NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

APPLYING DEEP LEARNING METHODS TO IDENTIFY TARGETS IN SYNTHETIC APERTURE RADAR IMAGES

by

Serkan Aktas

December 2020

Thesis Advisor: David A. Garren
Co-Advisor: John D. Roth

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 2020	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE
APPLYING DEEP LEARNING METHODS TO IDENTIFY TARGETS IN SYNTHETIC APERTURE RADAR IMAGES

5. FUNDING NUMBERS

6. AUTHOR(S) Serkan Aktas

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release. Distribution is unlimited.

12b. DISTRIBUTION CODE
A

13. ABSTRACT (maximum 200 words)

Synthetic aperture radar (SAR) provides high-resolution imagery and can operate in the day and at night and in every weather condition. SAR has been used for military reconnaissance and surveillance. Examining SAR images manually, however, is challenging even for a specialist, since it is difficult to find high-value targets in a wide area of SAR images. This is especially true when time is critical for operations. Thus, an efficient, reliable method to analyze SAR images automatically is needed. To solve this problem, deep learning (DL) methods are developed for automatic target recognition (ATR). A convolutional neural network (CNN) is a deep-learning algorithm made up of several processing layers for target recognition and classification. One of the challenges in developing and testing a CNN algorithm is to find relevant datasets. The dataset used in this thesis comes from the Moving and Stationary Target Acquisition and Recognition program (MSTAR).

In this research, the SAR ATR concept and performance are analyzed using several CNN DL architectures. Specifically, this investigation examines the effects of a few variable parameters within CNN DL architectures to gain insight into optimal strategies for using these architectures. Using CNN structures with different numbers of layers, it was possible to classify SAR targets successfully and automatically with state-of-the-art accuracy. This method proved useful for classification and recognition of military targets.

14. SUBJECT TERMS
synthetic aperture radar, SAR, convolutional neural network, CNN, deep learning, DL, automatic target recognition, ATR

15. NUMBER OF PAGES
79

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

APPLYING DEEP LEARNING METHODS TO IDENTIFY TARGETS IN SYNTHETIC APERTURE RADAR IMAGES

Serkan Aktas
Captain, Turkish Air Force
BEE, Turkish Air Force Academy , 2010

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF SCIENCE IN ENGINEERING SCIENCE
(ELECTRICAL ENGINEERING)**

from the

**NAVAL POSTGRADUATE SCHOOL
December 2020**

Approved by: David A. Garren
Advisor

John D. Roth
Co-Advisor

Douglas J. Fouts
Chair, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Synthetic aperture radar (SAR) provides high-resolution imagery and can operate in the day and at night and in every weather condition. SAR has been used for military reconnaissance and surveillance. Examining SAR images manually, however, is challenging even for a specialist, since it is difficult to find high-value targets in a wide area of SAR images. This is especially true when time is critical for operations. Thus, an efficient, reliable method to analyze SAR images automatically is needed. To solve this problem, deep learning (DL) methods are developed for automatic target recognition (ATR). A convolutional neural network (CNN) is a deep-learning algorithm made up of several processing layers for target recognition and classification. One of the challenges in developing and testing a CNN algorithm is to find relevant datasets. The dataset used in this thesis comes from the Moving and Stationary Target Acquisition and Recognition program (MSTAR).

In this research, the SAR ATR concept and performance are analyzed using several CNN DL architectures. Specifically, this investigation examines the effects of a few variable parameters within CNN DL architectures to gain insight into optimal strategies for using these architectures. Using CNN structures with different numbers of layers, it was possible to classify SAR targets successfully and automatically with state-of-the-art accuracy. This method proved useful for classification and recognition of military targets.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

ATR	Automatic Target Recognition
CNN	Convolutional Neural Network
DL	Deep Learning
HH	Horizontal-Horizontal
HV	Horizontal-Vertical
GPU	Graphics Processing Unit
ISAR	Inverse Synthetic Aperture Radar
ML	Machine Learning
MSTAR	The Moving and Stationary Target Acquisition and Recognition
RAR	Real Aperture Radar
ReLU	Rectified Linear Unit
SAR	Synthetic Aperture Radar
SGDM	Stochastic Gradient Descent with Momentum
VH	Vertical-Horizontal
VV	Vertical-Vertical

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

First, I would like to thank my thesis advisor, Prof. Garren, and co-advisor, Prof. Roth, for their help, guidance, and insight throughout the thesis. It is my privilege and honor to work with you.

I would like to thank my family for their endless support, love, and faith in me throughout my life. Having such a great family has been my biggest luck.

Finally, I want to thank my fiancé, the light of my life, Birce Elif, for being in my life. Without your smile and support, I do not believe I would have finished my degree.

THIS PAGE INTENTIONALLY LEFT BLANK

DISCLAIMER

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Republic of Turkey, Turkish Armed Forces, or the Turkish Air Force.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Operational synthetic aperture radar, or SAR, was developed in 1951, and the first images from this technology were generated in 1953 [1]. Since then, SAR has been used widely, not only for military applications, but also for civilian purposes like finding water sources and controlling vegetation, and in agriculture. SAR can operate in every weather state and provides high-resolution images. Yet, analyzing SAR images manually is difficult and time consuming. In particular, it is difficult to find or recognize high value targets within a multitude of SAR images, even for a specialist. Therefore, it is more efficient to process SAR imagery automatically with an algorithm. In particular, the interest in using automatic target recognition (ATR) applied to SAR images for military applications has increased recently [2], [3].

Enabling such applications is deep learning (DL), a type of the machine learning (ML) method. DL has drawn the attention of scholars since it requires less image processing, is easier to implement, and gives increased recognition accuracy when compared with other machine learning algorithms [3]. DL has been used in different areas such as image classification, movement tracking, and text detection, etc. Among DL methods such as auto encoder, deep belief networks, and restricted Boltzmann machine, convolutional neural networks (CNN) have presented especially outstanding performance in image recognition and classification tasks [4].

CNN is a type of DL algorithm that decreases the complexity of the image processing. Besides image classification and recognition, CNN has been widely used in various applications such as sound classification, road sign recognition, biomedicine, human motion recognition, and so on, due to its robustness [5]–[9]. Also, it gives users the flexibility to modify the number of layers, the pooling method, and the filter weight used in processing. Although it can be modified to have different structures, most often CNN consists of an input layer, convolution layers, subsampling layers, fully connected layers, and lastly an output layer [10]. The number of layers and methods like pooling, filter sizes, or normalization units vary in every CNN structure. There are some well-known and extensively used CNN structures like Alexnet, ZFNet, VGGNet, LeNet, and GoogleNet.

One of the most important aspects that affects performance of the CNN is the amount of data. Just like networks, there are well-known datasets that provide thousands—or even millions—of images for CNN users. Among these datasets are MNIST, CIFAR-10, and LabelMe. One particular dataset, called ImageNet, has up to 15 million images in approximately 22,000 categories [11].

While there are datasets that provide non-military images in both color and gray scale, it is difficult to find datasets that consist of SAR images of military targets. The Moving and Stationary Target Acquisition and Recognition (MSTAR) program provides public data, including various SAR images of military targets like tanks, trucks, and bulldozers from different angles. This dataset was created in the 1990s using an X band sensor in 1-foot resolution [12].

For training a neural network, it is better to have a large amount of data in order to get higher accuracy levels. A lack of datasets is one of the biggest problems for training neural networks that aim to classify or recognize an image dataset. Although SAR images are widely used, as previously mentioned, it is challenging to find a public dataset containing SAR images for CNNs for military purposes. Therefore, some improvements have been attempted to augment the quality of images. In [13], the authors removed the background of images before training the CNN for facial recognition. According to the authors, when the signal-to-noise ratio is high (i.e., when the object of interest covers most of the image), the background of the image does not significantly affect the training process, but if the signal-to-noise ratio is low (i.e., when the background of the image forms more than half of the image), then the background can affect the training process.

The main purpose of this study is to provide an initial examination of basic CNN performance with regards to some basic parameters of the CNN, such as the number of layers. Also, this study looks at the effects of the background surrounding targets through masking.

A. PURPOSE OF THESIS

The main aim of this thesis is to build a CNN that can identify and classify SAR images with a state-of-the-art accuracy level. In order to reach this level of accuracy,

different CNN structures with changing numbers of layers were created, and these structures were compared with each other. The secondary purpose of this study is to investigate the effect of the training dataset on CNN. The backgrounds used in this thesis cover most of the images, and thus, can affect the training process. In an attempt to increase the accuracy level of the developed CNN identification and classification process, partial masking of the image background is used, unlike in [13], where the background is completely darkened.

Target images are taken from the MSTAR public dataset. Targets in this dataset are separated into two groups. The first group of data, which has more training data, consists of a mix of eight targets such as tanks, trucks, and military vehicles. Since this dataset has more training data, the thesis tries to achieve a state-of-the-art accuracy level. The second group of data, which has less training data, consists of variants of a type of tank. With this group, the effect of the size of the training dataset on accuracy is investigated. Lastly, to increase the accuracy with the smaller dataset, masking of the background of the image is examined at the end of the simulation. The MATLAB deep learning toolbox is used for the simulation during this study.

