction and classification. However in

[43], a CNN was augmented with a QCF layer to take advantage of the target discrimination and localization properties of the QCF along with the classification properties of the CNN.

In this chapter, we review a few learning-based target recognition approaches. The first approach we will review is a dual-objective support vector machine which has a max margin objective and a correlation peak objective. Then, we will then review how neural networks can be augmented with the quadratic correlation filter, which is inherently a two-class discriminator, to generate effective multi-class discriminators.

3.1 Maximum Margin Correlation Filter

The maximum margin correlation filter (MMCF) is a dual-objective constrained optimization problem introduced in [19, 20]. It contains a correlation filter objective and a maximum margin objective. To fully understand the MMCF, we will briefly review correlation filters and support vector machines in this section.

3.1.1 Correlation Filters

A correlation is used to measure the degree to which two signals are similar [28]. A finite one-dimensional cross-correlation can be expressed as

$$g[n] = \mathbf{x} \otimes \mathbf{w} = \sum_{m} x[m] w_N[n+m], \tag{104}$$

for a periodic signal w of period N. The correlation output g gives peaks when the input image is similar to the filter. An ideal output of the correlation is equal to the Kronecker delta function $g[n] = \delta[n]$ for $n \in [1, N]$. We can take the Fourier Transform of (104) to get

$$\hat{g}[k] = \sum_{n}\sum_{m} x[m]w_N[n+m]\, e^{-\frac{2\pi i n k}{N}}. \tag{105}$$

If we let $l = n + m$, then (105) can be rewritten as

$$\hat{g}[k] = \sum_{l}\sum_{m} x[m]w_N[l]\, e^{-\frac{2\pi i (l-m) k}{N}} \tag{106}$$

which can be reorganized as

$$\hat{g}[k] = \sum_{l} w_N[l] e^{-\frac{2\pi i l k}{N}} \sum_{m} x[m]\, e^{\frac{2\pi i m k}{N}} \tag{107}$$

or just

$$\hat{g}[k] = \hat{w}[k]\hat{x}^*[k], 1 \le k \le K \tag{108}$$

where * represents the complex conjugate and K is the length of the discrete Fourier transform. We can denote Fourier transform of the vector $\mathbf{x}$ as $\hat{\mathbf{x}} = \mathcal{F}\{\mathbf{x}\}$ and create a diagonal matrix $\widehat{\mathbf{X}}$ such that

$$\widehat{\mathbf{X}} = \begin{bmatrix} \hat{x}[1] & 0 & \cdots & 0 \\ 0 & \hat{x}[2] & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \hat{x}[K] \end{bmatrix} \tag{109}$$

and efficiently compute the vector $\hat{\mathbf{g}}$ using $\hat{\mathbf{g}} = \widehat{\mathbf{X}}^*\hat{\mathbf{w}}$ where $\hat{\mathbf{w}} = [\hat{w}[1] \quad \hat{w}[2] \quad \cdots \quad \hat{w}[K]]$.

To find the optimal filter that responds to similar patterns as a delta function, we solve

$$\mathbf{w} = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{i=1}^{L} \|\mathbf{x}_i \otimes \mathbf{w} - \mathbf{g}_i\|^2, \tag{110}$$

which can also be expressed as

$$\mathbf{w} = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{i=1}^{L} \|\mathbf{y}_i\|^2, \tag{111}$$

For $\mathbf{y}_i = \mathbf{x}_i \otimes \mathbf{w} - \mathbf{g}_i$. If we allow $\hat{\mathbf{y}} = \boldsymbol{\mathcal{F}}\{\mathbf{y}\}$, then (111) is just

$$\widehat{\mathbf{w}} = \underset{\widehat{\mathbf{w}}}{\operatorname{argmin}} \frac{1}{K} \sum_{i=1}^{L} \hat{\mathbf{y}}_i^* \hat{\mathbf{y}}_i, \tag{112}$$

where we used *Parseval's Theorem*. We can see that $\hat{\mathbf{y}}_i = \widehat{\mathbf{X}}_i^* \widehat{\mathbf{w}} - \hat{\mathbf{g}}_i$ which gives

$$\widehat{\mathbf{w}} = \underset{\widehat{\mathbf{w}}}{\operatorname{argmin}} \frac{1}{K} \sum_{l=1}^{L} \left(\widehat{\mathbf{w}}^\dagger \widehat{\mathbf{X}}_l \widehat{\mathbf{X}}_l^* \widehat{\mathbf{w}} - 2\widehat{\mathbf{w}}^\dagger \widehat{\mathbf{X}}_l \hat{\mathbf{g}}_l + \hat{\mathbf{g}}_l^\dagger \hat{\mathbf{g}}_l \right), \tag{113}$$

where the † operator denotes a conjugate transpose. The spatial domain filter is just $\mathbf{w} = \mathcal{F}^{-1}\{\widehat{\mathbf{w}}\}$ [38].

3.1.2 Support Vector Machines

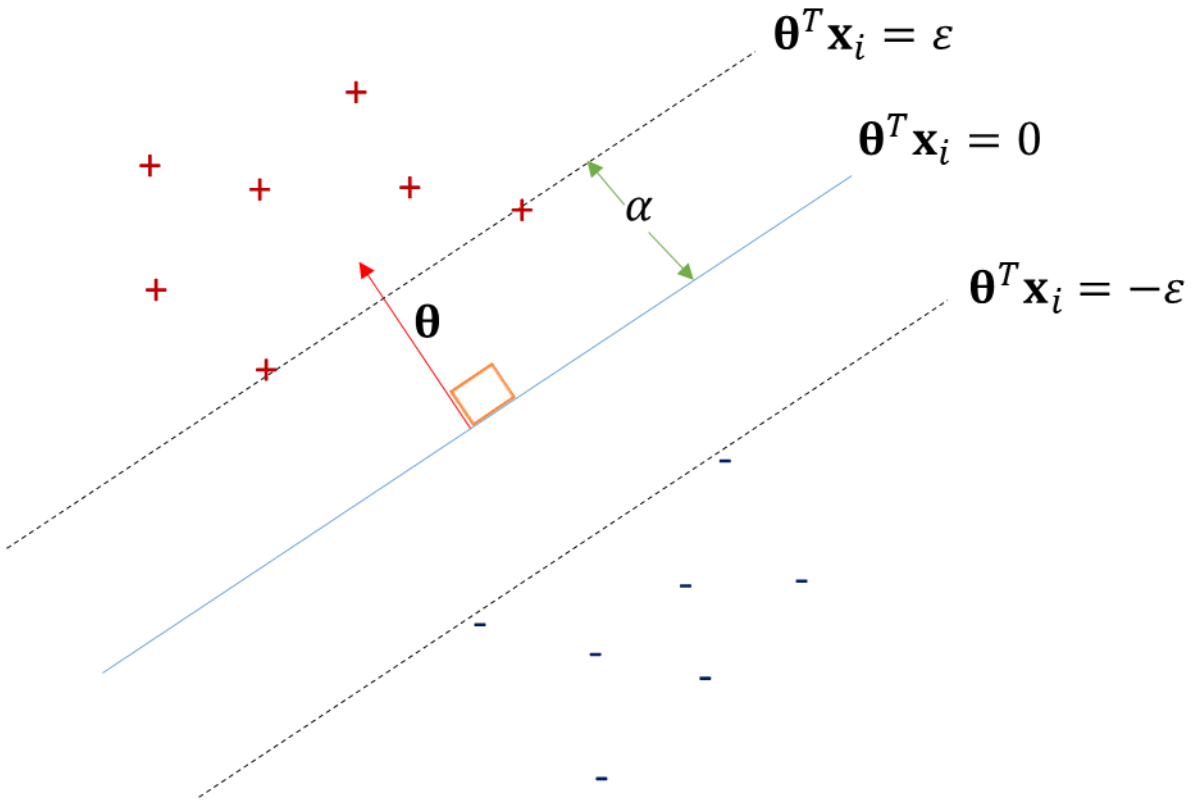


Figure 7 - Example of a support vector machine classification. Opposing classes are separated by a line ($\boldsymbol{\theta}^T \mathbf{x}_i = 0$) that maximizes the distance between them. The distance α is exactly half of that distance.

A support vector machines is a linear classifier which maximizes the distance between the closest opposing class members [40]. In this section, we will assume that an input pattern can be represented by

$\mathbf{x}_i$ and that it has a label $y_i \in \{-1,1\}$ for $1 \leq i \leq L$ where L is the number of training samples. Figure 7 provides and illustration of a support vector machine classification. The lines $\boldsymbol{\theta}^T\mathbf{x}_i = \varepsilon$ and $\boldsymbol{\theta}^T\mathbf{x}_i = -\varepsilon$ represent the support vectors $\mathbf{x}_i$ which correspond to the positive and negative classes, respectively. For simplicity in derivation, we have assumed that the support vectors are augmented with a one to account for the bias. The value, α, is the distance from each support vector to the classification line $\boldsymbol{\theta}^T\mathbf{x}_i = 0$ also called the *margin*. This equation for the classification line involves two vectors on the left-hand side. We know that two vectors $\mathbf{a}$ and $\mathbf{b}$ are orthogonal if $\mathbf{a}^T\mathbf{b} = 0$. From this definition, we can see that the vectors $\boldsymbol{\theta}$ and $\mathbf{x}_i$ are orthogonal. The margin α is the magnitude along the orthogonal vector $\boldsymbol{\theta}$ from the optimal classification line. Suppose we wish to find equation of a support vector in terms of the classification vector. We will assume that the classification vector is represented by $\mathbf{x}_i(0)$ and a support vector at the margin is $\mathbf{x}_i(\alpha)$. We will assume that α represents the distance between the two vectors, so the support vector is just

$$\mathbf{x}_i(\alpha) = \mathbf{x}_i(0) + \alpha\frac{y_i\boldsymbol{\theta}}{\|\boldsymbol{\theta}\|}, \tag{114}$$

where the scalar y_i is used to represent either the positive or negative support vector. To find a scalar value for α, we can multiply both sides by $y_i\boldsymbol{\theta}^T$. This gives

$$y_i\boldsymbol{\theta}^T\mathbf{x}_i(\alpha) = y_i\boldsymbol{\theta}^T\left[\mathbf{x}_i(0) + \alpha\frac{y_i\boldsymbol{\theta}}{\|\boldsymbol{\theta}\|}\right], \tag{115}$$

where $y_i\boldsymbol{\theta}^T\mathbf{x}_i(\alpha) = \varepsilon$ and $y_i\boldsymbol{\theta}^T\mathbf{x}_i(0) = 0$. We can rewrite this as

$$\varepsilon = \alpha\frac{\|\boldsymbol{\theta}\|^2}{\|\boldsymbol{\theta}\|}, \tag{116}$$

since $y_i^2 = 1$. Given this, we can see that the margin $\alpha = \frac{\varepsilon}{\|\boldsymbol{\theta}\|}$. So, to find the line that maximizes this margin, we solve

$$\max_{\boldsymbol{\theta},\mathrm{b}} \frac{\varepsilon}{\|\boldsymbol{\theta}\|} \text{ subject to } y_i(\boldsymbol{\theta}^T \mathbf{x}_i + b) \geq \varepsilon \text{ for } i = 1, \dots, L, \tag{117}$$

where we have explicitly included the bias (instead of using augmented vectors). This is the same as

$$\min_{\boldsymbol{\theta},\mathrm{b}} \frac{\|\boldsymbol{\theta}\|^2}{\varepsilon^2} \text{ subject to } y_i(\boldsymbol{\theta}^T \mathbf{x}_i + b) \geq \varepsilon \text{ for } i = 1, \dots, L \tag{118}$$

and can also be written as

$$\min_{\mathbf{w},\mathrm{b}} \|\mathbf{w}\|^2 \text{ subject to } y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 \text{ for } i = 1, \dots, L \tag{119}$$

if $\mathbf{w} = \frac{\boldsymbol{\theta}}{\varepsilon}$. This is the equation for a support vector machine and assumes there are no mistakes in the classification (i.e. the line perfectly separates the opposing classes). If the classes are not perfectly linearly separable then a *slack variable* is typically introduced and takes the form

$$\min_{\mathbf{w},\mathrm{b}} \|\mathbf{w}\|^2 + C \sum_i \xi_i \text{ subject to } y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i \text{ for } i = 1, \dots, L \tag{120}$$

for $\xi_i \geq 0$, where ξ_i are the slack variables [40].

There are several methods for converting a binary classifier to multi-class. One method is called the *one-versus-all* method. In one-versus-all, there is an SVM trained for each class with the negative class containing samples from all other classes. The SVM with the maximum response provides the label for a given test pattern [22].

3.1.3 Proposed Method

As previously mentioned, the maximum margin correlation filter (MMCF) is a dual-objective optimization problem. It combines the discrimination capability of support vector machines and the localization properties of correlation filters. The MMCF assumes that the output

$$g[n] = \sum_m \delta[n] x_i[m] w_N[n+m] = \delta[n] \mathbf{w}^T \mathbf{x}_i, 1 \le n \le N, \tag{121}$$

for $1 \le i \le L$, where L is the number of training images, which has transform

$$\hat{g}[k] = \sum_n \sum_m \delta[n] x_i[m] w_N[n+m] \, e^{-\frac{2\pi i n k}{N}} = \mathbf{w}^T \mathbf{x}_i, 1 \le k \le K, \tag{122}$$

or can be written in vector form as

$$\hat{\mathbf{g}} = \mathbf{e}\mathbf{x}_i^T \mathbf{w}, \tag{123}$$

where $\mathbf{e} = [1,1,1,\dots,1]^T$ and $\mathbf{e} \in \mathbb{R}^K$. From (122), we can take the inverse discrete Fourier transform of x and w giving

$$\hat{g}[k] = \sum_m \frac{1}{K} \sum_l \hat{x}_i[l] e^{\frac{2\pi i m l}{K}} \frac{1}{K} \sum_j \widehat{w}_N[j] \, e^{\frac{2\pi i m j}{K}}, 1 \le k \le K, \tag{124}$$

or

$$\hat{g}[k] = \frac{1}{K} \sum_l \hat{x}_i[l] \frac{1}{K} \sum_j \widehat{w}_N^*[j] \sum_m e^{\frac{2\pi i m (l-j)}{K}}, 1 \le k \le K. \tag{125}$$

But the last summation is just a geometric series equal to

$$\sum_m e^{\frac{2\pi i m (l-j)}{K}} = \frac{e^{2\pi i m (l-j)} - 1}{e^{\frac{2\pi i m (l-j)}{K}} - 1} = K\delta[l-j] \tag{126}$$

which gives

$$\hat{g}[k] = \frac{1}{K} \sum_l \hat{x}_i[l] \frac{1}{K} \sum_j \widehat{w}_N^*[j] \, K\delta[j-l] = \frac{1}{K} \sum_l \hat{x}_i[l] \widehat{w}_N^*[l] = \frac{1}{K} \widehat{\mathbf{w}}^\dagger \hat{\mathbf{x}}_i, \tag{127}$$

for $1 \le k \le K$ or in vector form it will be

$$\hat{\mathbf{g}} = \frac{1}{K} \mathbf{e} \hat{\mathbf{x}}_i^\dagger \widehat{\mathbf{w}}. \tag{128}$$