B. THESIS ORGANIZATION

As previously mentioned, this thesis aims to develop a CNN algorithm to identify and recognize SAR image targets automatically. Thus, this thesis is organized based on the research components: the development of the SAR algorithm, the construction of the CNN, the simulation, and results. Building on the brief explanation of SAR and CNN in this chapter, Chapter II provides details about SAR imaging. The chapter also includes methods and algorithms of SAR imaging, features of SAR images like resolution, frequency band, polarization, and their effects on an image. Chapter III delves into the CNN. Layers that form the CNN are covered in this chapter along with their mathematical concepts, which are input, convolution, sampling, and the fully connected and output layers. The methodology and design of the CNN and the dataset used in this thesis are analyzed in Chapter IV. Simulation results and their comparison are discussed in Chapter V. Finally,

an examination and summary of the results in comparison to other work, conclusions, and suggestions for future work are presented in Chapter VI.

II. SYNTHETIC APERTURE RADAR

Synthetic aperture radar was invented by Carl Wiley in 1951, and the first SAR images were taken in 1953 [1]. Since then, SAR has become an important tool for taking high-resolution images and has been widely used in environmental applications, like monitoring climate change and locating water sources, and for military applications. This technology provides high-resolution images of the Earth's surface and objects. Unlike infrared (IR) or optical imaging systems, it is not affected by fog, clouds, or dust. It can be installed on spaceborne or airborne platforms. Because it can operate in all weather conditions, day and night, SAR is widely used¾especially in military applications for reconnaissance and automatic target recognition.

Radar is a remote sensing system that is used to produce an image of a target. The main purpose of imaging from air or space platforms is to acquire high-resolution images of the Earth. Azimuth and range resolution help to determine the quality of the images. Real aperture radars (RAR) are typically installed on aircraft to have better range resolution. If the platform has a longer antenna, it can get finer azimuth resolution, but the size of the antenna cannot be increased too much since there is a space limitation in the aircraft. Airborne or spaceborne platforms can only carry radars of limited sizes. An airborne platform can carry up to two meters of antenna while spaceborne platforms can carry up to 15 meters of antenna. Synthetic aperture radar was developed to get around this limitation of the RAR. SAR systems make it possible to get finer azimuth resolution with a small antenna. SAR creates a synthetically bigger aperture using signal processing techniques without requiring bigger antenna sizes [14].

The main difference between SAR and RAR is the azimuth resolution. The azimuth resolution in the RAR is determined by aperture diameter. Thus, resolution is proportional with the range between the radar and the target. On the other hand, in SAR systems, much bigger aperture than RAR can be synthesized with signal processing. In SAR systems azimuth resolution is independent of the range between the radar and the target.

A basic operating principle of SAR can be seen in Figure 1. As seen in the figure, the platform moves through the direction of flight with a known direction, speed, and altitude. The SAR aperture gathers and sends pulses and reflected signals, and parameters such as speed, direction, angle, and height are calculated by signal processors. Then the same process repeats until the area selected is completely imaged [15].

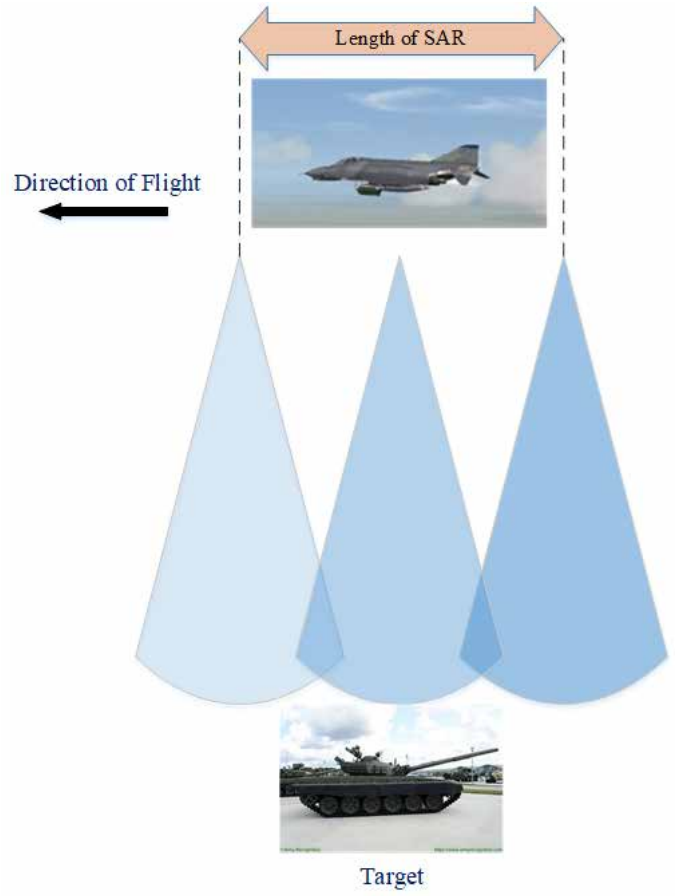

Figure 1. SAR System Operation Example. Adapted from [15].

In RAR systems, azimuth resolution δ_a is equal to the multiplication of the beamwidth θ_B of the antenna and range *R*. Beamwidth θ_B is equal to the wavelength λ divided by the diameter of the antenna *D*. Therefore, the azimuth resolution of the RAR is expressed as [16]

$$\delta_a = \theta_B \times R = \frac{R \times \lambda}{D}. \quad 2.1$$

In SAR, the synthetic length of the antenna *Ls* equals the beamwidth of SAR times the range *R*

$$Ls = \theta_s R, \quad 2.2$$

where the beamwidth of the SAR θ_s is calculated by

$$\theta_s = \frac{\lambda}{2L_s} = \frac{D}{2R}. \quad 2.3$$

Therefore, the azimuth resolution for SAR systems can be found by [16],

$$\delta_a = \theta_s R = \frac{R\lambda}{2L_s} = R\frac{D}{2R} = \frac{D}{2}. \quad 2.4$$

As seen in the equations, while in RAR, the azimuth resolution depends on range, and in SAR, it depends on antenna size. In SAR systems, a decrease in the antenna size results in better image resolution. In other words, shorter antennas increase the synthetic aperture. Increasing the synthetic aperture allows for more data to be gathered and increases the resolution of the images.

A. SAR OPERATING MODES

SAR systems can be run in various imaging modes by changing the antenna scanning pattern. These modes are stripmap, spotlight, scan SAR, interferometry, polarimetric SAR, and inverse SAR (ISAR) [17]. The spotlight, stripmap, and ISAR modes are briefly explained in the following section.

1. Stripmap SAR

Stripmap SAR is usually used for imaging relatively large areas. In this mode, it is possible to obtain images of targets that are positioned in a wide area [18]. The scanning of the beam pattern stays the same during the movement of the platform, and the SAR scans the area without changing the angle of the antenna, as seen in Figure 2.

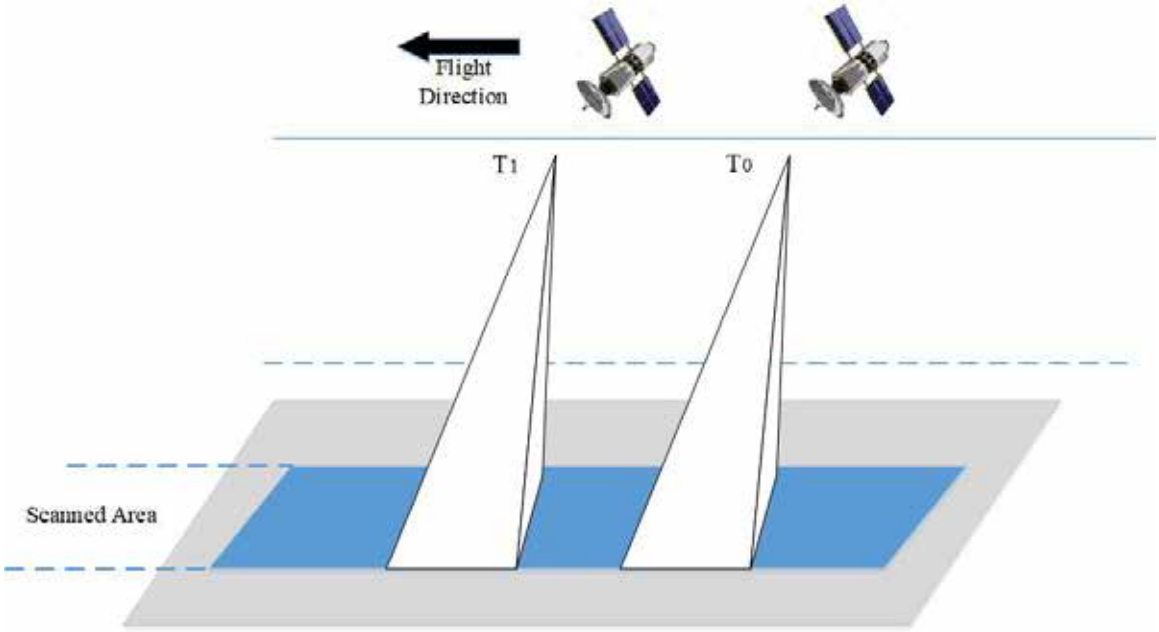

Figure 2. Stripmap Mode Scanning Pattern. Adapted from [18].

2. Spotlight SAR

Spotlight SAR is used to get a finer resolution of the areas or targets. Thus, during the movement of the platform, the antenna beam scans only specific areas or targets wanted for imaging [18]. In this mode, antennas monitor only a focused area from different angles depending on the movement of the platform, as seen in Figure 3. The resolution of the spotlight SAR is better than that of the stripmap SAR mode.

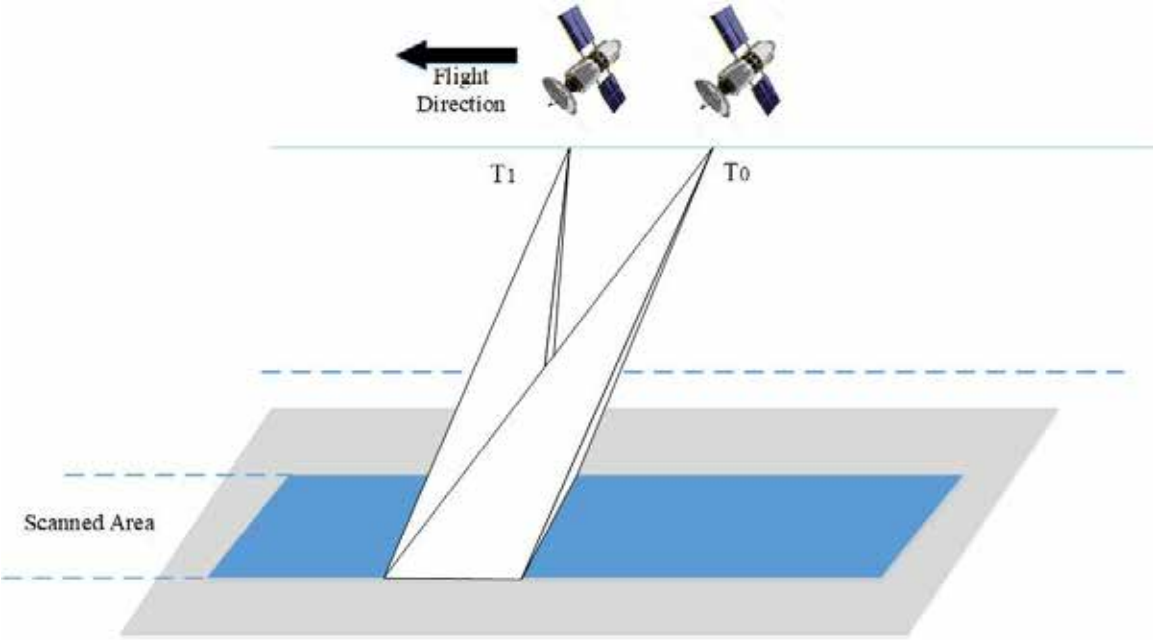

Figure 3. Spotlight Mode Scanning Pattern. Adapted from [18].

3. Inverse SAR

While the target is stable and the radar is moving in SAR applications, the opposite applies in ISAR. The target moves, and the radar is relatively stable in this case. This technique is most often used to get images of ships.

B. SAR OPERATING FREQUENCIES

In SAR, just like other radar systems, in order to monitor objects, signals are sent to and echoes are received from objects. In SAR, the received signals from objects or targets change based on their physical features like permittivity, geometry, and roughness. These properties also affect the penetration of the signal. For example, signal penetration ratio differs from dry to wet land. Therefore, the utilization of frequency in SAR applications changes according to the purpose of the imaging. Frequently used frequencies and related applications in SAR are shown in Table 1 [19]. Military applications concentrate on the X Band (8–12 GHz) frequencies since they give high resolutions.

Frequencies between 1 and 30 GHz can be used for remote sensing operations. Nevertheless, frequencies between 1 and 10 GHz have better transmissivity; therefore, this

range of frequencies is utilized to monitor objects or targets independent of the weather conditions in SAR applications. It should also be noted that transmissivity decreases as frequency increases [14].

Table 1. SAR Frequencies and Related Applications. Source: [19].

Frequency Band	**Frequency Range**	**Application Example**
VHF	300 KHz – 300 MHz	Foliage, Ground Penetration, Biomass
P-Band	300 MHz – 1 GHz	Biomass, Soil Moisture, Penetration
L-Band	1 GHz – 2 GHz	Agriculture, Forestry, Soil Moisture
C- Band	4 GHz – 8 GHz	Ocean, Agriculture
X- Band	8 GHz – 12 GHz	Agriculture, Ocean, High Resolution Radar
Ku- Band	14 GHz – 18 GHz	Glaciology (snow cover mapping)
Ka-Band	27 GHz – 47 GHz	High Resolution Radar

C. POLARIZATION

Polarization is another important parameter for SAR imaging. Horizontal-horizontal (HH), vertical-vertical (VV), horizontal-vertical (HV), and vertical-horizontal (VH) are the polarization types. Traditional SAR systems use single polarization like VV or HH, but modern SAR systems can utilize dual polarization. Just like frequency, every polarization type has a different backscattered signal from the object. Thus, polarization is a key parameter to determine the features of an object [14].

III. CONVOLUTIONAL NEURAL NETWORK

As mentioned in Chapter I, deep learning is a sub-set of machine learning, which has a layered structure that uses neural networks. It is known as “deep” because of the layers of neural networks. A convolutional neural network, also known as ConvNet or CNN, is a type of DL method that is widely used for image recognition and classification. The structure of this type of neural network is inspired by the human neural system. It is a remarkable method that consists of multiple layers. Those layers are connected to each other end to end, which means that the output of one layer connects to the input of the next.

CNN was utilized by Kunihiko Fukushima in 1979 for the first time [20]. Fukushima designed a network called Neocognitron, which consists of pooling and convolution layers [21]. In 1988, Yann LeCun and his friends created LeNet, the first CNN architecture, which was used to recognize characters and digits [22]. In the 1990s, with the development of computer technologies like graphic processing units (GPU) and faster data processing computers, studies about DL and CNN accelerated [23]. Until 2009, lack of data was one of the biggest drawbacks for CNN. In 2009, Fei-Fei Li and his colleagues created the ImageNet database, which includes more than 14 million images [22], helping to resolve the lack of data, and in 2012, AlexNet, a renowned CNN architecture, won the ImageNet competition based on image recognition. After 2012, CNN became a widely known image-recognition method around the world [20]. Researchers and governments increased the number of studies about CNN, and besides AlexNet, many CNN architectures were created, including ZFNet, VGGNet, and GoogleNet [22]. Although CNN is mostly used in image recognition, in recent years it has also been used for sound classification, road sign recognition, biomedicine, and human motion recognition due to its robustness.

A. CNN ARCHITECTURE

Every CNN consists of a multi-layered structure, but the number of layers and the order of these layers vary in each CNN architecture depending on the purpose of the network. In a simple view, every CNN contains three basic layers: convolutional, pooling, and fully connected. The basic CNN architecture is given in Figure 4.

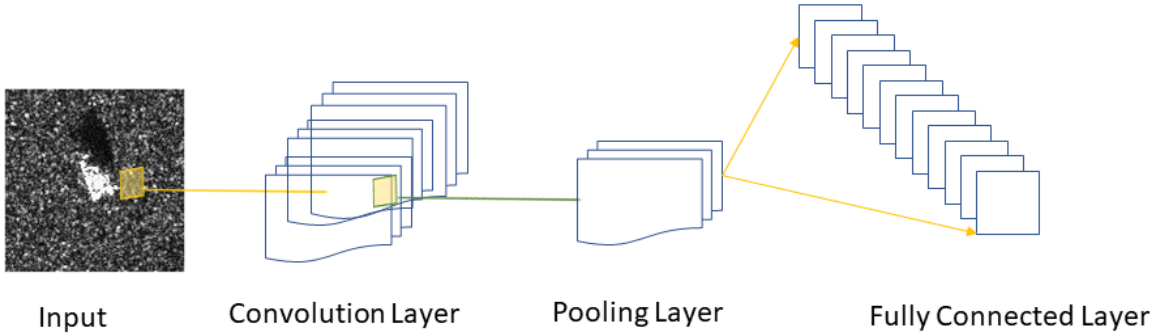

Figure 4. Basic CNN Architecture

As seen in Figure 4, the CNN has an end-to-end structure, which means that the product of one layer is the input of the next or connected layer.

B. CNN LAYERS

CNN is a DL method that utilizes different kinds of layers. These layers start with the input layer, which is commonly made up of images. Then, it continues with the convolution layer, from which the name of the method is derived. The activation layer follows the convolution layer. In this layer, a rectified linear unit (ReLU) is typically used. After the ReLU, pooling layers come to do subsampling. Fully connected and classification layers are the last layers of the CNN cycle.

1. Input Layer

As the name indicates, the input layer is the initial layer of a CNN. In this part of the CNN, the raw data is transmitted to the network. The most important consideration for this layer is the size of the data [24]. The size of the data determines the performance and the success of the network. For example, a high-resolution image requires high capacity processors and takes more time to train the algorithm. Moreover, it can improve the success of the classification. On the other hand, low-resolution images take less time to train the algorithm and do not need high capacity processors, but the resulting performance of the CNN may be low in terms of classification.