If we substitute this into (113), we have

$$\widehat{\mathbf{w}} = \underset{\widehat{\mathbf{w}}}{\operatorname{argmin}} \frac{1}{K} \sum_{l=1}^{L} \left(\widehat{\mathbf{w}}^{\dagger} \widehat{\mathbf{X}}_l \widehat{\mathbf{X}}_l^{*} \widehat{\mathbf{w}} - \frac{2}{K} \widehat{\mathbf{w}}^{\dagger} \widehat{\mathbf{X}}_l \mathbf{e} \hat{\mathbf{x}}_l^{\dagger} \widehat{\mathbf{w}} + \frac{1}{K^2} \widehat{\mathbf{w}}^{\dagger} \hat{\mathbf{x}}_l \mathbf{e}^T \mathbf{e} \hat{\mathbf{x}}_l^{\dagger} \widehat{\mathbf{w}} \right), \tag{129}$$

or

$$\widehat{\mathbf{w}} = \underset{\widehat{\mathbf{w}}}{\operatorname{argmin}} \frac{1}{K} \sum_{l=1}^{L} \left(\widehat{\mathbf{w}}^{\dagger} \widehat{\mathbf{X}}_l \widehat{\mathbf{X}}_l^{*} \widehat{\mathbf{w}} - \frac{2}{K} \widehat{\mathbf{w}}^{\dagger} \hat{\mathbf{x}}_l \hat{\mathbf{x}}_l^{\dagger} \widehat{\mathbf{w}} + \frac{1}{K} \widehat{\mathbf{w}}^{\dagger} \hat{\mathbf{x}}_l \hat{\mathbf{x}}_l^{\dagger} \widehat{\mathbf{w}} \right), \tag{130}$$

for $\widehat{\mathbf{X}}_l \mathbf{e} = \hat{\mathbf{x}}_l$ and $\mathbf{e}^T \mathbf{e} = K$. We can rewrite this as

$$\widehat{\mathbf{w}} = \underset{\widehat{\mathbf{w}}}{\operatorname{argmin}} \frac{1}{K} \widehat{\mathbf{w}}^{\dagger} \left(\sum_{l=1}^{L} \widehat{\mathbf{X}}_l \widehat{\mathbf{X}}_l^{*} - \frac{1}{K} \sum_{l=1}^{L} \hat{\mathbf{x}}_l \hat{\mathbf{x}}_l^{\dagger} \right) \widehat{\mathbf{w}} = \underset{\widehat{\mathbf{w}}}{\operatorname{argmin}} \frac{1}{K} \widehat{\mathbf{w}}^{\dagger} \hat{\mathbf{Z}} \widehat{\mathbf{w}}. \tag{131}$$

The MMCF operates in the frequency domain, so we must take the DFT of (120)

$$\min_{\mathbf{w},\mathrm{b}} \frac{1}{K} \widehat{\mathbf{w}}^{\dagger} \widehat{\mathbf{w}} + C \sum_i \xi_i \text{ subject to } \frac{y_i}{K} \left(\widehat{\mathbf{w}}^{\dagger} \hat{\mathbf{x}}_i + b \right) \geq 1 - \xi_i \text{ for } i = 1, \dots, L \tag{132}$$

for $\xi_i \geq 0$ where we have made use of *Parseval's Theorem* shown (112) and (127). This can also be put into a more general form

$$\min_{\mathbf{w},\mathrm{b}} \frac{1}{K} \widehat{\mathbf{w}}^{\dagger} \widehat{\mathbf{w}} + C \sum_i \xi_i \text{ subject to } \frac{y_i}{K} \left(\widehat{\mathbf{w}}^{\dagger} \hat{\mathbf{x}}_i + b \right) \geq y_i u_i - \xi_i \text{ for } i = 1, \dots, L, \tag{133}$$

where $\xi_i \geq 0$. If we let $\xi_i' = K\xi_i$, $b' = Kb$ and $c_i = K y_i u_i$, we can write (133) as

$$\min_{\mathbf{w},\mathrm{b}} \widehat{\mathbf{w}}^{\dagger} \widehat{\mathbf{w}} + C \sum_i \xi_i' \text{ subject to } y_i \left(\widehat{\mathbf{w}}^{\dagger} \hat{\mathbf{x}}_i + b' \right) \geq c_i - \xi_i' \text{ for } i = 1, \dots, L, \xi_i' \geq 0 \,. \tag{134}$$

To combine the SVM objective in (133) and the correlation filter objective in (131), we will use a parameter λ to control the contribution of each. Applying the parameter λ and adding the contribution of (131) to (133), we have

$$\min_{\mathbf{w},\mathrm{b}} \lambda \widehat{\mathbf{w}}^{\dagger} \widehat{\mathbf{w}} + \lambda C \sum_i \xi_i' + (1 - \lambda) \widehat{\mathbf{w}}^{\dagger} \hat{\mathbf{Z}} \widehat{\mathbf{w}} \tag{135}$$

$$\text{subject to } y_i\left(\hat{\mathbf{w}}^{\dagger}\hat{\mathbf{x}}_i + b'\right) \geq c_i - \xi_i' \text{ for } i = 1, \ldots, L, \xi_i' \geq 0 \,.$$

If we let $\hat{\mathbf{S}} = \lambda\mathbf{I} + (1-\lambda)\hat{\mathbf{Z}}$ and $C' = \lambda C$, then we have

$$\min_{\mathbf{w},\mathrm{b}} \hat{\mathbf{w}}^{\dagger}\hat{\mathbf{S}}\hat{\mathbf{w}} + C'\sum_i \xi_i' \qquad (136)$$

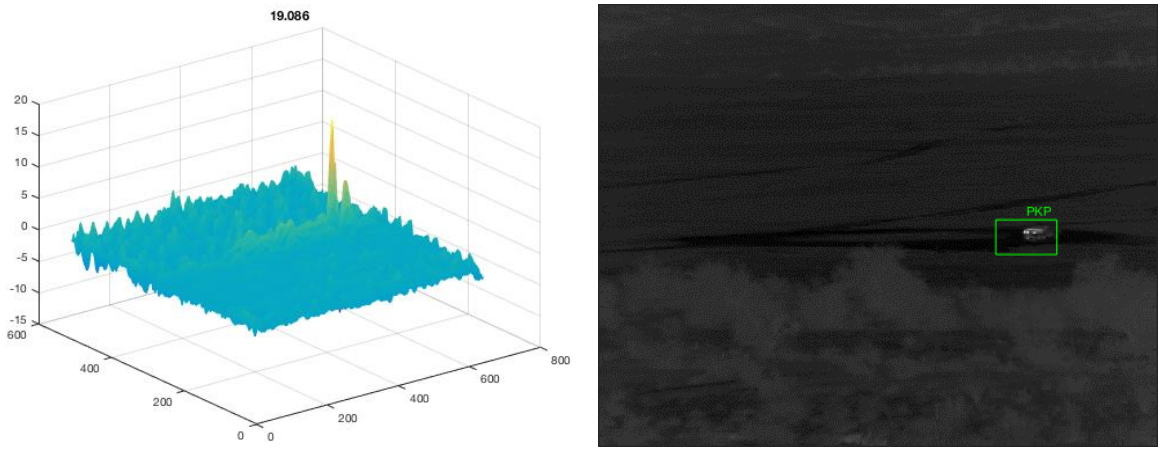
$$\text{subject to } y_i\left(\hat{\mathbf{w}}^{\dagger}\hat{\mathbf{x}}_i + b'\right) \geq c_i - \xi_i' \text{ for } i = 1, \ldots, L, \xi_i' \geq 0 \,.$$

To use a standard SVM solver, we have let $\tilde{\mathbf{w}} = \hat{\mathbf{S}}^{1/2}\hat{\mathbf{w}}$ and $\tilde{\mathbf{x}}_i = \hat{\mathbf{S}}^{-1/2}\hat{\mathbf{x}}_i$ which gives

$$\min_{\mathbf{w},\mathrm{b}} \tilde{\mathbf{w}}^{\dagger}\tilde{\mathbf{w}} + C'\sum_i \xi_i' \qquad (137)$$

$$\text{subject to } y_i\left(\tilde{\mathbf{w}}^{\dagger}\tilde{\mathbf{x}}_i + b'\right) \geq c_i - \xi_i' \text{ for } i = 1, \ldots, L, \xi_i' \geq 0 \,.$$

Figure 8 shows an example of the output of the MMCF and an application of the MMCF on an image using its identification and localization properties.



(a) (b)

Figure 8 – MMCF identification and location example: (a) the output of the MMCF, where the strongest peak indicates the presence of the target and the location, (b) the image with the PKP target identified by using the MMCF with the highest peak and centered on the peak.

3.2 Neural Network Approaches for Automatic Target Recognition

In Section 2.2.3, we recalled a single-layer perceptron QCF for two-class target detection. However, target recognition requires a multi-class discriminator. Therefore, in this section we propose a multi-layer perceptron neural network with a quadratic filter input layer for multi-class target recognition. Since multi-layer perceptron neural networks are fully-connected networks, the input size must match the training input patch size. To remedy this, we propose an all-convolutional CNN with a QCF layer for multi-class target recognition and take advantage of the invariance properties of the CNN [22]. In addition, all-convolutional CNN offers the ability to identify targets in an image while training on image patches [44]. This makes this type of CNN ideal for ATR applications.

3.2.1 Quadratic Multi-Layer Perceptron Neural Network for Target Recognition

In this subsection, we introduce a quadratic multi-layer perceptron neural network (QMLPNN) for multi-class target recognition. The quadratic multi-layer perceptron neural network (QMLPNN) for multi-class target recognition was introduced in [43]. To fully understand the QMLPNN, we will briefly review multi-layer perceptron neural networks and the backpropagation training algorithm.

3.2.1.1 Multi-Layer Perceptron Neural Network

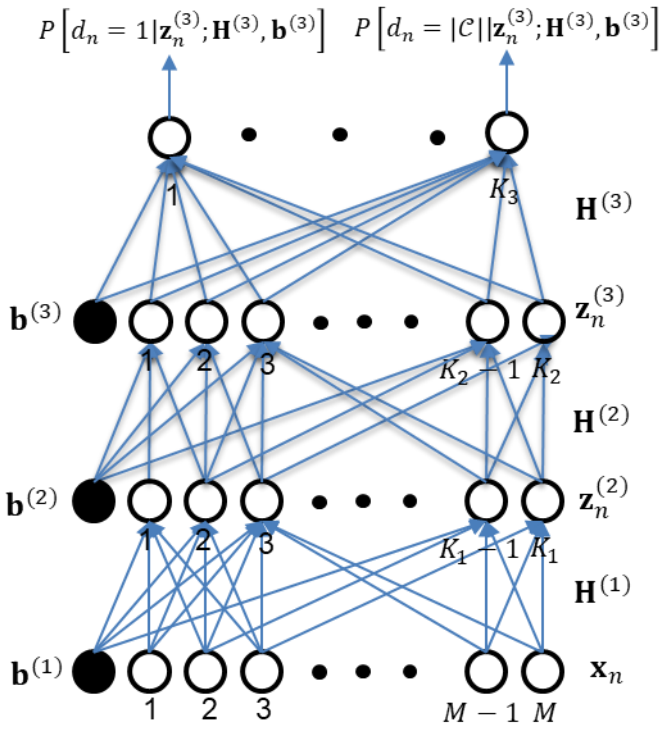


Figure 9 - Fully-connected multi-layer perceptron neural network. This network has an input layer, two hidden layers and an output softmax layer. The first hidden layer has K_1 nodes, the second has K_2 nodes and the output softmax layer has K_3 nodes which is equal to the number of classes. The input layer has M nodes with is the length of the vector $\mathbf{x}_n$. The weight matrices $\mathbf{H}^{(1)}$, $\mathbf{H}^{(2)}$ and $\mathbf{H}^{(3)}$ and bias vectors $\mathbf{b}^{(1)}$, $\mathbf{b}^{(2)}$ and $\mathbf{b}^{(3)}$ are learned from the training data using the backpropagation training algorithm.

Figure 9 illustrates as sample multi-layered perceptron neural network [22, 45-47]. In this example, there are three layers: an input layer, an output layer and a hidden layer. The number of nodes in the input layer is equal to the length of the input pattern vector $\mathbf{x}_n$ plus one for the bias node. The hidden layer nodes are denoted by $\mathbf{z}_n^{(2)}$ and $\mathbf{z}_n^{(3)}$ with lengths K_1 and K_2, respectively. These lengths are equal to the number of nodes in the hidden layer (not including the bias node). The first hidden layer can be determined using

$$\mathbf{z}_n^{(2)} = \sigma\left(\mathbf{H}^{(1)}\mathbf{x}_n + \mathbf{b}^{(1)}\right), \tag{138}$$

where $\mathbf{H}^{(1)}$ is the weight matrix and $\mathbf{b}^{(1)}$ is the bias vector for the first hidden layer. The function σ is a non-linear activation function which can be a sigmoid

$$\sigma(x) = \frac{1}{1 + e^{-x}}, \tag{139}$$

hyperbolic tangent

$$\sigma(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}, \tag{140}$$

rectified linear function

$$\sigma(x) = \max(x, 0) \tag{141}$$

or any number of other differentiable functions. The inner hidden layers can be determined using

$$\mathbf{z}_{\boldsymbol{n}}^{(\ell+1)} = \sigma\left(\mathbf{H}^{(\ell)}\mathbf{z}_{\boldsymbol{n}}^{(\ell)} + \mathbf{b}^{(\ell)}\right), 2 \leq \ell \leq L - 1. \tag{142}$$

Since this network only has three layers, we have $L = 3$. We reserve the weight matrix $\mathbf{H}^{(3)}$ and bias vector $\mathbf{b}^{(3)}$ for the objective function which is defined as

$$E(\mathbf{H}, \mathbf{b}) = -\sum_{n=1}^{N}\sum_{c\in\mathcal{C}}[d_n = c]\log\left(\frac{\exp\left(\langle\mathbf{h}_c^{(L)}, \mathbf{z}_n^{(L)}\rangle + \mathbf{b}_c^{(L)}\right)}{\sum_{j\epsilon\mathcal{C}}\exp\left(\langle\mathbf{h}_j^{(L)}, \mathbf{z}_n^{(L)}\rangle + \mathbf{b}_j^{(L)}\right)}\right), \tag{143}$$

where $\mathcal{C}$ is the set of possible target labels and

$$[d_n = c] = \begin{cases} 0 & d_n \neq c \\ 1 & d_n = c \end{cases} \tag{144}$$

is the Iverson bracket notation for the Kronecker delta function.