CNN is used primarily for classification and recognition; therefore, images are the type of data generally used in this layer. The input layer has the pixel features of the input data. Thus, the data size is described as N x N x D, which denotes height, width, and color channels, respectively. D can only be assigned values of one, two, and three since it symbolizes red, green, and blue colors. For instance, a 512 x 512 x 3 input-size image has 512-by-512-pixel size and R, G, B color. Two represents the dual combination of R, G, and B color. In Figure 5, a 128 x 128 x 1 image is given as an input example. As seen in the figure, the input is a grayscale image; therefore, the color channel number is one.

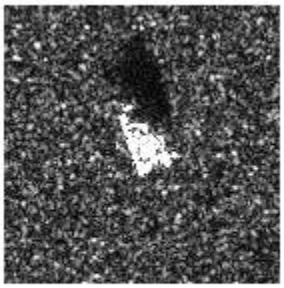

Figure 5. Input Example (128 x 128 x 1 Image)

2. Convolutional Layer

CNN takes its name from this layer; thus, this layer is the core part of the network. The main function of this layer is to perform feature extraction from the input data. Extraction is done by a convolution operation using two matrixes. One is the input matrix and the other is the filter, or in other words, the kernel.

The filter scans all over the input matrix by moving horizontally and vertically. It starts from the top left corner of the matrix and finishes at the bottom right corner of the input matrix. During this movement, the convolution operation is done, and results are saved for the output, which is the feature map of the input. Filter sizes and values change according to the intended purpose of the filter. It can be used for edge or curve detection, sharpening, and masking. In the convolution layer, there are three parameters that

determine the dimensions of the output. These parameters are depth, stride, and zero-padding [25].

The depth parameter is related to how many filters are applied to the network. Each filter is used for a different purpose, so they search various features like edges, colors, or lines. Multiple filters can be implemented in this layer. Among them, the first layer gets sparse factors like lines, edges, and corners. The following layers get higher degrees of features. As more filters are applied in the convolutional layer, then more features are gathered, but this operation increases the complexity of the network. For example, applying a 32-pixel sized filter to a 256 x 256 sized image leads to a 256 x 256 x 32 size of calculations, which is equivalent to almost two million calculations. This high number of data calculations decreases the performance of the network and needs high processing devices. Utilizing fewer filters may reduce the number of neurons in the network, but this may lead to the loss of some training abilities.

Stride is another parameter that affects the complexity of the network. It is used to decide how to move the filter over the input matrix. It is a number that determines how the filter slides in every convolution operation. For example, if the stride number is two, then after every convolution operation, the filter shifts two pixels. Having a greater slide number can build a lower output volume and decrease the processing time, but it may also lead to the loss of some features. Generally, small stride sizes, such as two or three, are used. In Figure 6, the process by which the filter and the stride are applied during the convolution operation is demonstrated. In this operation, the stride number is taken as two. As seen in the figure, the convolution operation starts from the top left side of the input matrix. The filter matrix is multiplied element-wise with the input matrix, which is shown by the blue color, and the result is saved to the output matrix. Subsequently, the filter matrix shifts two pixels to the right. The operation continues until the filter matrix covers the entire input matrix.

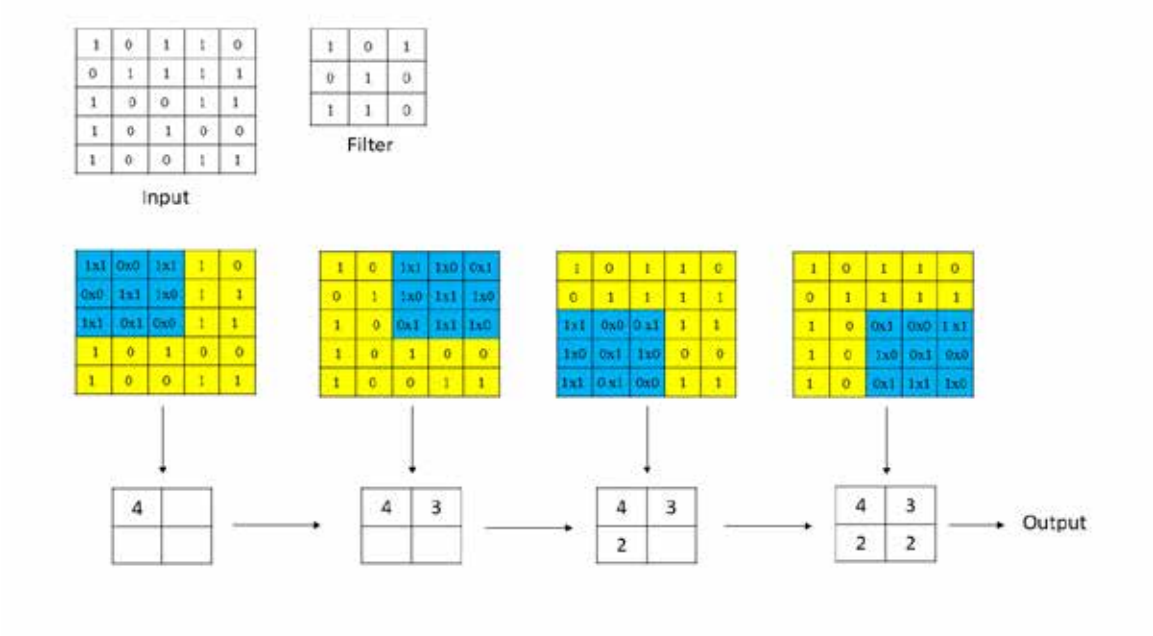

Figure 6. Convolution Operation

In the convolution operation, the size of the output can be configured by applying zero padding. Zero padding is a process that applies zeros outside of the input to assist in managing the size of the output. It is typically applied whenever a higher dimensional output volume is desired. In Figure 7, zero padding is applied to the previous example, which is given in Figure 6. Two-line zeros are implemented outside of the input, and the same filter is used for the convolution operation. After zero padding, the size of the output becomes 4 x 4, while the output size is 2 x 2 without zero padding.

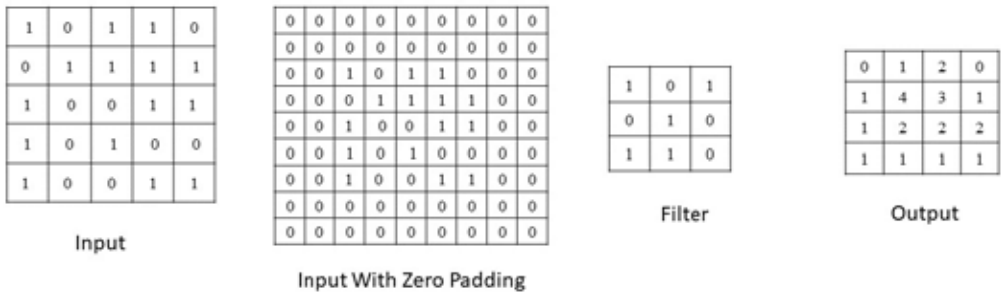

Figure 7. Convolution with Zero Padding

In the convolutional layer, the size of the output is calculated with the following formula [26]:

$$O = \frac{(W-F+2P)}{S} + 1\,, \qquad 3.1$$

where O is the output size, W is the size of the input, F is the filter size, P is the zero padding, and S is the stride [26]. For example, for a 5 x 5 input with a 3 x 3 filter and two stride numbers and zero padding, the output size is computed as $O = \frac{(5-3+2x0)}{2} + 1$ and the result equals 2 x 2.

3. Activation / ReLU Layer

The activation layer is subsequent to the convolution layer. In this layer, the rectified linear unit, or ReLU, is generally used; therefore, it is also known as the ReLU layer. ReLU is a function that sets the negative values to zero in the matrix. The mathematical illustration for it is y=max (x, 0). Although it does not change the input or the output size, the ReLU helps to enhance nonlinear features and boosts overall network performance. In contrast to other functions used in CNN as non-linear functions, the ReLU is agile and helps the network to learn faster than other non-linear functions [27]. Figure 8 illustrates how the ReLU function works. As seen in the figure, the ReLU sets negative values to zero.

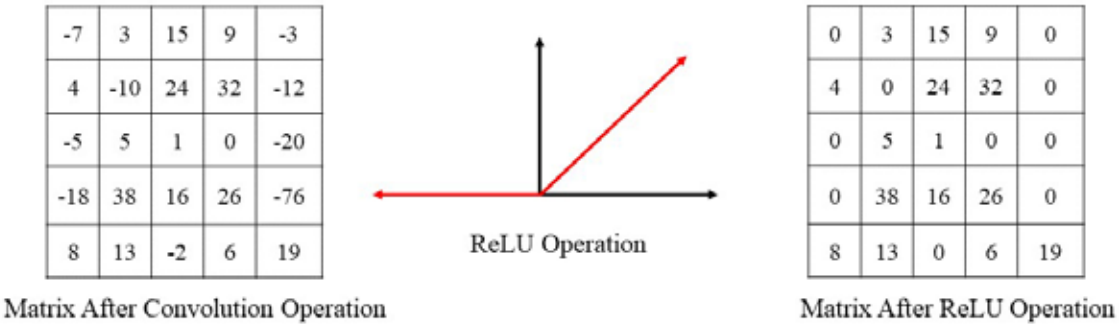

Figure 8. ReLU Function

4. Pooling Layer

After the ReLU layer comes the pooling layer. The purpose of the pooling layer is to reduce the size of the input data constantly to lessen the complexity of the network and the data that needs to be processed. While decreasing the size of the input, pooling does not affect the depth of the data. After this layer, the data size decreases substantially; hence, this layer is also known as the subsampling or down sampling layer.

There are two types of pooling: average and max pooling. As it can be understood by their names, average pooling uses the average value and max pooling uses the maximum value of the matrix for subsampling [28]. CNN architectures typically utilize the max pooling function.

Similar to the convolutional layer, in the pooling layer computation is done by the filter and the stride. Likewise, the max pooling is applied with the two-by-two filter and the stride of two, which reduces the size of the data to 25% of the original data while keeping the depth volume at the original value. There are cases in which the filter size is set to three and the stride is set to two, which results in an overlap in the pooling. As it changes the data size, having filter sizes higher than three generally reduces the performance of the network. The output size of the pooling layer can be computed with [26]:

$$O = \frac{(W-F)}{S} + 1\,, \qquad 3.2$$

where O is the size of the output, W is the size of the input data, F is the filter size, and S is the stride number [26].

In Figure 9, max pooling and average pooling are applied to a 6 x 6 convolved and rectified matrix with the filter size of 2 x 2 and the stride number of 2. As illustrated in the figure, the pooling process starts from the top left side of the matrix with the 2 x 2 filter and takes the max or average value of the sub-matrix. Afterwards, the filter slides to the next sub-matrix according to the stride number. The process goes until the filter covers the entire matrix in a similar way to the convolution operation.

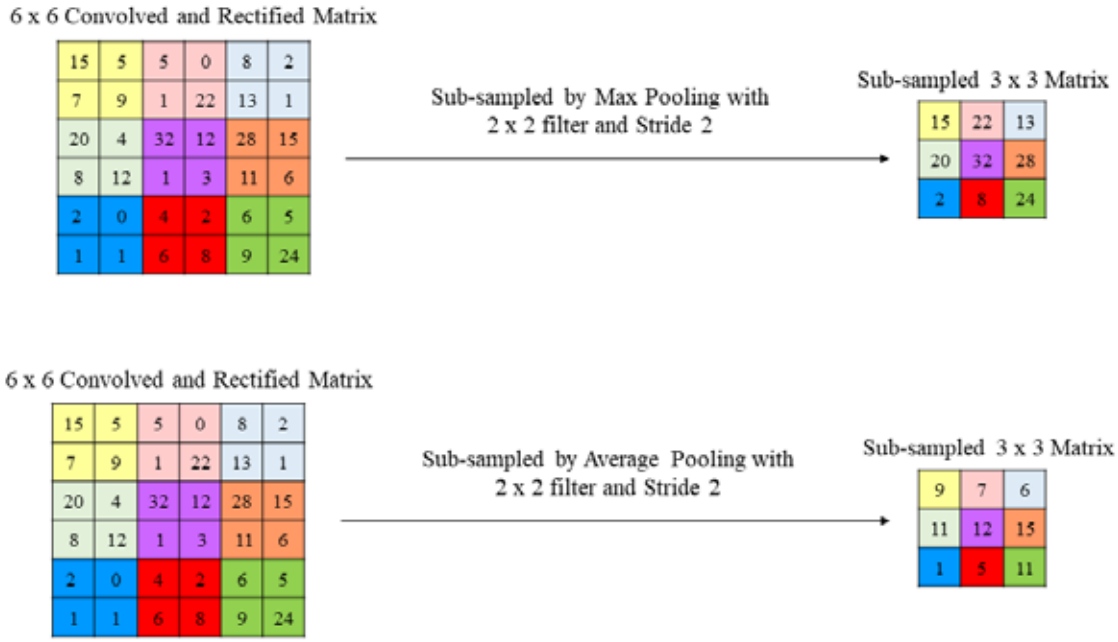

Figure 9. Max and Average Pooling Operation Demonstration

5. Fully Connected Layer

The fully connected layer is usually applied at the end of the CNN. One or multiple fully connected layers are used after the convolution and pooling layers. In this layer, every input neuron is connected to every output neuron. It is the reason it is called the fully connected layer. While the feature extraction is executed in the previous layers, the classification is done in the fully connected layer. The product of the convolutional and pooling layers contains high-level properties of the input data. The goal of the fully connected layer is to benefit from these high-level properties from previous layers to classify the input data. In the fully connected layer, every element from the earlier layers is used to calculate elements of the output.

6. Classification Layer

Following the fully connected layer, the classification layer is the final layer of the CNN architecture. In this layer, the classification is completed among the input objects. In this stage, features, which belong to the input image, are extracted from previous layers. As this layer is used for the classification, the output of this layer should be equal to the

number of classes that are to be classified [29]. For instance, if ten objects are to be classified, the output of the classification layer should be ten. Various classifiers are used in this layer. Softmax, due to its higher success rate than other classifiers, is preferred. The function of the Softmax classifier is to normalize the output values of a neural network and convert them to probability values, which are between zero and one. In Figure 10, the classification layer is shown with other CNN layers. The SAR image of a tank is given as an input and the classification is done with high accuracy. At the end of the image, the probabilities of the classification are provided. Notably, the sum of the probabilities is one.

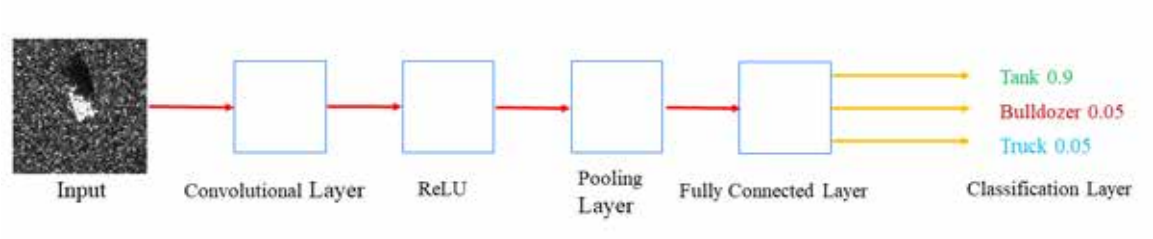

Figure 10. Classification Layer with Overall CNN Layers

C. TRAINING CNN

The training of CNN aims to find filters and weights in the layers in order to decrease the variation between the training data and output predictions. In CNN training, back propagation and forward propagation algorithms are used to calculate and update the errors [30]. First, filters and weights are measured with a loss function using forward propagation during the training. Then, filters and weights are restored in accordance with the loss value using the back propagation with the gradient descent optimization algorithm [31]. This process repeats a specific number of times, which is called epoch number or iteration.

The gradient descent is a widely used optimization algorithm, which aims to decrease loss during the training, through the use of updating parameters, filters, and weights. Stochastic gradient descent with momentum, SGDM, RMSprop, and Adam are widely used gradient descent algorithms. The gradient is formulated as [31]:

$$w = w_i - \alpha * \frac{\partial L}{\partial w}, \qquad 3.3$$

where w stands for the weight, which is the learning parameter, w_i denotes the initial weight, α stands for the learning rate, and L denotes the loss function [31]. As seen in Equation *3.3*, the learning rate is an important parameter for the training of the CNN. It is adjusted before the training is started by the user. Setting high values for the learning rate causes greater steps during the weight updates; therefore, it may take less time to train. Nevertheless, a higher learning rate may not decrease the loss as desired, since it causes massive jumps that are not accurate enough to come to the ideal point. During the training, validation loss and accuracy is calculated according to validation frequency. The validation frequency number determines how many validation iterations are done in every epoch.

D. CNN STRUCTURE IN THIS STUDY

Even though it seems like every CNN structure looks similar, in fact every one of them differs from the others in several aspects: the number of layers, the filter sizes and numbers, the activation layer that is used, the padding size or usage, etc. These parameters shape the structure. The user decides how to use these parameters according to the intended purpose, processor, or hardware capacity.

In this study, the classic CNN structure method is pursued, because it has been studied extensively and is a proven computing application for image classification. In this method, convolutional layers are followed by ReLU and pooling layers, respectively. This structure repeats five, six, or seven times according to simulation results. In the convolutional layer, the two-by-two (2 x 2) filter size, and the stride of one (stride 1) are applied to the input matrix. The padding size is adjusted to make the same output and input size. In the pooling part, max pooling is utilized with the 2 x 2 filter size and stride 2. The repeated convolutional structure is finalized with one fully connected layer and one classification layer. Figure 11 illustrates the structure.