Backpropagation is used for neural networks to solve for the gradients for each layer so that gradient descent can be used to update the weights and biases. Gradient descent for a weight matrix is

$$\mathbf{H}^{(\ell)} = \mathbf{H}^{(\ell)} - \mu \nabla_{\mathbf{H}^{(\ell)}} E(\mathbf{H}, \mathbf{b}), 1 \leq \ell \leq L \tag{145}$$

and for the bias it is

$$\mathbf{b}^{(\ell)} = \mathbf{b}^{(\ell)} - \mu \nabla_{\mathbf{b}^{(\ell)}} E(\mathbf{H}, \mathbf{b}), 1 \leq \ell \leq L \tag{146}$$

where μ is the learning rate which can either be a static or dynamic number.

The gradient for the last layer is

$$\frac{\partial E(\mathbf{H}, \mathbf{b})}{\partial \mathbf{H}^{(L)}} = \boldsymbol{\delta}^{(L+1)} \left(\mathbf{z}_n^{(L)}\right)^T \tag{147}$$

where

$$\boldsymbol{\delta}^{(L)} = -\left(\sum_{n=1}^{N} [d_n = c] - \mathrm{P}\left[d_n = c \mid \mathbf{z}_n^{(L)}; \mathbf{H}^{(L)}, \mathbf{b}^{(L)}\right]\right)_{c \in \mathcal{C}} \tag{148}$$

and

$$\mathrm{P}\left[d_n = c \mid \mathbf{z}_n^{(L)}; \mathbf{H}^{(L)}, \mathbf{b}^{(L)}\right] = \frac{\exp\left(\langle \mathbf{h}_c^{(L)}, \mathbf{z}_n^{(L)} \rangle + \mathbf{b}_c^{(L)}\right)}{\sum_{j \epsilon \mathcal{C}} \exp\left(\langle \mathbf{h}_j^{(L)}, \mathbf{z}_n^{(L)} \rangle + \mathbf{b}_j^{(L)}\right)} \tag{149}$$

which is called the error function (see proof in APPENDIX B). To find the gradient for the weights in the next layer, we just perform the chain rule such as

$$\frac{\partial E(\mathbf{H}, \mathbf{b})}{\partial \mathbf{H}^{(L-1)}} = \frac{\partial E(\mathbf{H}, \mathbf{b})}{\partial \mathbf{z}_n^{(L)}} \frac{\partial \mathbf{z}_n^{(L)}}{\partial \mathbf{H}^{(L-1)}} = \left(\mathbf{H}^{(L)}\right)^T \boldsymbol{\delta}^{(L+1)} \frac{\partial \mathbf{z}_n^{(L)}}{\partial \mathbf{a}_n^{(L)}} \frac{\partial \mathbf{a}_n^{(L)}}{\partial \mathbf{H}^{(L-1)}} \tag{150}$$

where

$$\mathbf{a}_n^{(L)} = \mathbf{H}^{(L-1)} \mathbf{z}_{\boldsymbol{n}}^{(L-1)} + \mathbf{b}^{(L-1)}. \tag{151}$$

We will denote $\frac{\partial \mathbf{z}_n^{(L)}}{\partial \mathbf{a}_n^{(L)}}$ as $\sigma'\left(\mathbf{a}_n^{(L)}\right)$ and create another error function equal to

$$\boldsymbol{\delta}^{(L)} = \left(\mathbf{H}^{(L)}\right)^T \boldsymbol{\delta}^{(L+1)} \circ \sigma'\left(\mathbf{a}_n^{(L)}\right) \tag{152}$$

where $\circ$ denotes the elementwise Hadamard product. So, now we have

$$\nabla_{\mathbf{H}^{(L-1)}} E(\mathbf{H}, \mathbf{b}) = \frac{\partial E(\mathbf{H}, \mathbf{b})}{\partial \mathbf{H}^{(L-1)}} = \boldsymbol{\delta}^{(L)} \left(\mathbf{z}_{\boldsymbol{n}}^{(L-1)}\right)^{\boldsymbol{T}} \tag{153}$$

So, in general, we have for an internal layer

$$\boldsymbol{\delta}^{(\ell-1)} = \left(\mathbf{H}^{(\ell)}\right)^T \boldsymbol{\delta}^{(\ell)} \circ \sigma'\left(\mathbf{a}_n^{(\ell)}\right), 3 \leq \ell \leq L \tag{154}$$

and the gradient is

$$\nabla_{\mathbf{H}^{(\ell)}} E(\mathbf{H}, \mathbf{b}) = \frac{\partial E(\mathbf{H}, \mathbf{b})}{\partial \mathbf{H}^{(\ell)}} = \boldsymbol{\delta}^{(\ell+1)} \left(\mathbf{z}_{\boldsymbol{n}}^{(\ell)}\right)^T, 2 \leq \ell \leq L \tag{155}$$

for the weights and

$$\nabla_{\mathbf{b}^{(\ell)}} E_n(\mathbf{H}, \mathbf{b}) = \boldsymbol{\delta}^{(\ell+1)} \tag{156}$$

for the bias vectors with the first layer being

$$\nabla_{\mathbf{H}^{(1)}} E(\mathbf{H}, \mathbf{b}) = \frac{\partial E(\mathbf{H}, \mathbf{b})}{\partial \mathbf{H}^{(1)}} = \boldsymbol{\delta}^{(2)} (\mathbf{x}_n)^T \tag{157}$$

$$\nabla_{\mathbf{b}^{(1)}} E(\mathbf{H}, \mathbf{b}) = \boldsymbol{\delta}^{(2)}.$$

3.2.1.2 Proposed Method

In this section, we present the quadratic multi-layered perceptron (QMLPNN) from [43] which combines the properties of the quadratic filter and the discrimination capability of a multi-layer perceptron neural network. In this network configuration, we assume that a training pattern or image patch $\mathbf{x}_n, 1 \leq n \leq N$ of length M is the input to the first layer of the network. The first hidden layer

consists of a weight tensor, which is learned through the network backpropagation training and takes the form

$$\mathbf{z}_n^{(2)} = \left(\sigma\left(\mathbf{x}_n^T \mathbf{H}_k^{(1)} \mathbf{x}_n + b_k^{(1)}\right)\right)_{1 \le k \le K_1}, \tag{158}$$

where K_1 is the number of nodes in the hidden layer, $\left(\mathbf{H}_k^{(1)}\right)_{1 \le k \le K_1}$ is a learned three-dimensional weight tensor of size $M \times M \times K_1$, $\left(b_k^{(1)}\right)_{1 \le k \le K_1}$ is the bias vector and σ is a non-linear activation function such as a hyperbolic tangent or sigmoid. The internal layers in the network are conventional multi-layer perceptron layers

$$\mathbf{z}^{(\ell+1)} = \left(\sigma\left(\langle \mathbf{h}_k^{(\ell)}, \mathbf{z}_n^{(\ell)} \rangle + b_k^{(\ell)}\right)\right)_{1 \le k \le K_\ell}, \qquad 2 \le \ell \le L-1, \tag{159}$$

where $\mathbf{H}^{(\ell)} = \left(\mathbf{h}_k^{(\ell)}\right)_{1 \le k \le K_\ell}$ is the learned weight matrix of size $K_\ell \times K_{\ell-1}$, $\mathbf{b}^{(\ell)} = \left(b_k^{(\ell)}\right)_{1 \le k \le K_\ell}$ represents the learned bias vector for layer ℓ and L is the total number of internal hidden layers [22, 45, 46].

We denote the output of the network as $d_n \in \mathcal{C}, 1 \le n \le N$ where $\mathcal{C} \coloneqq \{\text{Pickup}, \text{SUV}, \dots, \text{ZSU23} - 4, \text{Background}\}$ is the set of all possible network outputs. We note that the last layer in the network has K_L nodes equal to the size of the set $\mathcal{C}$ (i.e. $K_L = |\mathcal{C}|$). Given this, the learned weight matrix in the last layer is denoted as $\mathbf{H}^{(L)} = \left(\mathbf{h}_c^{(L)}\right)_{c \in \mathcal{C}}$ for convenience. The objective function for the network $E_n(\mathbf{H}, \mathbf{b})$ can be defined as

$$E_n(\mathbf{H}, \mathbf{b}) = -\sum_{c \in \mathcal{C}} [d_n = c] \log\left(\frac{\exp\left(\langle \mathbf{h}_c^{(L)}, \mathbf{z}_n^{(L)} \rangle + \mathbf{b}_c^{(L)}\right)}{\sum_{j \in \mathcal{C}} \exp\left(\langle \mathbf{h}_j^{(L)}, \mathbf{z}_n^{(L)} \rangle + \mathbf{b}_j^{(L)}\right)}\right), \tag{160}$$

where $\mathbf{H} = \left\{\mathbf{H}^{(1)}, \mathbf{H}^{(2)}, \dots, \mathbf{H}^{(L)}\right\}$ and $\mathbf{b} = \left\{\mathbf{b}^{(1)}, \mathbf{b}^{(2)}, \dots, \mathbf{b}^{(L)}\right\}$. Given this, the total network objective is just

$$E(\mathbf{H},\mathbf{b}) = \sum_{n=1}^{N} E_n(\mathbf{H},\mathbf{b}). \tag{161}$$

Training for this network is accomplished using gradient descent for each layer with gradients determined by the backpropagation algorithm for neural networks. The gradient for the weight matrices and bias vectors are determined using (155) and (156), respectively, for internal layers $2 \leq \ell \leq L-1$ [46].

The architecture of the first layer is specifically designed for target recognition and has a unique gradient of the form

$$\nabla_{\mathbf{H}_k^{(1)}} E_n(\mathbf{H},\mathbf{b}) = \delta_k^{(2)} \mathbf{x}_n \mathbf{x}_n^T, \qquad 1 \leq k \leq K_1, \tag{162}$$

for the weight tensor $\left(\mathbf{H}_k^{(1)}\right)_{1\leq k\leq K_1}$ and

$$\nabla_{\mathbf{b}^{(1)}} E_n(\mathbf{H},\mathbf{b}) = \boldsymbol{\delta}^{(2)} \tag{163}$$

for the bias vector. Using gradient descent, we can determine the weight tensor in the first layer using

$$\mathbf{H}_k^{(1)} = \mathbf{H}_k^{(1)} - \mu \nabla_{\mathbf{H}_k^{(1)}} E(\mathbf{H},\mathbf{b}), 1 \leq k \leq K_1. \tag{164}$$

3.2.2 Quadratic Correlation Filter Convolutional Neural Network for Target Recognition

The quadratic correlation filter convolution neural network (QCFCNN) first presented in [43] is an alternative target recognition approach to the MMCF. By using a fully convolutional CNN, we can train on image patches and allow the network to localize the targets on full images [48] as well as identify the target type using an augmented convolutional neural network. The weight tensor in the first hidden layer $\mathbf{H}^{(1)} = \left(\mathbf{H}_k^{(1)}\right)_{1\leq k\leq K_1}$ contains multiple quadratic correlation filters $\mathbf{H}_k^{(1)}$ of size $M_1 \times M_1$ that are applied to the input image. To apply the QCF to the input image, we define

$$\boldsymbol{\Theta}_k^{(1)} = \mathbf{H}_k^{(1)} + \left(\mathbf{H}_k^{(1)}\right)^T, \tag{165}$$

and perform its eigendecomposition

$$\boldsymbol{\Theta}_k^{(1)}\mathbf{V}_k = \mathbf{V}_k\boldsymbol{\Lambda}_k, \tag{166}$$

where the diagonal matrix $\boldsymbol{\Lambda}_k$ has real eigenvalues $\lambda_{k,M_1} \geq \lambda_{k,M_1-1} \geq \cdots \geq \lambda_{k,1}$ as its diagonal entries and orthonormal matrix $\mathbf{V}_k = \left[\mathbf{v}_{k,1}, \ldots, \mathbf{v}_{k,M_1}\right]$ contains the corresponding eigenvectors. The first layer is

$$\mathbf{z}_n^{(2)} = \left(\sigma\left(\sum_{i=1}^{M_1} \lambda_{k,i}\left|\mathbf{x}_n \otimes \mathbf{v}_{k,i}\right|^2 + b_k^{(1)}\right)\right)_{1\leq k\leq K_1} \tag{167}$$

for every location in the image $\mathbf{x}$ of height M and width K, where $\otimes$ indicates the 2D cross-correlation operation. The internal layers are standard convolutional layers

$$\mathbf{z}_n^{(\ell+1)} = \left(\sigma\left(\sum_{i=1}^{K_{\ell-1}} \mathbf{z}_{n,i}^{(\ell)} \otimes \boldsymbol{h}_{k,i}^{(\ell)} + b_k^{(\ell)}\right)\right)_{1\leq k\leq K_\ell}, \qquad 2 \leq \ell \leq L-1, \tag{168}$$

where L is the total number of layers in the CNN.

Similar to the multi-layer perceptron networks, gradient descent is used for training with gradients defined through standard backpropagation for convolutional neural networks. The standard error function for CNN network internal layers is

$$\boldsymbol{\delta}_k^{(\ell)} = \text{upsample}\left(\sum_{i=1}^{K_{\ell+1}} \mathbf{h}_{i,k}^{(\ell)} \otimes_f \boldsymbol{\delta}_i^{(\ell+1)}\right) \circ \sigma'\left(\sum_{i=1}^{K_{\ell-1}} \mathbf{z}_{n,i}^{(\ell)} \otimes \mathbf{h}_{k,i}^{(\ell)} + b_k^{(\ell)}\right). \tag{169}$$

which can be used to define the gradient, where , $2 \leq \ell \leq L-1$, $1 \leq k \leq K_\ell$ and the notation $\otimes_f$ denotes the full 2D cross-correlation operation. The gradient for the internal network layers is defined as

$$\nabla_{\mathbf{H}^{(\ell)}} E_n(\mathbf{H}, \mathbf{b}) = \left(\mathbf{z}_{\boldsymbol{n}}^{(\ell)} \otimes \boldsymbol{\delta}_k^{(\ell+1)} \right)_{1 \le k \le K_\ell}, \tag{170}$$

and similarly for the bias, we define

$$\nabla_{\mathbf{b}^{(\ell)}} E_n(\mathbf{H}, \mathbf{b}) = \left(\sum_{x,y} \left(\boldsymbol{\delta}_k^{(\ell+1)} \right)_{\boldsymbol{x},\boldsymbol{y}} \right)_{1 \le k \le K_\ell}. \tag{171}$$

However, for the first layer gradient definition, we must first define the vectorization of a $d \times d$ subimage $\mathbf{X}_{l,m}^{(n)}$

$$\begin{aligned} &\operatorname{vec}\left(\mathbf{X}_{l,m}^{(n)} \right) \\ &= \left[x_{l,m}^{(n)}, \dots, x_{l,m+d-1}^{(n)}, x_{l+1,m}^{(n)}, \dots, x_{l+1,m+d-1}^{(n)}, \dots, x_{l+d-1,m}^{(n)}, \dots, x_{l+d-1,m+d-1}^{(n)} \right]^T. \end{aligned} \tag{172}$$

The gradient for the first layer can now be defined as

$$\nabla_{\mathbf{H}^{(1)}} E_n(\mathbf{H}, \mathbf{b}) = \frac{\partial E_n}{\partial \mathbf{H}_k^{(1)}} = \frac{\partial E_n}{\partial \mathbf{\Theta}_k^{(1)}} \frac{\partial \mathbf{\Theta}_k^{(1)}}{\partial \mathbf{H}_k^{(1)}} = \sum_{l,m} \delta_k^{(2)} \operatorname{vec}\left(\mathbf{X}_{l,m}^{(n)} \right) \operatorname{vec}\left(\mathbf{X}_{l,m}^{(n)} \right)^T, \tag{173}$$

where $1 \le k \le K_1$. Figure 10 demonstrates how a trained QCFCNN network can be utilized to localize and identify targets. The mode of each connected component is used to identify the type of target whereas the location of the target is determined by the centroid of each connected component.

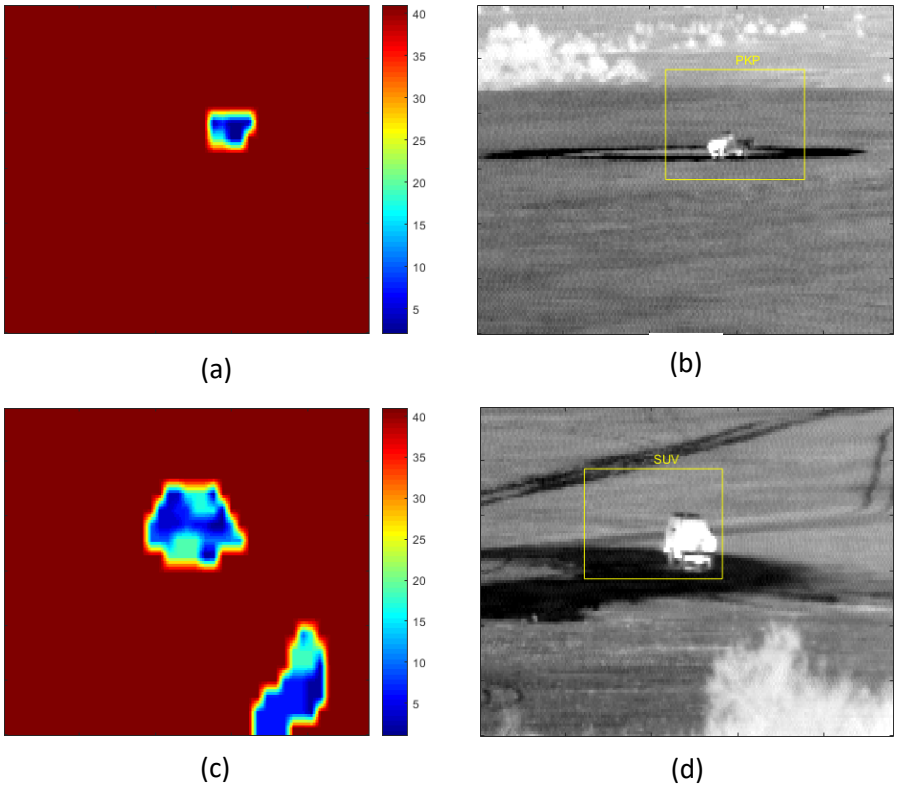


Figure 10 – Sample target recognition examples using QCFCNN: (a) and (c) contain target identification map outputs from the QCFCNN for the PKP and SUV target types, respectively. (b) and (d) are pictures of the identified target using a 40×80 bounding box.

CHAPTER 4 – COMPRESSED TARGET RECOGNITION

Target recognition, like target detection, on a compressive imaging system can be performed in traditional ways after full image reconstruction has taken place. However, just as with compressed target detection, a more efficient approach is to process the compressed measurements directly [49]. Since the goal is to identify targets, a full image reconstruction is not necessary [50-52].

The sensor presented in Figure 11 represents a block-wise compressive sensing architecture. While similar to the single-pixel camera [7], the block-wise compressive sensing structure uses light from blocks on the spatial light modulator (SLM), in this case a DMD, to multiple detectors on a photodetector array while the single-pixel camera uses light from all elements of the SLM to a single photodetector. In this architecture, the FPA senses mid-wave infrared (MWIR) signals.

We assume that an image on the DMD can be represented by $\mathbf{X}$, which can be broken up into sub-images of size $t \times s$

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_{1,1} & \mathbf{X}_{1,2} & \dots & \mathbf{X}_{1,Q} \\ \mathbf{X}_{2,1} & \mathbf{X}_{2,2} & \cdots & \mathbf{X}_{2,Q} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{X}_{P,1} & \mathbf{X}_{P,2} & \dots & \mathbf{X}_{P,Q} \end{bmatrix}, \tag{174}$$

where $\mathbf{X}_{p,q}$ represents a sub-image captured by an DMD block onto a single photodetector on the photodetector array. Using the definition of vectorization given by (172), we can represent a sub-image by a vector. We will denote a sub-image in vectorized form by $\mathbf{x}_{p,q}$ using

$$\mathbf{x}_{\boldsymbol{p},q} = \text{vec}\left(\mathbf{X}_{\boldsymbol{p},q}\right), 1 \le p \le P, 1 \le q \le Q, \tag{175}$$

forming a $ts \times 1$ vector for a $t \times s$ DMD block. A set of codes on the DMD will be represented by $\mathbf{\Phi}$ and a compressed measurement captured by the photodetector from the DMD block is denoted by $\mathbf{y}_{p,q}$ where

$$\mathbf{y}_{p,q} = \mathbf{\Phi}\mathbf{x}_{p,q}, 1 \le p \le P, 1 \le q \le Q. \tag{176}$$

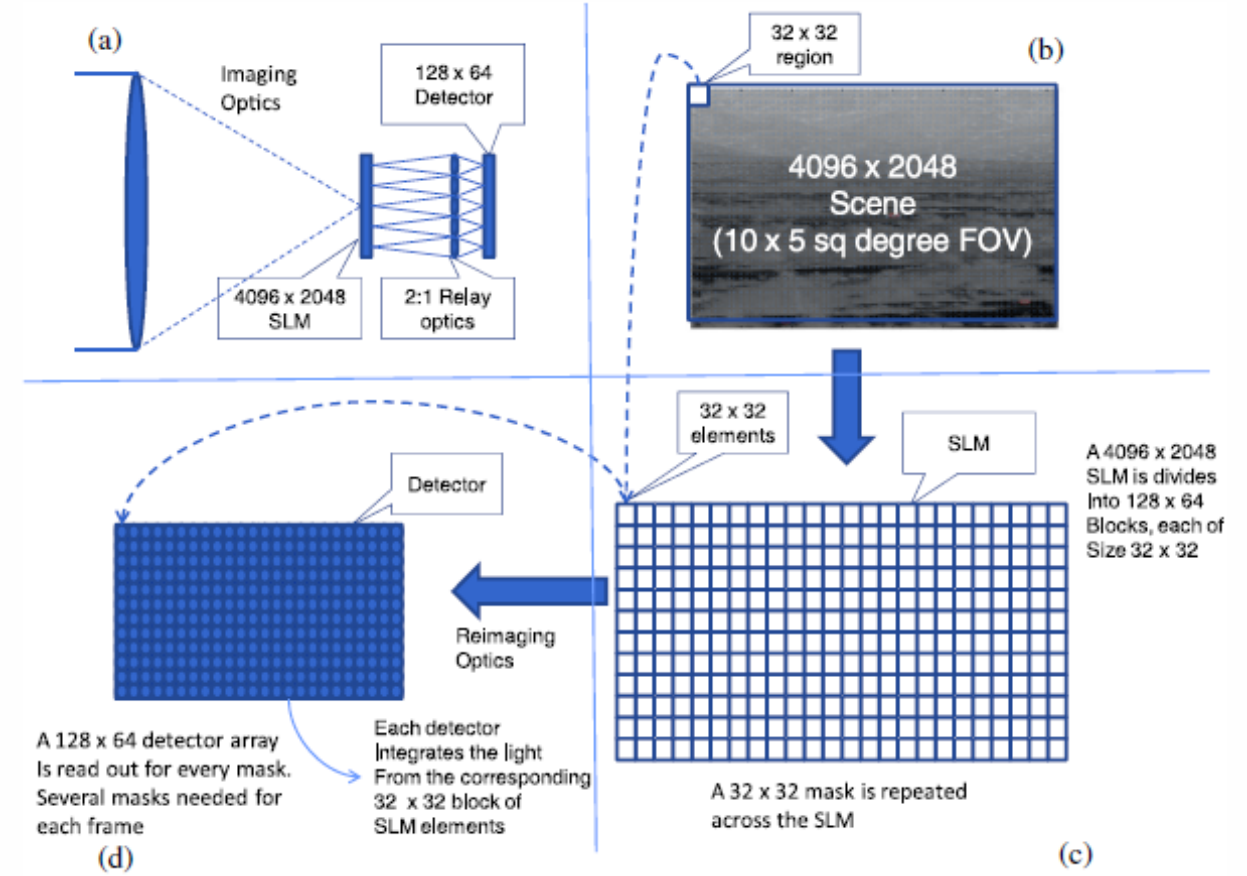


Figure 11 – Architecture for block-wise compressive sensing. : (a) Scene is optically imaged on an SLM, in this case a DMD, and a coded version is reimaged onto a photodetector array. The scene is partitioned in block (b) that is weighted by a code at the SLM (c). The relay optical system is designed such that light from one coded block is collected on a single element on the photodetector array (d). Reprinted with permission [13].

Just as in (16), this vectorized image can be sparsely represented in a basis $\mathbf{\Psi}$

$$\mathbf{x}_{p,q} = \mathbf{\Psi}\boldsymbol{\alpha}_{p,q}, 1 \le p \le P, 1 \le q \le Q \tag{177}$$

where $\boldsymbol{\alpha}_{p,q}$ is the sparse representation of $\mathbf{x}_{p,q}$. We can employ the STLS or WLS recovery methods given in Section 2.1 to find an approximate recovered image or use (17) to find a more exact solution. We will denote the solution to STLS or WLS as $\boldsymbol{\alpha}^*_{\boldsymbol{p,q}}$ and a recovered sub-image by $\hat{\mathbf{x}}_{p,q}$ such that

$$\hat{\mathbf{x}}_{p,q} = \mathbf{\Psi}\boldsymbol{\alpha}^*_{p,q}, 1 \le p \le P, 1 \le q \le Q. \tag{178}$$

A completely reconstructed image $\widehat{\mathbf{X}}$ is then given by

$$\widehat{\mathbf{X}} = \begin{bmatrix} \widehat{\mathbf{X}}_{1,1} & \widehat{\mathbf{X}}_{1,2} & \dots & \widehat{\mathbf{X}}_{1,Q} \\ \widehat{\mathbf{X}}_{2,1} & \widehat{\mathbf{X}}_{2,2} & \cdots & \widehat{\mathbf{X}}_{2,Q} \\ \vdots & \vdots & \ddots & \vdots \\ \widehat{\mathbf{X}}_{P,1} & \widehat{\mathbf{X}}_{P,2} & \dots & \widehat{\mathbf{X}}_{P,Q} \end{bmatrix},$$

where $\widehat{\mathbf{X}}_{p,q}$ is the vector $\hat{\mathbf{x}}_{p,q}$ of length $ts \times 1$ reshaped into a $t \times s$ sub-image.

4.1 Related Work

Compressive target classification has been addressed in other works, but most methods attempt to classify the entire sample rather than localize the target within the sample. In [53], which is an extension to the work in [54-57], a method is proposed to find the closest point on a k-dimensional manifold to the compressive measurement, which represents the most likely class for the compressive measurement under consideration. Alternatively, the work in [51] primarily deals with object detection in DCT-based compression methods like JPEG or MPEG. In this work, the distance criterion in eigenspace is used to classify images. The linearity of the DCT transform preserves the distance measurement so that the eigendecomposition can be used. Lastly in [50, 52], a MACH correlation filter is integrated in a polynomial correlation filter, also known as the discrete wavelet transform, to form a tiered object recognition filter. This is similar to our approach of integrating a correlation filter, but the neural network in our method offers the ability to learn a non-linear mapping of inputs to outputs that the MACH filter does not offer.

4.2 Target Recognition from Compressed Samples

4.2.1 Compressed Target Detection for Target Recognition

Target detection is typically used for reducing false detections in target recognition as mentioned in CHAPTER 2. Similarly, in compressed target recognition, compressed target detection is used to find

candidate image areas for application of compressed target recognition and localization algorithms. A candidate set of potential target areas is determined using the discrimination criterion

$$\mathrm{S} \coloneqq \{p \in [1,P], q \in [1,Q] \mid \varphi(\mathbf{y}_{p,q}) \geq \kappa\} \tag{179}$$

where κ is the candidate threshold value. The target recognition algorithm will only consider compressed image patch $\mathbf{y}_{p,q}$ such that $p, q \in S$ and its neighbors, which will allow an accurate identification.

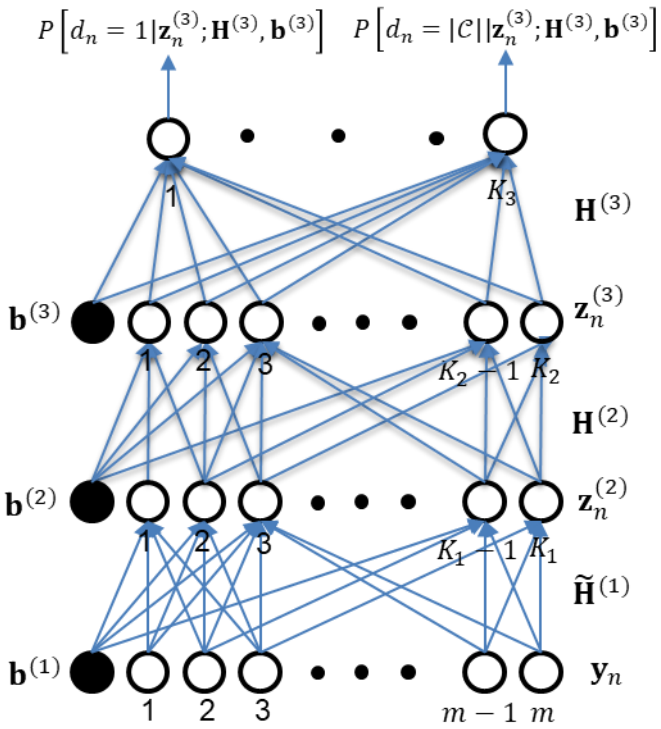


Figure 12 – Example of a Compressed Multi-Layered Perceptron Neural Network. In this network, the first layer has and compressed weight matrix with compression values determined by the WLS or STLS reconstruction method. The other layers have weights that are learned through the backpropagation algorithm.