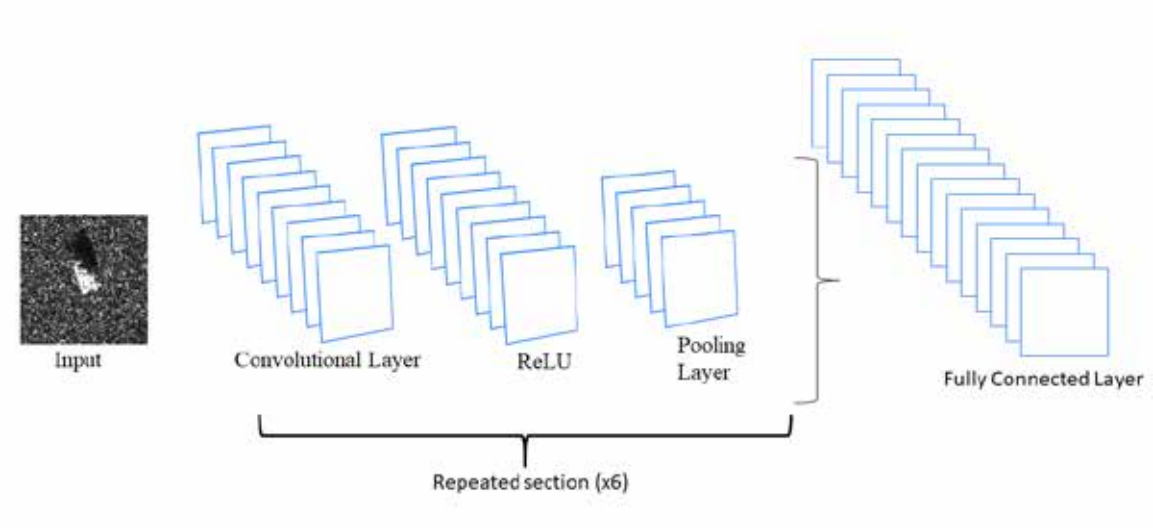

Figure 11. Basic Thesis CNN Structure

THIS PAGE INTENTIONALLY LEFT BLANK

IV. SIMULATION

The scope of this study is the recognition and the identification of military targets through the use of the CNN algorithm. Although there are some widely known, ready-to-use CNN toolboxes like TensorFlow, Matconvnet, and PyTorch via Python, MATLAB deep learning toolbox is used to train and classify targets in this simulation. MATLAB deep learning toolbox has commands for creating layers and optimizing training options. These commands and training options are explained in detail in this chapter. Furthermore, the CNN structure prepared for this simulation is specified. Simulations for this study are performed for four separate cases. The cases and how they are handled during this work are described in this chapter.

Aside from the simulation tool or the environment, the data is another important aspect for CNN. SAR images of military targets are not as easy to find as the images used for civil purposes are. Since SAR images of military targets are limited, collecting data for the CNN is challenging. Yet, there are some ways to overcome data shortages, such as the refinement of the image quality. In this thesis study, simulation cases to improve image quality and their effects on CNN training are analyzed. In this work, the SAR image data is taken from the moving and stationary target acquisition and recognition, MSTAR, program, which is available for public use [32]. Features and types of the data are thoroughly presented in the following sections of this chapter.

A. SIMULATION METHOD

As mentioned earlier, MATLAB deep learning toolbox is used to carry out the simulation part of this study. MATLAB is a user-friendly tool for creating customized CNN structures. It has commands for building layers according to user needs. Users can easily modify their own CNN architecture by employing MATLAB.

The simulation structure begins with the input layer as described in Chapter II. The input is a SAR image that is 128 x 128 x 1 in size. The input size is the same for the training and test data. Then, the method proceeds to the 2D convolution layer. The first convolutional layer is constituted with eight 2 x 2 sized filters. The number of filters

doubles with every following convolutional layer. For example, the first layer has eight filters, the second one has 16, the third one 32, and continues in this way. The number of filters increases because of the ascending data complexity as layers go deeper or forward. For instance, the first layer extracts or processes raw data like edges, lines, or dots, whereas subsequent layers include finer data such as the combination of two or more lines or dots. Consequently, it is better to implement more filters to final layers. Additionally, zero padding is utilized to make the output size and the input size equal in the final layer.

Next, 2D convolutional layer batch normalization is applied to the structure. It is used to adjust the input for each layer in the CNN structure, it aids to normalize learning progress and decreases the training time of the network. It is applied after every convolutional layer in the structure.

The ReLU layer follows the batch normalization part in the method. It improves the performance of the network as it accelerates the training time. The main purpose of using both ReLU and batch normalization is to increase efficiency of the network and reduce the time of training.

The pooling layer completes the package, which is constituted of the convolutional layer, batch normalization, and ReLU. In the CNN structure designed for this study, max pooling is applied after every ReLU layer. It has a 2 x 2 filter with two strides. In this structure, the package repeats itself until the fully connected layer is reached. This study analyzes the training and validation accuracy using three to eight convolutional layers, in other words, a three-to-eight package of the convolutional layer, batch normalization, ReLU, and pooling. In Figure 12, the MATLAB view of the first package is demonstrated.

```
convolution2dLayer(2,8,'Padding','same')
batchNormalizationLayer
reluLayer
maxPooling2dLayer(2,'Stride',2)
```

Figure 12. MATLAB View of the First Package

After this package repeats six to eight times according to the simulation sequence, the fully connected layer, Softmax, and the classification layer come at the end of the CNN architecture in the simulation. Softmax is a function that regulates the output by converting numbers into probabilities whose values are betwen zero and one. It is used just before the classification layer, and its output equals the number of targets.

Next, in creating the CNN architecture for the simulation, it is necessary to set up the parameters of the training options to train the CNN. For the training, stochastic gradient descent with momentum (SGDM) is used as a solver, and the learning rate is set between 0.01 and 0.012. Higher learning rates provide faster learning for the network; on the other hand, they decrease the performance of the network. Maximum epoch or iteration number, the main parameter that determines the training time, is adjusted between 50 to 100. It takes approximately 300 minutes to complete 100 epochs with a computer that has a single CPU processor. Eventually, validation frequency, which calculates the validation accuracy and loss during the training, is set to ten in the training options of the network.

B. TESTING AND TRAINING DATA

In this work, there are two types of data available for public use on the MSTAR internet site. The first data types are provided with the name of “Mixed Targets,” while the other ones are given with the name of “T72 Variants.” The types of images in the mixed targets dataset and the number of data that are used for training and testing are specified in Table 2. As seen in the table, mixed targets consist of tanks, bulldozers, military vehicles, armored personnel carriers, and fake targets. These targets look relatively different from each other.

Table 2. Mixed Target Types and Data Set. Adapted from [32].

Target Picture	Target Name	Number of Training Data	Number of Test Data	Target Picture	Target Name	Number of Training Data	Number of Test Data
	BRDM2	1141	274		T62	299	273
	BTR60	256	195		ZIL131	299	274
	D7	299	274		ZSU23	1127	274
	SLICY	2265	274		2S1	890	274

The types of images in the T72 variants dataset and the number of data that are used for training and testing are given in Table 3. The contents of this dataset, however, look relatively like each other as all of them are a type of tank. In this dataset, there are fewer images than in the mixed targets dataset.

Table 3. T72 Variant Types and Data Set. Adapted from [32].

Target Picture	Target Name	Number of Training Data	Number of Test Data	Target Picture	Target Name	Number of Training Data	Number of Test Data
	A04	299	274		A32	298	274
	A05	299	274		A62	299	274
	A07	299	274		A63	299	274
	A10	296	271		A64	1143	274

In both tables, the training data number is higher than the number of the test data because more data is required in the training part of the simulation. Moreover, although images are given in color in both tables, black and white SAR images are used in the simulation.

C. SIMULATION CASES

In this study, five different cases are taken into consideration. Mainly, data types and numbers form these cases, but the number of layers and the effect of CNN architecture are also tested in various cases.

In the first case, mixed target data is trained and analyzed for CNN structures having different numbers of layers, concentrating in particular on structures of six to eight layers. In this case, the training dataset is arranged as demonstrated in Table 2. In the second case, the T72 variants dataset given in Table 3 is examined with the CNN architecture. Since there are fewer images in this dataset, further analyses are done to increase the accuracy level. For the T72 variants dataset, the training dataset is increased by taking images from the test data. Thus, in the third case, the T72 variant dataset is examined with more training data in the CNN. In this case, a relation between the training data and the training accuracy is observed. Then, in the fourth case, to improve the accuracy level for this dataset, the noise on the images is masked to try to enhance the quality of the images. In this case, with the same training and testing data as in Table 3, the effect of masking on the accuracy is tested. In the last case, mixed targets and T72 variants datasets are combined and tested with the CNN architecture developed for this work.

In this study, separate datasets are used for the training and testing. It aims to keep the amount of training data higher than the testing data for a better training process. The MSTAR dataset provides images at different depression angles. Different depression angles are used for training and testing considering the numbers. For the mixed targets, images are taken at 15, 16, 17, 29, 30, 31, 43, 44, and 45-degree depression angles. For the T72 variants, images are taken at 15, 17, 30, and 45-degree depression angles. In this experiment, a total of 6,576 images, gathered at 17 to 45-degree depression angles, is used for the training of mixed targets, and a total of 2,112 images, taken at 15-degree depression

angles, is used for the testing. Similarly, for the T72 variants, a total of 2,189 images, taken at 15-degree depression angles, is used for the testing, and a total of 3,232 images is used for the training. It is aimed to keep the training data high; therefore, only 15-degree depression angle data is used for the testing part.

The performance of the CNN is evaluated on how accurately objects are classified. In the CNN, training and validation accuracy decides the success of the structure. The training challenges, the learning process of the network, and the training accuracy indicate how well the data is learned. On the other hand, validation accuracy shows how well the structure is built by testing the data. The other performance metric of the network is the processing time. The fastest computation time is desired. Nevertheless, overall performance depends on the combination of both. For instance, 95% accuracy with an hour of processing time would be better than 80% accuracy with a minute processing time.

V. RESULTS AND ANALYSIS

A. RESULTS

In this chapter, the simulation results are provided in five separate sections corresponding to the cases described in the previous chapter.