4.2.2 Compressed Target Recognition Using a Multi-Layered Perceptron Neural Network

In Figure 12, a sample compressed multi-layered perceptron neural network is shown. Similar to the method presented in Section 3.2.1, this network can also be augmented to support compressed target recognition by replacing the first weight layer with a compressed three-dimensional tensor. We will

denote a compressed input pattern, indexed by n, by $\mathbf{y}_n$ for $1 \leq n \leq N$ where N denotes the number of training input patterns. Throughout this section, we will assume that $\hat{\mathbf{x}}_n$ is a reconstructed image using either the STLS method presented in Section 2.1.2 such that $\hat{\mathbf{x}}_n = \boldsymbol{\Psi}\widehat{\boldsymbol{\Phi}}_T^{+}\mathbf{y}_n = \mathbf{A}\mathbf{y}_n$ with $\widehat{\boldsymbol{\Phi}}$ defined by (53).

The output of the first hidden layer is a quadratic function with a compressed filter the is learned by the backpropagation algorithm, which can be defined as

$$\mathbf{z}_n^{(2)} = \left(\sigma\left(\hat{\mathbf{x}}_n^T\mathbf{H}_k^{(1)}\hat{\mathbf{x}}_n + b_k^{(1)}\right)\right)_{1\leq k\leq K_1} = \left(\sigma\left(\mathbf{y}_n^T\widetilde{\mathbf{H}}_k^{(1)}\mathbf{y}_n + b_k^{(1)}\right)\right)_{1\leq k\leq K_1}, \tag{180}$$

where σ is a non-linear activation function and $\widetilde{\mathbf{H}}^{(1)} = \left(\mathbf{A}^T\mathbf{H}_k^{(1)}\mathbf{A}\right)_{1\leq k\leq K_1}$ is a compressed weight tensor of dimension $m \times m \times K_1$. The other hidden layers are composed of standard activation layers,

$$\mathbf{z}_n^{(\ell+1)} = \left(\sigma\left(\langle\mathbf{h}_k^{(\ell)}, \mathbf{z}_n^{(\ell)}\rangle + b_k^{(\ell)}\right)\right)_{1\leq k\leq K_\ell}, \qquad 2 \leq \ell \leq L-1, \tag{181}$$

with the output of the last hidden layer denoted by $\mathbf{z}_n^{(L)}$ and L being the total number of layers in the network [22, 25].

Each recovered input pattern $\hat{\mathbf{x}}_n$ with associated compressed pattern $\mathbf{y}_n$ has a label $d_n \in \mathcal{C}$ where $\mathcal{C}$ is the set containing all target type labels. With these definitions, the objective function $E_n(\mathbf{H}, \mathbf{b})$ can be defined as

$$E_n(\mathbf{H}, \mathbf{b}) = -\sum_{c\in\mathcal{C}}[d_n = c]\log\left(\frac{\exp\left(\langle\mathbf{h}_c^{(L)}, \mathbf{z}_n^{(L)}\rangle + b_c^{(L)}\right)}{\sum_{j\epsilon\mathcal{C}}\exp\left(\langle\mathbf{h}_j^{(L)}, \mathbf{z}_n^{(L)}\rangle + b_j^{(L)}\right)}\right), \tag{182}$$

where $[i = j]$ is the Iverson bracket notation for the Kronecker delta function.

Training is accomplished using gradient descent with gradients determined by the backpropagation learning algorithm. All layers will result in a weight matrix, with the exception of the first layer, which is

a compressed three-dimensional tensor. To define the gradient, we must first define the error-term [25] for internal layers as

$$\boldsymbol{\delta}^{(\ell)} = \left(\mathbf{H}^{(\ell)}\right)^T \boldsymbol{\delta}^{(\ell+1)} \circ \sigma'\left(\mathbf{H}^{(\ell)}\mathbf{z}^{(\ell)} + \mathbf{b}^{(\ell)}\right), \qquad 2 \le \ell \le L-1. \tag{183}$$

Now, the gradient for the first layer can be defined as

$$\nabla_{\widetilde{\mathbf{H}}^{(1)}} E_n(\mathbf{H}, \mathbf{b}) = \delta_k^{(2)} \mathbf{y}_n \mathbf{y}_n^T, \qquad 1 \le k \le K_1, \tag{184}$$

where $\delta_k^{(2)}$ is an element from the vector determined from (183). All other layers are standard fully-connected layers with gradients as defined in [22, 25].

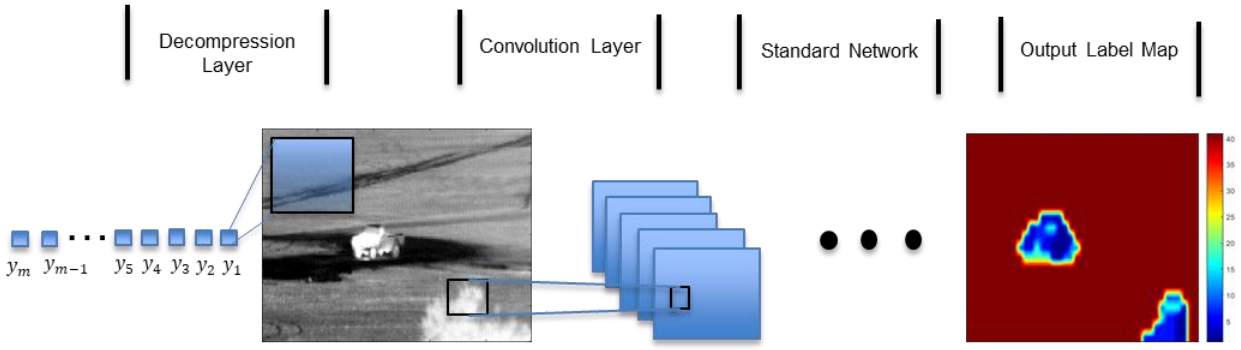


Figure 13 - QCFCNN with fixed-weight deconvolution layer. The decompression layer is used to create an estimate of the image patch before the QCFCNN is applied for target recognition and localization.

4.2.3 Compressed Target Recognition using Convolutional Neural Networks

The compressed QCFCNN can be used for target localization and identification for compressed measurements. By comparison, the MMCF presented in 3.1 can also be used to localize and identify targets in an image, but only after a full recovery of the measurements has been accomplished. The compressed QCFCNN will decompress compressed input patterns using a decompression layer and then perform target recognition and localization. The decompression is necessary for localization as most basis sets do not preserve spatial coherence.

To define the compressed target recognition for convolutional neural networks, we must first define a neighborhood for the set, S. The neighborhood for S is defined as

$$N(\mathbf{y}_{p,q}) := \{i \in [1,P], j \in [1,Q] |\ p-1 \leq i \leq p+1, q-1 \leq j \leq q+1 \text{ and } p,q \in S\}. \tag{185}$$

The convolutional neural network input will be an image area consisting of members from this neighborhood set.

A recovered neighborhood input sample for the convolutional neural network can be represented by $\widehat{\mathbf{X}}_{N(\mathbf{y}_{p,q})}$ and is defined as

$$\widehat{\mathbf{X}}_{N(\mathbf{y}_{p,q})} = \begin{bmatrix} \widehat{\mathbf{X}}_{p-1,q-1} & \widehat{\mathbf{X}}_{p-1,q} & \widehat{\mathbf{X}}_{p-1,q+1} \\ \widehat{\mathbf{X}}_{p,q-1} & \widehat{\mathbf{X}}_{p,q} & \widehat{\mathbf{X}}_{p,q+1} \\ \widehat{\mathbf{X}}_{p+1,q-1} & \widehat{\mathbf{X}}_{p+1,q} & \widehat{\mathbf{X}}_{p+1,q+1} \end{bmatrix} \tag{186}$$

in the spatial domain. We will define a vector $\hat{\mathbf{x}}_{p,q}$ as

$$\hat{\mathbf{x}}_{p,q} = \text{vec}(\widehat{\mathbf{X}}_{p,q}) = \boldsymbol{\Psi}\widehat{\boldsymbol{\Phi}}_T^{+}\mathbf{y}_{p,q} = \mathbf{A}\mathbf{y}_{p,q}, \tag{187}$$

where $\mathbf{y}_{p,q}$ of dimension m is the vector is compressed space and $\mathbf{A}$ is matrix that performs STLS decompression and provides a spatial domain representation. Each image patch in the neighborhood set has an associated compressed vector

$$\mathbf{y}_{N(\mathbf{y}_{p,q})} = \begin{bmatrix} \mathbf{y}_{p-1,q-1} & \mathbf{y}_{p-1,q} & \mathbf{y}_{p-1,q+1} \\ \mathbf{y}_{p,q-1} & \mathbf{y}_{p,q} & \mathbf{y}_{p,q+1} \\ \mathbf{y}_{p+1,q-1} & \mathbf{y}_{p+1,q} & \mathbf{y}_{p+1,q+1} \end{bmatrix}. \tag{188}$$

As mentioned, we can recreate $\widehat{\mathbf{X}}_{N(S)}$ of dimension $3t \times 3s$ containing sub-images of dimension $t \times s$ from $\mathbf{y}_{N(S)}$ of dimension $3m \times 3$ by using correlations. The matrix $\mathbf{A}$ is of dimension $ts \times m$ with column vectors $\mathbf{a}_i^T$ of length $ts \times 1$ can be represented as

$$\mathbf{A} = [\mathbf{a}_1^T \quad \mathbf{a}_2^T \quad \cdots \quad \mathbf{a}_{m-1}^T \quad \mathbf{a}_m^T]. \tag{189}$$

To define the decompression operation, we must first define a $\mathrm{reshape}()$ function to form a matrix from a vector

$$\mathrm{reshape}(\mathbf{a}^T) = \begin{bmatrix} a_1 & a_2 & \cdots & a_{t-1} & a_t \\ a_{t+1} & a_{t+2} & \dots & a_{2t-1} & a_{2t} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{t(s-2)+1} & a_{t(s-2)+2} & \cdots & a_{t(s-1)-1} & a_{t(s-1)} \\ a_{t(s-1)+1} & a_{t(s-1)+2} & \cdots & a_{ts-1} & a_{ts} \end{bmatrix}. \tag{190}$$

Using this definition, we can recover an image patch $\widehat{\mathbf{X}}_{p,q}$ by performing a correlation of the reshaped column vector with each element $y_{p,q}^i$ of vector $\mathbf{y}_{p,q}$

$$\widehat{\mathbf{X}}_{p,q} = \sum_{i=1}^{m} \mathrm{reshape}(\mathbf{a}_i^T) \otimes y_{p,q}^i. \tag{191}$$

Following this decompression layer, we define a quadratic correlation filter layer for automatic target recognition. To do so, we define a symmetric weight matrix

$$\mathbf{\Theta}_k^{(2)} = \mathbf{H}_k^{(2)} + \left(\mathbf{H}_k^{(2)}\right)^T \tag{192}$$

just as in Section 3.2.2. As before, we perform the eigendecomposition of the symmetric weight matrix

$$\mathbf{\Theta}_k^{(2)}\mathbf{V}_k = \mathbf{V}_k\mathbf{\Lambda}_k. \tag{193}$$

and compute the correlation

$$\mathbf{z}_n^{(3)} = \left(\sigma\left(\sum_i \lambda_i \left| \widehat{\mathbf{X}}_{N(\mathbf{y}_{p,q})}^{(n)} \otimes \mathbf{v}_{k,i} \right|^2 + b_k^{(2)} \right) \right)_{1 \le k \le K_2}, \tag{194}$$

where K_2 is the number of hidden nodes in the second layer. All other layers are as defined in Section 3.2.2. The gradient for the second layer is

$$\nabla_{\mathbf{H}^{(2)}} E_n(\mathbf{H}, \mathbf{b}) = \frac{\partial E_n}{\partial \mathbf{H}_k^{(2)}} = \frac{\partial E_n}{\partial \mathbf{\Theta}_k^{(2)}} \frac{\partial \mathbf{\Theta}_k^{(2)}}{\partial \mathbf{H}_k^{(2)}} = \sum_{l,m} \delta_k^{(3)} \mathrm{vec}\left(\widehat{\mathbf{X}}_{l,m}^{(n)}\right) \mathrm{vec}\left(\widehat{\mathbf{X}}_{l,m}^{(n)}\right)^T,$$

where the objective function as defined in (160).

4.3 Experiments

We conducted our experiments on a Dell® Precision T5610 with dual Intel® Xeon® E5-2680 @ 2.70 GHz with 64.0 GB of RAM installed and an Apple® Mac Pro (Early 2008) with dual Intel® Xeon® 5400 series @ 2.80 GHz series processors and 16.0GB RAM. For target detection we created a bounding box of 20×40 for each target type, resizing as necessary based on range to target to create the correlation matrices. For each target detection tile that meets the threshold criterion, we create a window of 60×120 around the detection for target recognition. To evaluate the QMLPNN, we assumed the target was centered in a 40×80 image patch. The QMLPNN can evaluate a larger image patch by using a sliding window approach. In QCFCNN, the entire window is provided to the network so that the network can identify and localize the target

We compared CQCF, WLS/QCF, STLS/QCF with some traditional decompression methods like basis pursuit [14], orthogonal matching pursuit [16] and initialized iterative reweighted least squares [15] paired with QCF to evaluate the performance of our compressed target detection method. In Figure 14, we note that STLS/QCF, WLS/QCF and CQCF have high accuracy even with few measurements whereas the performance degrades for BP/QCF, OMP/QCF and IIRLS/QCF when there are fewer measurements.

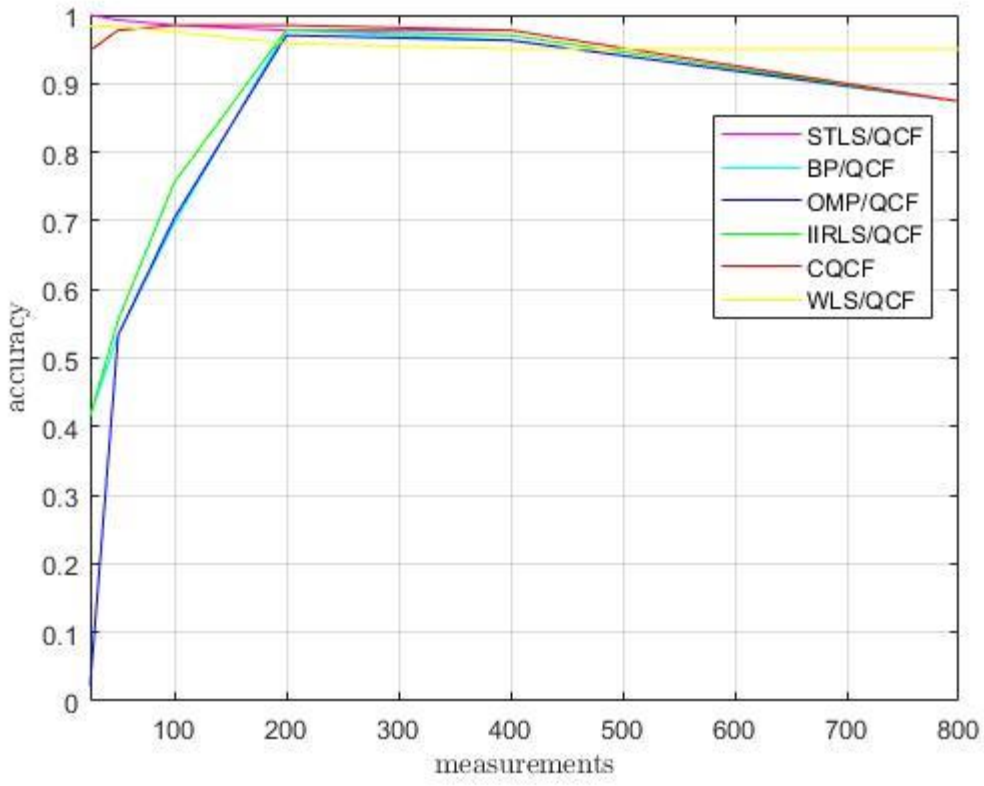


Figure 14 - Detection accuracy versus number of measurements. This chart compares STLS/QCF, BP/QCF, OMP/QCF, IIRLS/QCF, CQCF and WLS/QCF.

To evaluate the performance of our compressive classifier, we utilized the compressed and uncompressed QMLPNN. Figure 15 shows us that the unique patterns due to data lost from compression can be learned in network training giving almost perfect accuracy. However, if we train on uncompressed data and provide a compressed input or train on compressed data and provide uncompressed input, the performance degrades as the number of measurements decreases due to the loss of information. In Figure 16, we can see how classes are confused together as information is lost due to compression in the QMLPNN. We can see that similar target types and ones that appear together in the training data are misclassified as less measurements are used to represent the target.

Figure 17 shows a comparison of target recognition methods with varying levels of compression. The MLPNN and QMLPNN both improve as more measurements are added. These networks are trained on uncompressed samples and given compressed samples, as they both have decompression layers. The CNN and QCFCNN both perform very well for the compressed samples. In fact, for as little as 25 samples out of a possible 3200, they both have almost 70% recognition accuracy. Figure 18 shows some sample filters from the QCFCNN and the corresponding eigenvalue plots for each. We can see that the number of eigenvalues used to support the target and background can vary from filter to filter. Figure 19 shows the output of the QCFCNN for various levels of compression. As can be seen, we can accurately locate and identify the target with as little as 25 measurements. Figure 20 further demonstrates the invariance of the output of the quadratic correlation filter layer to the level of compression. This figure shows the output of the QCF layer for varying levels of compression and very little difference is seen in the correlation output even though there is significant change in the spatial detail.

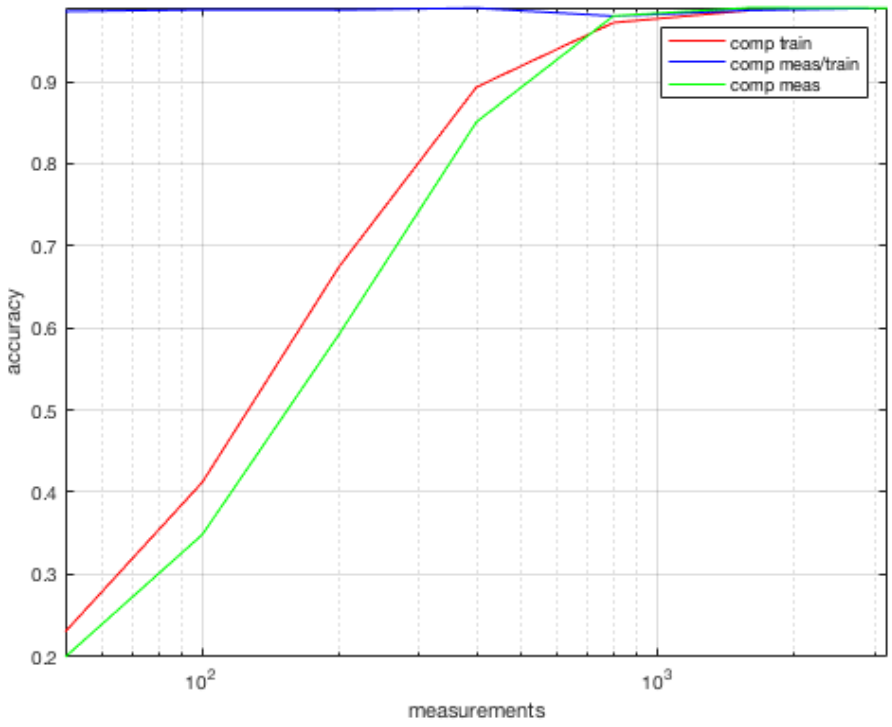


Figure 15 - Measurements vs. accuracy for QMLPNN network. Three cases are considered: (1) compressed training set, uncompressed inputs, (2) compressed training set, compressed inputs and (3) uncompressed training set, compressed inputs. In this chart, the accuracy is measured by the percentage of correctly labeled samples from the test set.

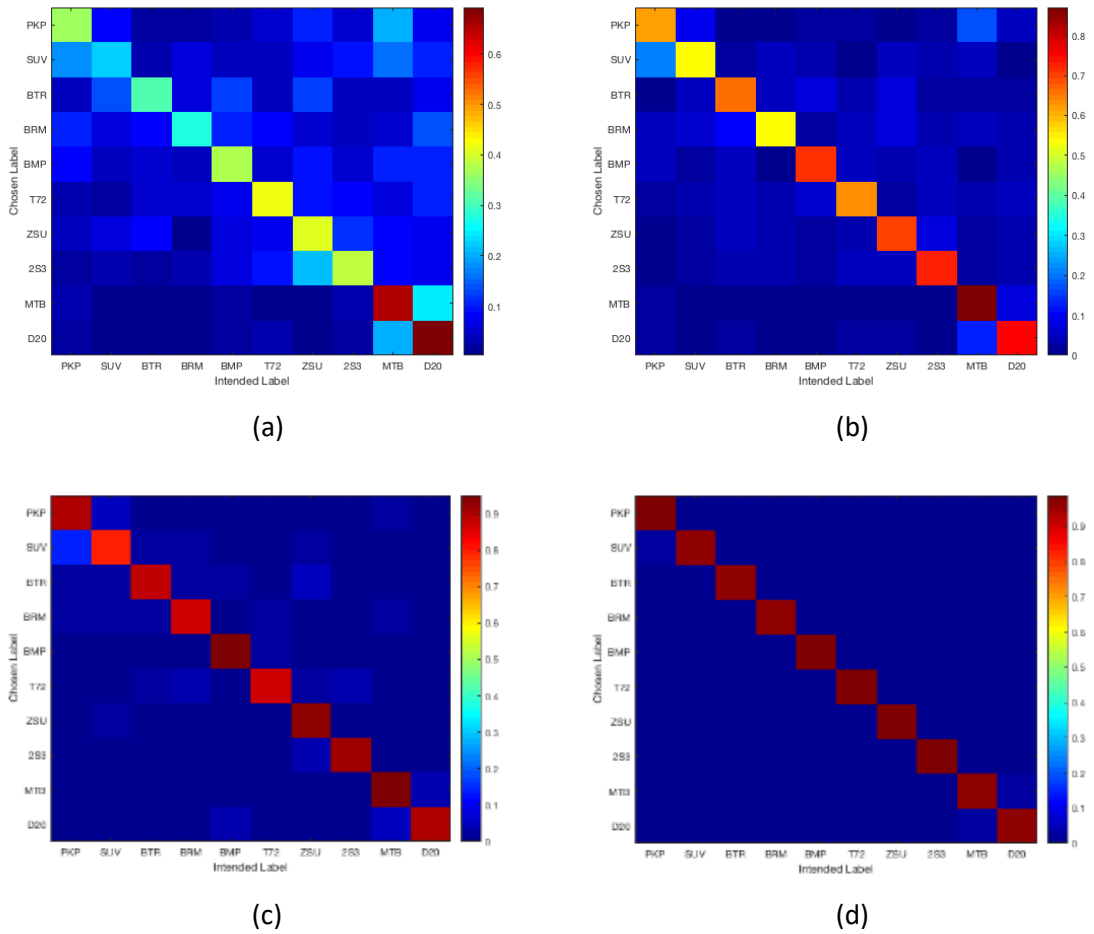


Figure 16 - Confusion matrices for QMLPNN with differing levels of compression : (a) 100, (b) 200, (c) 400 and (d) 800 measurements out of a total of 3200. We can see that as the number of measurements is decreased that similar classes (e.g. SUV and PKP) or classes that typically appear together in the training data (e.g. MTB and D20) are confused together. Using the compressed QMLPNN, we can see that it is not necessary to use more than 800 measurements to get accurate classification.

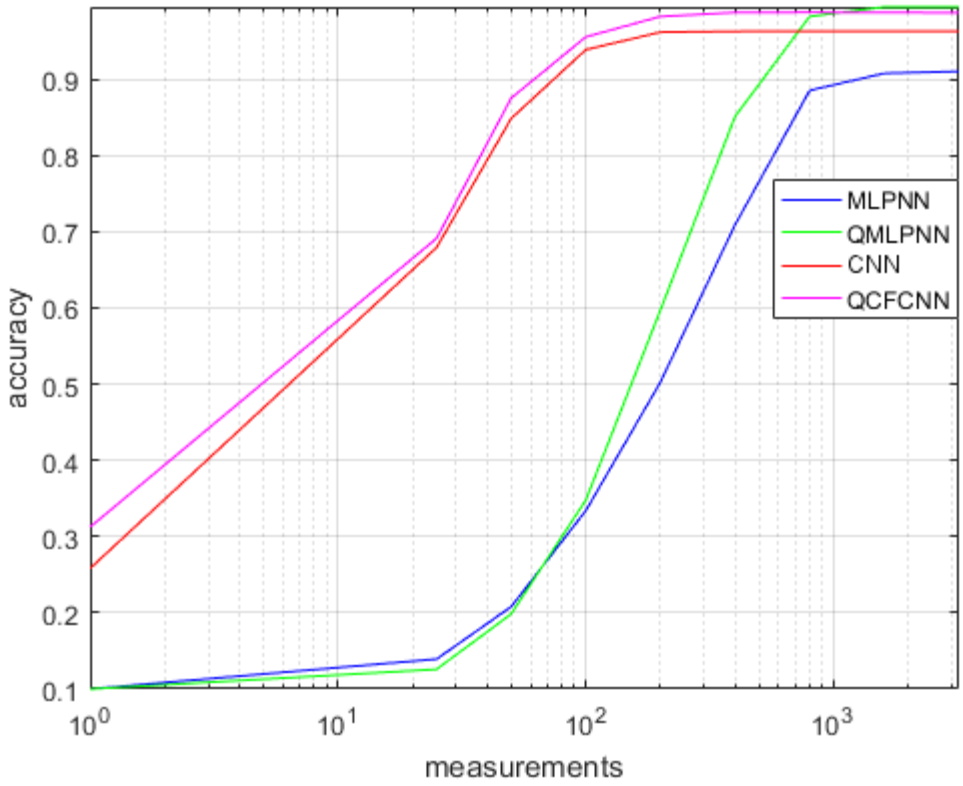


Figure 17 - Comparison of target recognition accuracy vs. compression level for methods presented. We note that the MLPNN (blue) achieves only about 90% accuracy with the maximum number of measurements. The QMLPNN (green) achieves close to 100% accuracy as the number of measurements is increased. The CNN (red) and QCFCNN (magenta) both perform well for all levels of compression.

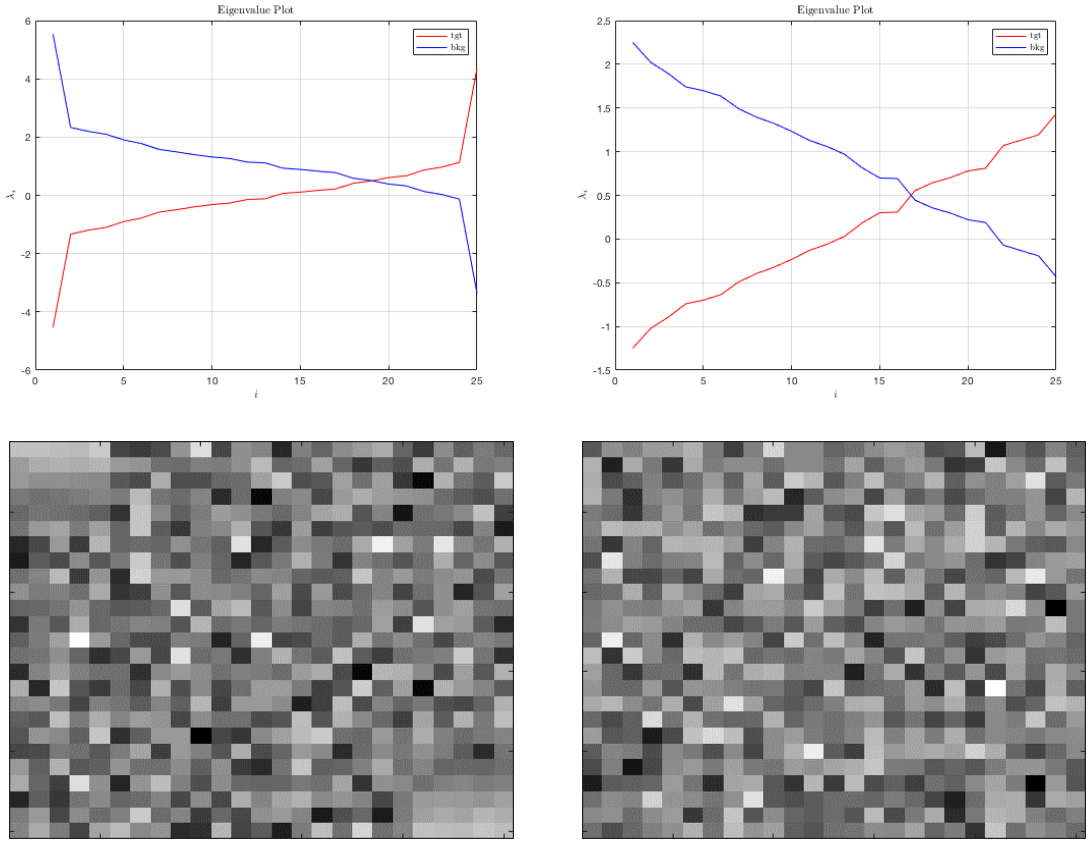


Figure 18 - Eigenvalue plot and mask plot for two different filters. For this network, the masks are 5×5 and the weight matrix from the quadratic correlation filter layer is 25×25 (containing 25 filters). From the eigenvalue plots, we note that the number of values that support the target and background for each filter can vary.