1. Mixed Target Results

In the first case, CNN is tested for the targets, which are given in Table 1. As stated in the previous chapter, these targets are relatively distinct from each other and the dataset includes more data. Therefore, better accuracy results are achieved in this test. In this case, the CNN structure is examined for six, seven, and eight convolutional layers. Among all the tests that are run, a maximum 99.76% accuracy is achieved; however, the results are based on the average accuracy of four separately run test, where seven-layer CNN structure has the highest average accuracy. In CNNs, as the number of layers increases in the network, accuracy gets better. In this experiment, the seven-layer CNN structure is found to be the optimum because after seven layers, the accuracy level stayed the same. During the study, CNN structures with different numbers of layers are examined, but CNN structures composed of six to eight layers are the main focus, since structures having that many layers achieve optimal results and a maximum accuracy level.

The study started with three layers, with 100 epochs iteration, and an accuracy of 96.64%. Then, four layers are examined, and the accuracy goes slightly higher to 96.78%. After four layers of CNN structure, five layers are tested, and accuracy is improved to 98.72%. The training progress and confusion matrix related to three, four, and five layers of CNN are given in Figures 13, 14, and 15, respectively. In those figures, it is seen that the accuracy level slightly increases as the layers are increased.

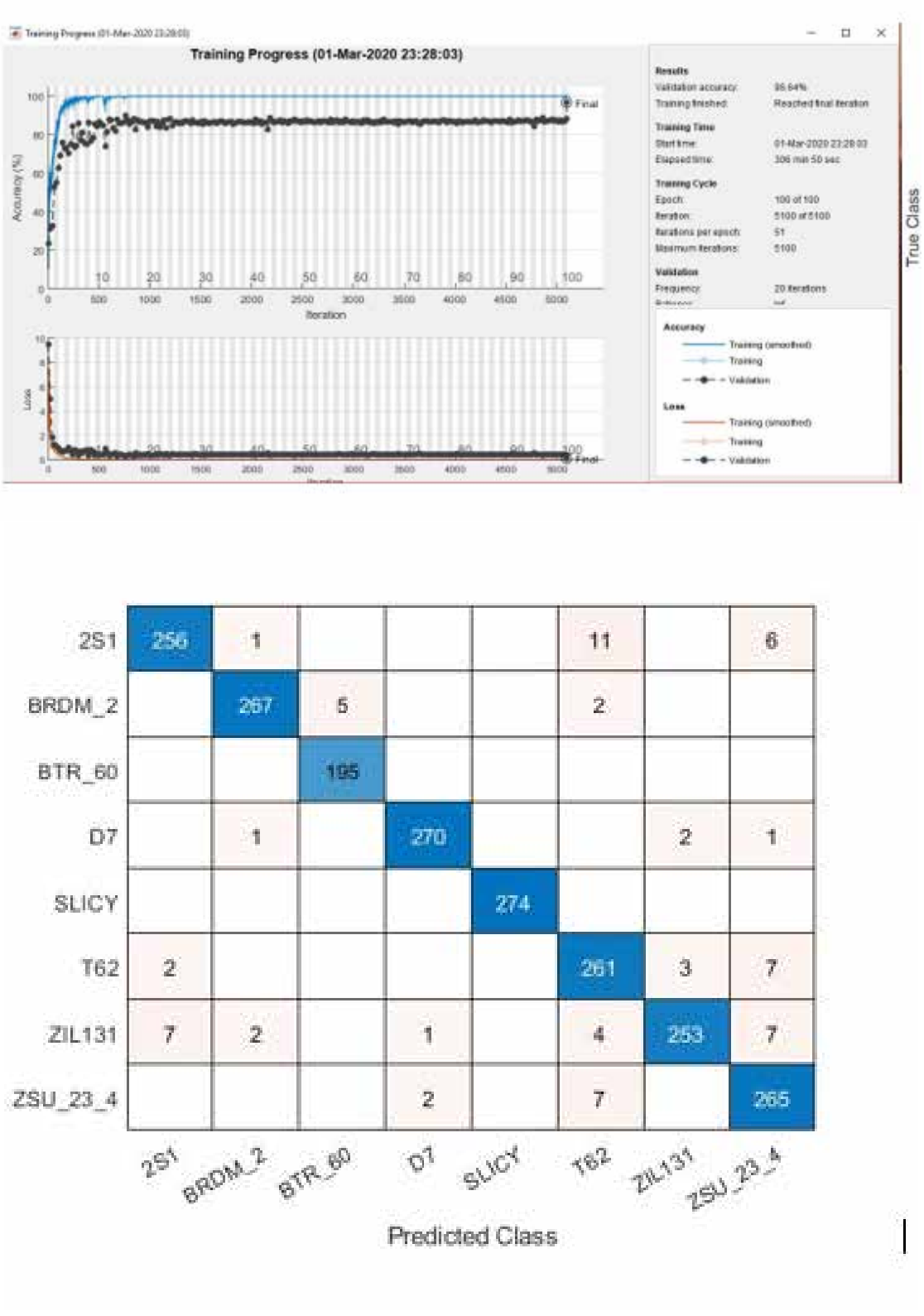

Figure 13. Three-Layer CNN Training Progress and Confusion Matrix for Mixed Targets

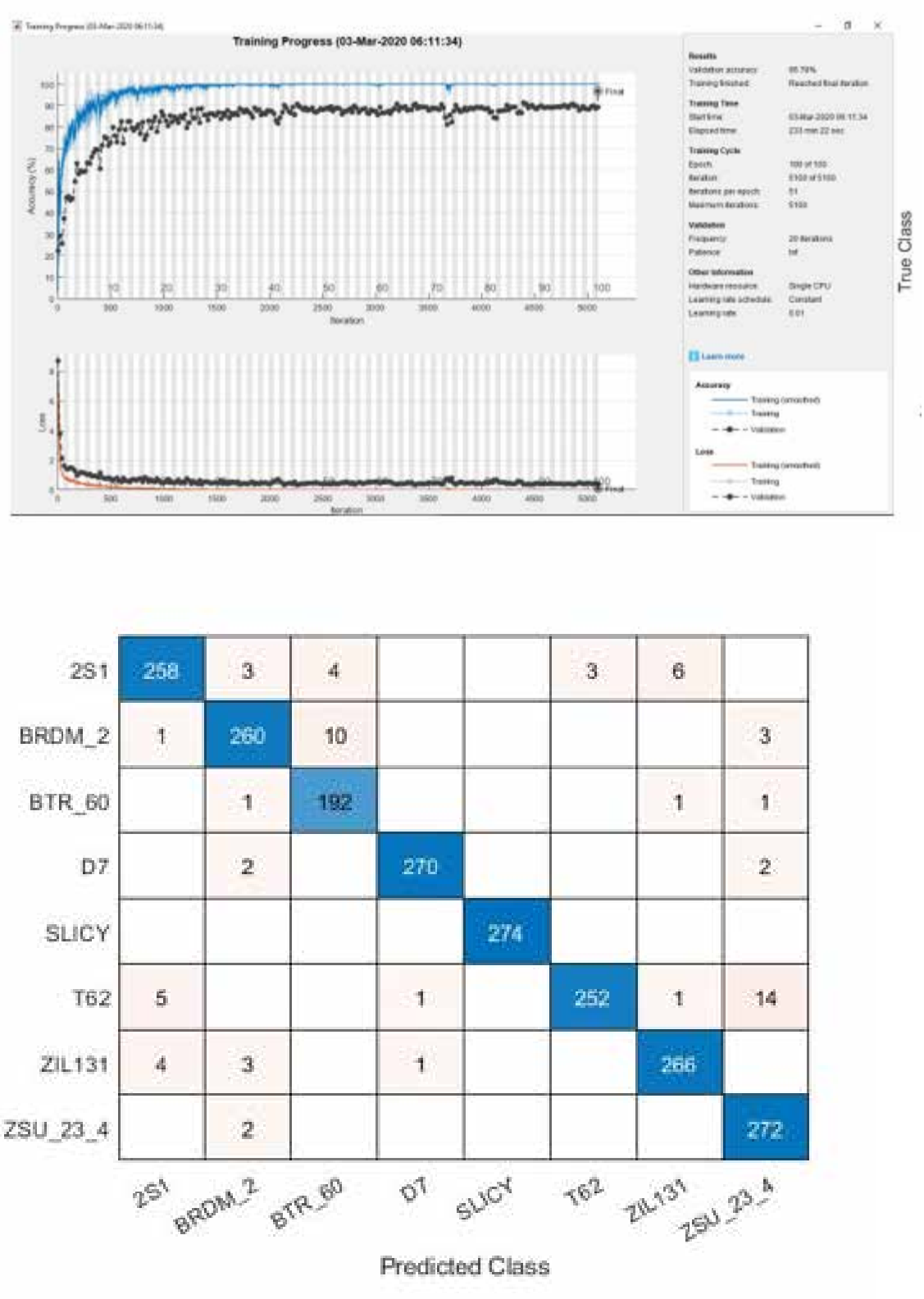

Figure 14. Four-Layer CNN Training Progress and Confusion Matrix for Mixed Targets