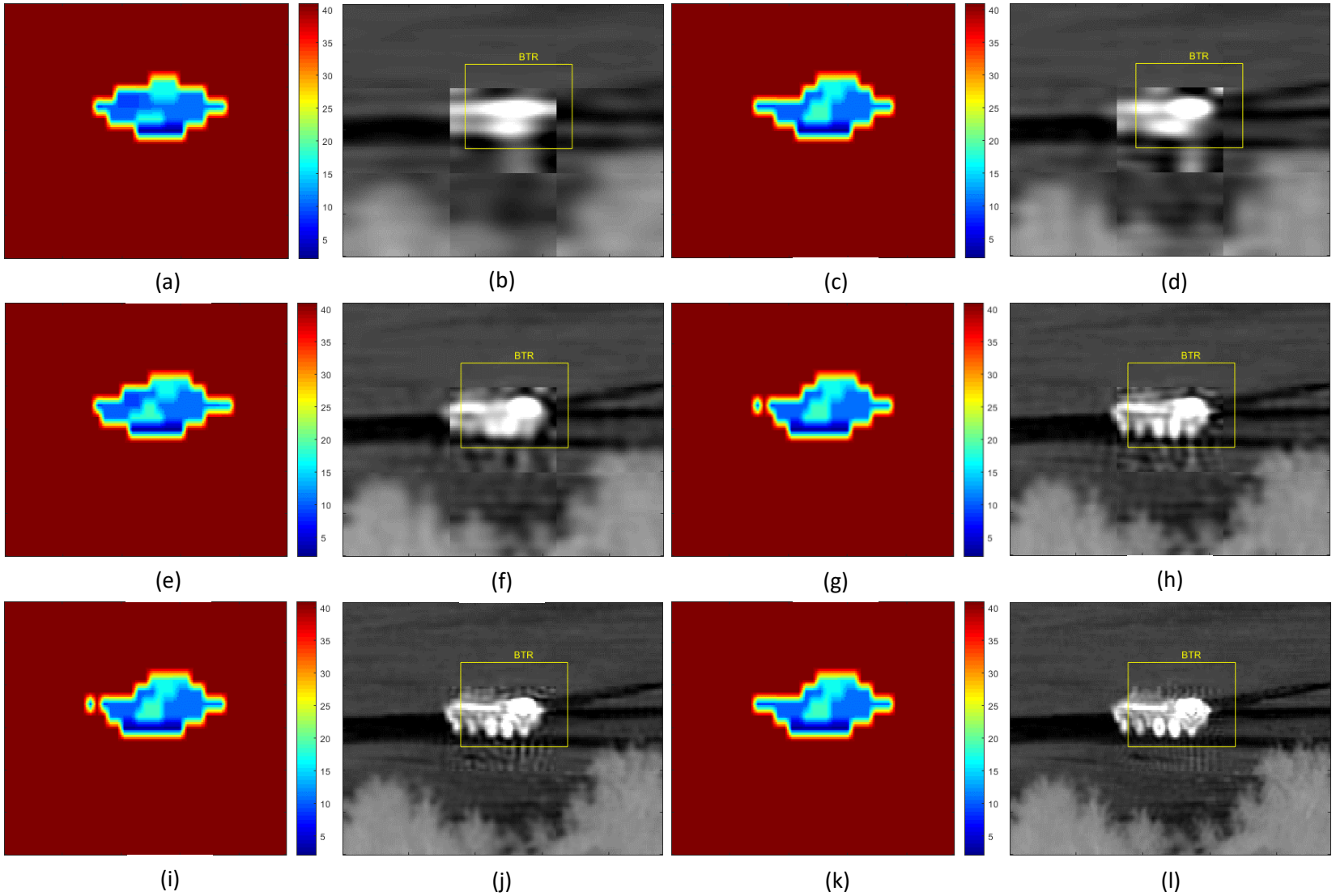


Figure 19 - QCFCNN applied to compressed data. Output label map and labeled decompressed image for (a)-(b) 25, (c)-(d) 50, (e)-(f) 100, (g)-(h) 200, (i)-(j) 400, and (k)-(l) 800 samples. We note that even if the target is not visibly discernable, the label map correctly identifies and locates the target for the cases presented.

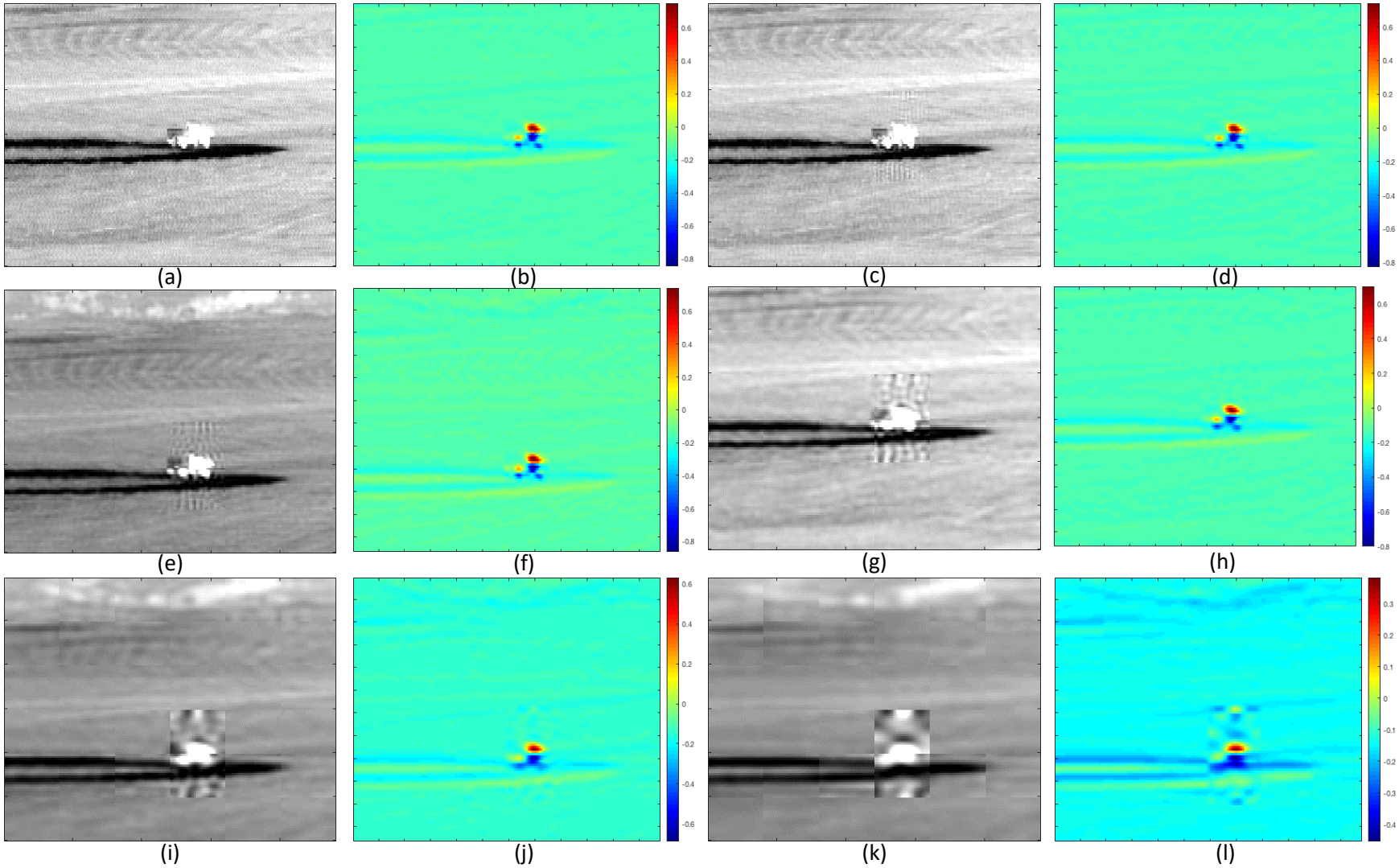


Figure 20 – Correlation layer output for varying levels of compression. Input image and correlation output for (a)-(b) uncompressed, (c)-(d) 400 samples, (e)-(f) 200 samples, (g)-(h) 100 samples, (i)-(j) 50 samples and (k)-(l) 25 samples. We note that the compression level does not significantly affect the output of the filter.

CHAPTER 5 – CONCLUSION

High Resolution MWIR sensors are not cost-effective for most applications, but with compressive sensing, a high resolution MWIR sensor can be realized using a high resolution spatial light modulator, such as a digital micromirror device, and a small MWIR focal plane array. For military applications, it is common to have automatic target recognition with these sensor systems. In this work, we propose an ATR system that is integrated with a linear decompression algorithm to increase the overall efficiency of the system.

5.1 Contributions

In this work, we proposed a target recognition algorithm for an MWIR compressive imager. While it is possible to perform decompression using non-linear decompression techniques with standard ATR algorithms, it may not fit into a constrained timeframe. By integrating the ATR algorithm with the decompression step, we are able to significantly decrease the amount of time required to process each frame of data. For instance, a 60 Hertz frame rate MWIR camera must process a frame of data every 16.7 milliseconds. Because of these hard constraints, it makes iterative non-linear approaches to decompression less desirable due to inaccuracy if it is terminated before convergence and uncertainty because of an unknown convergence time. By using a linear decoder, we eliminate the need for an iterative algorithm. Although there is a decrease in reconstruction accuracy, the goal is accurate target recognition and the loss of reconstruction accuracy is tolerable as long as ATR performance is not hindered.

The training time for a convolutional neural network can be excessive. However, inference can be very efficiently implemented when the weights are pre-determined. If we require a non-linear decoder before applying the CNN, the time to complete the operation would then become indefinite, which is why our

work combines a linear decoder with the convolutional neural network to create a target recognition system for a fixed timeframe system.

We proposed using the fast compressed target detection method in [18] to detect candidate areas for target recognition. By adjusting the detection threshold, we can control the sensitivity of the target detection on the compressed MWIR sensor. Without resampling, we take each candidate area and process it using our proposed quadratic correlation filter convolution neural network with a deconvolution layer added to process the compressed samples. This allows the QCFCNN to identify the target within the candidate area.

5.2 Conclusions

From our experiments, we can see that the level of compression does affect the target detection or recognition accuracy. However, there is a point where adding more measurements does not have a significant impact on accurate target detection or recognition and by using a linear decoder, we can provide a time deterministic solution so that the ATR algorithm can be run on a real-time system. We have also seen how neural networks can compensate for the compression of data by adapting the weights for the compressed pattern. This was seen in Figure 15 where the network accuracy was almost perfect for many different levels of compression given that the network was also trained with compressed samples. As seen in Figure 19, the level of compression has little impact on the output of the neural network. Given this, we believe the use of neural networks can have a significant impact on compressive sensor ATR systems.

Future Work

In our experiments, we typically used the DCT as the basis for our compression. This may not be optimal for sparse representation of the data. There are many other options for the dictionary including trained dictionaries, wavelet dictionaries and Karhunen-Loève Transform (KLT) to name a few. This suggests the exploration of how these other basis sets may impact target detection and recognition accuracy.

Another area of potential research is in the sensor design itself. By choosing an optimal basis set for the intended signals, there may also be an optimal SLM block to detector size that maximizes compression and target recognition accuracy.

A typical ATR system includes symbology on a cockpit display. One area of future work would be to find a method, which possibly uses the given compressed samples, to generate a spatial domain cockpit display so that the symbology can be overlayed. The closed-form reconstruction methods presented in this work may not suffice in this scenario where accuracy in the decompressed image is key.

We have tested our methods on a MWIR *ATR Algorithm Development Image Database*. But, there are many types of ATR sensor types that include short-wave infrared, long-wave infrared, SAR, LADAR and many others. Another area of further research would be to see how these methods could be applied to these other sensor types.

Our target recognition algorithms were primarily spatial. It might be useful to add temporal processing to the neural networks and see if the temporal characteristics of targets can assist in target recognition and localization.

APPENDIX A: COPYRIGHT PERMISSION LETTER

August 2017

PERMISSION TO USE COPYRIGHTED WORKS IN A PUBLICATION

Brian Millikan
2401 Rock Lane
Oviedo, FL 32765

19 August, 2017

Lockheed Martin Missiles & Fire Control – Orlando
5600 Sand Lake Road
Orlando, FL 32819

Dear Dr. Robert Muise:

I am a graduate student at the University of Central Florida. I am in the process of preparing a dissertation for publication and am seeking permission to include the following material in my publication. A copy of the work is enclosed.

The work will be used in the following manner: I will include this illustration as an example of a block-wise compressive sensing system in my dissertation publication. The publication information is as follows: Compressed Automatic Target Recognition Using A Compressive Infrared Imager, dissertation, Spring 2018, University of Central Florida.

Please indicate your approval of this request by signing the letter where indicated below and returning it to me as soon as possible. Your signing of this letter will also confirm that you own the copyright to the above-described material.

Very truly yours,

Brian Millikan
brian.millikan@knights.ucf.edu

For copyright owner use:

PERMISSION GRANTED FOR THE USE REQUESTED ABOVE:

By:

Title: Senior Staff Engineer

Date: 8-21-17

APPENDIX B: DERIVATIVE OF SOFTMAX OBJECTIVE FUNCTION

The softmax cross entropy objective function is equal to

$$E_n(\mathbf{H},\mathbf{b}) = -\sum_{c\in\mathcal{C}}[d_n = c]\log\left(\frac{\exp\left(\langle\mathbf{h}_c^{(L)},\mathbf{z}_n^{(L)}\rangle + b_c^{(L)}\right)}{\sum_{j\epsilon\mathcal{C}}\exp\left(\langle\mathbf{h}_j^{(L)},\mathbf{z}_n^{(L)}\rangle + b_j^{(L)}\right)}\right), \tag{B.1}$$

where the softmax is equal to

$$\mathrm{P}\left[d_n = c \mid \mathbf{z}_n^{(L)}; \mathbf{H}^{(L)}, \mathbf{b}^{(L)}\right] = \frac{\exp\left(\langle\mathbf{h}_c^{(L)},\mathbf{z}_n^{(L)}\rangle + b_c^{(L)}\right)}{\sum_{j\epsilon\mathcal{C}}\exp\left(\langle\mathbf{h}_j^{(L)},\mathbf{z}_n^{(L)}\rangle + b_j^{(L)}\right)}. \tag{B.2}$$

To find the derivative of the cross entropy function, we evaluate

$$\frac{\partial E_n(\mathbf{H},\mathbf{b})}{\partial\mathbf{h}_c^{(L)}} = \frac{\partial}{\partial\mathbf{h}_c^{(L)}}\left[-\sum_{c\in\mathcal{C}}[d_n = c]\log\left(\frac{\exp\left(\langle\mathbf{h}_c^{(L)},\mathbf{z}_n^{(L)}\rangle + b_c^{(L)}\right)}{\sum_{j\epsilon\mathcal{C}}\exp\left(\langle\mathbf{h}_j^{(L)},\mathbf{z}_n^{(L)}\rangle + b_j^{(L)}\right)}\right)\right], \tag{B.3}$$

where

$$[d_n = c] = \begin{cases} 0 & d_n \neq c \\ 1 & d_n = c \end{cases}, \tag{B.4}$$

which gives

$$\frac{\partial E_n(\mathbf{H},\mathbf{b})}{\partial\mathbf{h}_c^{(L)}} = -\frac{\partial}{\partial\mathbf{h}_c^{(L)}}\left[\log\left(\frac{\exp\left(\langle\mathbf{h}_c^{(L)},\mathbf{z}_n^{(L)}\rangle + \mathbf{b}_c^{(L)}\right)}{\sum_{j\epsilon\mathcal{C}}\exp\left(\langle\mathbf{h}_j^{(L)},\mathbf{z}_n^{(L)}\rangle + \mathbf{b}_j^{(L)}\right)}\right),\right]. \tag{B.5}$$

Since $\frac{d}{dx}\log(u) = \frac{1}{u}\frac{du}{dx}$, we have

$$= -\frac{\sum_{j\epsilon\mathcal{C}}\exp\left(\langle\mathbf{h}_j^{(L)},\mathbf{z}_n^{(L)}\rangle + \mathbf{b}_j^{(L)}\right)}{\exp\left(\langle\mathbf{h}_c^{(L)},\mathbf{z}_n^{(L)}\rangle + \mathbf{b}_c^{(L)}\right)}\frac{\partial}{\partial\mathbf{h}_c^{(L)}}\left[\frac{\exp\left(\langle\mathbf{h}_c^{(L)},\mathbf{z}_n^{(L)}\rangle + \mathbf{b}_c^{(L)}\right)}{\sum_{j\epsilon\mathcal{C}}\exp\left(\langle\mathbf{h}_j^{(L)},\mathbf{z}_n^{(L)}\rangle + \mathbf{b}_j^{(L)}\right)}\right]. \tag{B.6}$$

Looking at the second part of this, we can see that this is just the derivative of a quotient which is $\frac{d}{dx}\frac{u}{v} = \frac{v\frac{du}{dx} - u\frac{dv}{dx}}{v^2}$ and given $\frac{\mathrm{d}}{\mathrm{dx}}\exp(u) = \exp(u)\frac{du}{dx}$ and letting $v = \exp\left(\sum_{j\epsilon\mathcal{C}}\exp\left(\langle\mathbf{h}_j^{(L)},\mathbf{z}_n^{(L)}\rangle + \mathbf{b}_j^{(L)}\right)\right)$ and $u = \exp\left(\langle\mathbf{h}_c^{(L)},\mathbf{z}_n^{(L)}\rangle + \mathbf{b}_c^{(L)}\right)$ we have

$$\frac{\partial}{\partial \mathbf{h}_c^{(L)}}\left[\frac{u}{v}\right] = \frac{vu\mathbf{z}_n^{(L)} - u^2\mathbf{z}_n^{(L)}}{v^2} \tag{B.7}$$

which gives

$$\frac{\partial E_n(\mathbf{H},\mathbf{b})}{\partial \mathbf{h}_c^{(L)}} = -\frac{v}{u}\left[\frac{vu\mathbf{z}_n^{(L)} - u^2\mathbf{z}_n^{(L)}}{v^2}\right]. \tag{B.8}$$

This can further be reduced to

$$\frac{\partial E_n(\mathbf{H},\mathbf{b})}{\partial \mathbf{h}_c^{(L)}} = -\mathbf{z}_n^{(L)}\left[1 - \frac{u}{v}\right] = -\mathbf{z}_n^{(L)}\left[1 - \mathrm{P}\left[d_n = c \mid \mathbf{z}_n^{(L)}; \mathbf{H}^{(L)}, \mathbf{b}^{(L)}\right]\right] \tag{B.9}$$

where

$$\mathrm{P}\left[d_n = c \mid \mathbf{z}_n^{(L)}; \mathbf{H}^{(L)}, \mathbf{b}^{(L)}\right] = \frac{\exp\left(\langle \mathbf{h}_c^{(L)}, \mathbf{z}_n^{(L)}\rangle + b_c^{(L)}\right)}{\sum_{j\epsilon\mathcal{C}} \exp\left(\langle \mathbf{h}_j^{(L)}, \mathbf{z}_n^{(L)}\rangle + b_j^{(L)}\right)} \tag{B.10}$$

is the softmax function. However, (B.9) is true only if $j = c$, otherwise we have

$$\frac{\partial}{\partial \mathbf{h}_j^{(L)}}\left[\frac{u}{v}\right] = \frac{-u^2\mathbf{z}_n^{(L)}}{v^2}, \tag{B.11}$$

for $j \neq c$ giving the partial derivative of the cross entropy function

$$\frac{\partial E_n(\mathbf{H},\mathbf{b})}{\partial \mathbf{h}_j^{(L)}} = -\mathbf{z}_n^{(L)}\left[-\frac{u}{v}\right] = \mathbf{z}_n^{(L)}\mathrm{P}\left[d_n = c \mid \mathbf{z}_n^{(L)}; \mathbf{H}^{(L)}, \mathbf{b}^{(L)}\right]. \tag{B.12}$$

Or, we can write this compactly as

$$\frac{\partial E_n(\mathbf{H},\mathbf{b})}{\partial \mathbf{h}_j^{(L)}} = -\mathbf{z}_n^{(L)}\left[[y_n = c] - \mathrm{P}\left[d_n = c \mid \mathbf{z}_n^{(L)}; \mathbf{H}^{(L)}, \mathbf{b}^{(L)}\right]\right]. \tag{B.13}$$

LIST OF REFERENCES

[1] D. E. Dudgeon and R. T. Lacoss, "An Overview of Automatic Target Recognition," 1993.

[2] T. M. Bayik, "Automatic target recognition in infrared imagery," Middle East Technical University Press, Ankara, 2004.

[3] M. W. Roth, "Survey of neural network technology for automatic target recognition," *IEEE Transactions on neural networks,* vol. 1, no. 1, pp. 28-43, 1990.

[4] B. J. Schachter, *Automatic target recognition*. International Society for Optics and Photonics, 2016.

[5] G. Koretsky, J. Nicoll, and M. Taylor, "A tutorial on electro-optical/infrared (EO/IR) theory and systems," INSTITUTE FOR DEFENSE ANALYSES ALEXANDRIA VA, 2013.

[6] V. C. Coffey, "Seeing in the dark: Defense applications of IR imaging," *Optics and Photonics News,* vol. 22, no. 4, pp. 26-31, 2011.

[7] M. F. Duarte *et al.*, "Single-pixel imaging via compressive sampling," *IEEE signal processing magazine,* vol. 25, no. 2, pp. 83-91, 2008.

[8] E. J. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on information theory,* vol. 52, no. 2, pp. 489-509, 2006.

[9] E. J. Candès and T. Tao, "Near-optimal signal recovery from random projections: Universal encoding strategies?," *IEEE transactions on information theory,* vol. 52, no. 12, pp. 5406-5425, 2006.

[10] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling," *IEEE signal processing magazine,* vol. 25, no. 2, pp. 21-30, 2008.

[11] D. L. Donoho, "Compressed sensing," *IEEE Transactions on information theory,* vol. 52, no. 4, pp. 1289-1306, 2006.

[12] S. Foucart and H. Rauhut, *A mathematical introduction to compressive sensing* (no. 3). Basel: Birkhäuser, 2013.

[13] A. Mahalanobis, R. Shilling, R. Murphy, and R. Muise, "Recent results of medium wave infrared compressive sensing," *Applied optics,* vol. 53, no. 34, pp. 8060-8070, 2014.

[14] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM review,* vol. 43, no. 1, pp. 129-159, 2001.

[15] B. Millikan, A. Dutta, N. Rahnavard, Q. Sun, and H. Foroosh, "Initialized iterative reweighted least squares for automatic target recognition," in *Military Communications Conference, MILCOM 2015-2015 IEEE*, 2015, pp. 506-510: IEEE.

[16] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Transactions on information theory,* vol. 53, no. 12, pp. 4655-4666, 2007.

[17] L. Baldassarre, Y.-H. Li, J. Scarlett, B. Gözcü, I. Bogunovic, and V. Cevher, "Learning-based compressive subsampling," *IEEE Journal of Selected Topics in Signal Processing,* vol. 10, no. 4, pp. 809-822, 2016.

[18] B. Millikan, A. Dutta, Q. Sun, and H. Foroosh, "Fast Detection of Compressively-Sensed IR Targets Using Stochastically Trained Least Squares and Compressed Quadratic Correlation Filters," *IEEE Transactions on Aerospace and Electronic Systems,* 2017.

[19] A. Rodriguez, V. N. Boddeti, B. V. Kumar, and A. Mahalanobis, "Maximum margin correlation filter: A new approach for localization and classification," *IEEE Transactions on Image Processing,* vol. 22, no. 2, pp. 631-643, 2013.

[20] A. F. Rodriguez-Perez, "Maximum Margin Correlation Filters," 137, 2012.

[21] A. Mahalanobis, R. R. Muise, S. R. Stanfill, and A. Van Nevel, "Design and application of quadratic correlation filters for target detection," *IEEE Transactions on Aerospace and Electronic Systems,* vol. 40, no. 3, pp. 837-850, 2004.

[22] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., 2006.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097-1105.

[24] SENSIAC. *ATR Algorithm Development Image Database*. Available: http://www.sensiac.org/external/products/list_databases.jsf

[25] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature,* vol. 521, no. 7553, pp. 436-444, 2015.

[26] J. Bouvrie, "Notes on convolutional neural networks," 2006.

[27] A. Antoniou, *Digital Signal Processing*. Mcgraw-Hill, 2016, p. 800.

[28] J. G. Proakis and D. K. Manolakis, *Digital Signal Processing (4th Edition)*. Prentice-Hall, Inc., 2006.

[29] E. J. Candès, "Lectures on compressive sampling and frontiers in signal processing," *The Institute for Mathematics and its Applications,* pp. 2006-2007, 2007.

[30] T. Blumensath and M. E. Davies, "Iterative hard thresholding for compressed sensing," *Applied and Computational Harmonic Analysis,* vol. 27, no. 3, pp. 265-274, 2009/11/01/ 2009.

[31] D. Needell and J. A. Tropp, "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples," *Applied and Computational Harmonic Analysis,* vol. 26, no. 3, pp. 301-321, 2009/05/01/ 2009.

[32] P. Boufounos, M. F. Duarte, and R. G. Baraniuk, "Sparse signal reconstruction from noisy compressive measurements using cross validation," in *Statistical Signal Processing, 2007. SSP'07. IEEE/SP 14th Workshop on*, 2007, pp. 299-303: IEEE.

[33] A. Mahalanobis and R. Muise, "Object specific image reconstruction using a compressive sensing architecture for application in surveillance systems," *IEEE transactions on aerospace and electronic systems,* vol. 45, no. 3, 2009.

[34] M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*, 1 ed. New York: Springer-Verlag New York, 2010.

[35] E. Y. Lam and J. W. Goodman, "A mathematical analysis of the DCT coefficient distributions for images," *IEEE Transactions on image processing,* vol. 9, no. 10, pp. 1661-1666, 2000.

[36] K. Fukunaga and W. L. Koontz, "Application of the Karhunen-Loeve expansion to feature selection and ordering," *IEEE Transactions on computers,* vol. 100, no. 4, pp. 311-318, 1970.

[37] J. Xu, X. Zhang, and Y. Li, "Kernel neuron and its training algorithm," in *Proceedings of 8th International Conference on Neural Information Processing*, 2001, vol. 2, pp. 861-866.

[38] B. V. Kumar, A. Mahalanobis, and R. D. Juday, *Correlation pattern recognition*. Cambridge University Press, 2005.

[39] S. R. F. Sims and A. Mahalanobis, "Performance evaluation of quadratic correlation filters for target detection and discrimination in infrared imagery," *Optical Engineering,* vol. 43, no. 8, pp. 1705-1711, 2004.

[40] N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[41] Y. LeCun *et al.*, "Backpropagation applied to handwritten zip code recognition," *Neural computation,* vol. 1, no. 4, pp. 541-551, 1989.

[42] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE,* vol. 86, no. 11, pp. 2278-2324, 1998.

[43] B. Millikan, Q. Sun, and H. Foroosh, "Deep Convolutional Neural Networks with Integrated Quadratic Correlation Filters for Automatic Target Recognition," 2018.

[44] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431-3440.

[45] C. Christodoulou and M. Georgiopoulos, *Applications of Neural Networks in Electromagnetics*. Artech House, Inc., 2000, p. 530.

[46] A. Ng, "Deep Learning Tutorial," *URL http://ufldl.stanford.edu/tutorial/,* 2015.

[47] S. Theodoridis and K. Koutroumbas, *Pattern Recognition, Fourth Edition*. Academic Press, 2008, p. 900.

[48] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Is object localization for free?-weakly-supervised learning with convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 685-694.

[49] J. Li, S. Kim, and C.-C. J. Kuo, "Focus of attention (FOA) identification from compressed video for automatic target recognition (ATR)," in *Image Processing, 1995. Proceedings., International Conference on*, 1995, vol. 3, pp. 508-511: IEEE.

[50] C. E. Daniell, "Object recognition in compressed imagery," California Institute of Technology, 2000.

[51] W. B. Seales, C. J. Yuan, W. Hu, and M. D. Cutts, "Object recognition in compressed imagery," *Image and Vision Computing,* vol. 16, no. 5, pp. 337-352, 1998.

[52] C. Daniell, A. Mahalanobis, and R. Goodman, "Object recognition in subband transform-compressed images by use of correlation filters," *Applied optics,* vol. 42, no. 32, pp. 6474-6487, 2003.

[53] M. A. Davenport *et al.*, "The smashed filter for compressive classification and target recognition," *Computational Imaging V at SPIE Electronic Imaging,* 2007.

[54] M. A. Davenport, M. B. Wakin, and R. G. Baraniuk, "Detection and estimation with compressive measurements," *Dept. of ECE, Rice University, Tech. Rep,* 2006.

[55] M. F. Duarte, M. A. Davenport, M. B. Wakin, and R. G. Baraniuk, "Sparse signal detection from incoherent projections," in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, 2006, vol. 3, pp. III-III: IEEE.

[56] J. Haupt, R. Castro, R. Nowak, G. Fudge, and A. Yeh, "Compressive sampling for signal classification," in *Signals, Systems and Computers, 2006. ACSSC'06. Fortieth Asilomar Conference on*, 2006, pp. 1430-1434: IEEE.

[57] D. Waagen, N. Shah, M. Ordaz, and M. Cassabaum, "Random subspaces and SAR classification efficacy," in *Proc. SPIE*, 2005, vol. 5808, pp. 257-268.