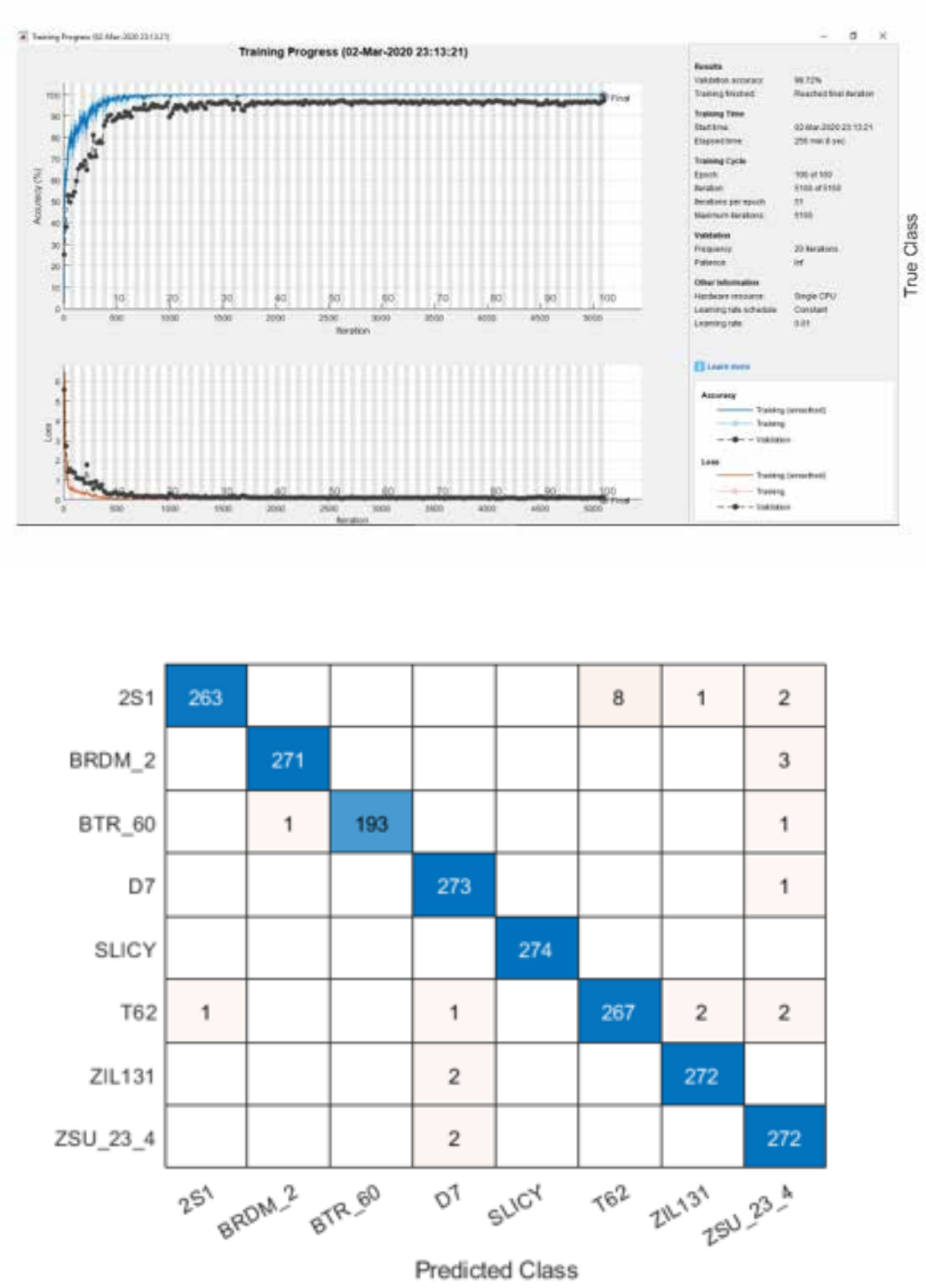

Figure 15. Five-Layer CNN Training Progress and Confusion Matrix for Mixed Targets

After three, four, and five layers, CNN structures are examined in four separate runs for six, seven, and eight layers, respectively. Results of all four runs are given in Table 4 for each layer. As seen in the table, the best result for all layers is 99.76% accuracy. However, the seven-layer CNN structure's average accuracy for the four runs is better than for the other layers' structures.

Table 4. Results from Six-, Seven-, and Eight-Layer CNN Structures for Mixed Targets

Runs	Accuracy of 6 Layers CNN	Accuracy of 7 Layers CNN	Accuracy of 8 Layers CNN
1	99.53%	99.62%	99.43%
2	99.76%	99.67%	99.72%
3	99.20%	99.76%	99.34%
4	99.62%	99.67%	99.76%
	Avg: 99.5275%	Avg: 99.68%	Avg: 99.5625%

The training progress and confusion matrix of the best result, which is 99.76%, is given in Figure 16. According to the bottom figure, a total of five confusions occur during the classification, and most of them are seen between D7 and ZIL131.

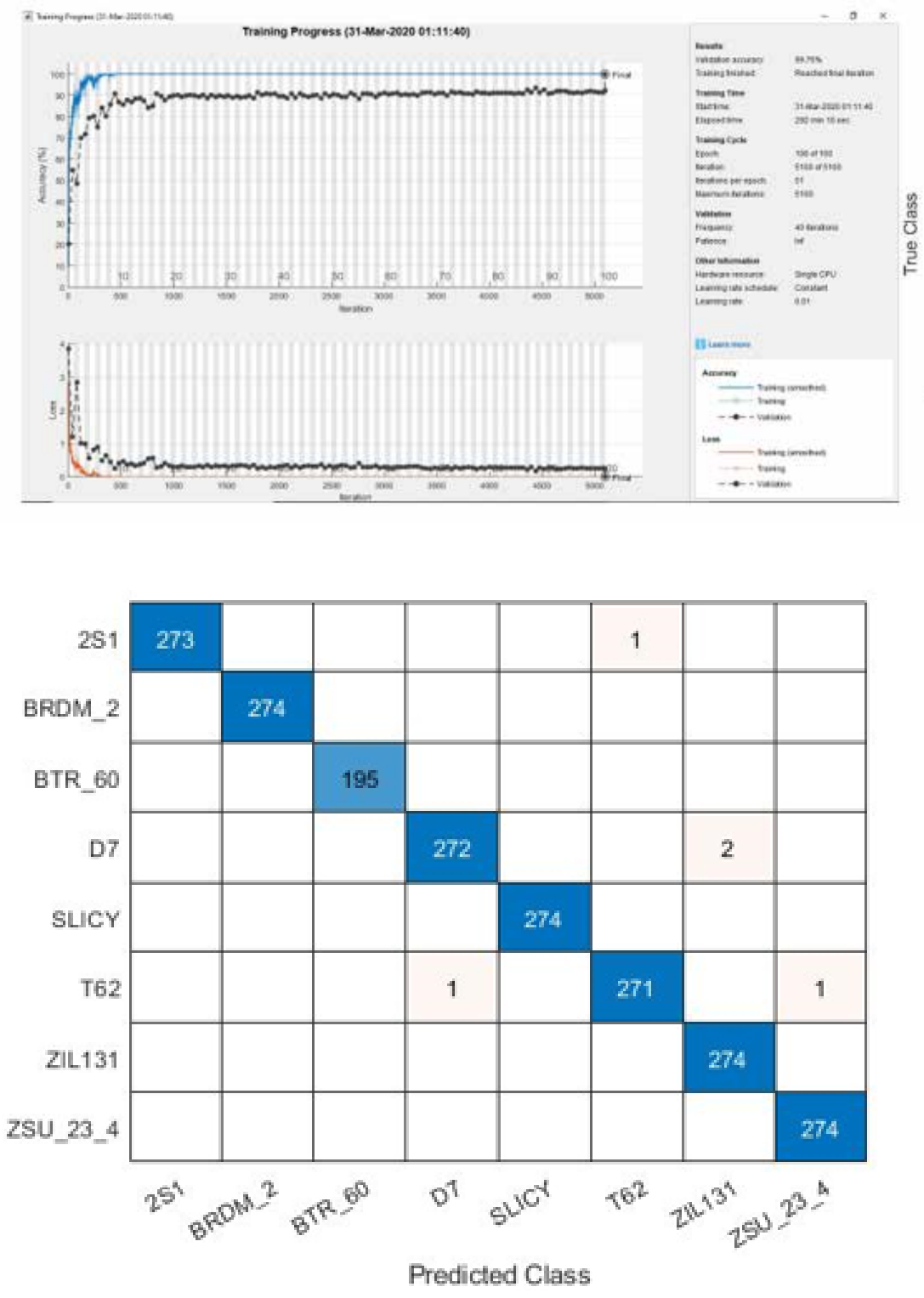

Figure 16. Training Progress and Confusion Matrix for 99.76% Accuracy for Mixed Targets

2. T72 Variants

In this part of the simulation, the data given in Table 3 is used for the training and testing of the CNN architecture. When this dataset is compared with the mixed targets dataset, there is less data in the training part and these targets look alike as all of them are T72 tank types. As a result, a lower level of validation accuracy is achieved compared to previous case in this part of the experiment. In this case only 90.77% accuracy is reached. The training progress and the confusion matrix for this experiment are given in Figure 17. As seen in the figure, the training time is shorter than in the previous runs because there is less data. Also, during the validation most of the confusions occurred between A64 and A04 tanks.

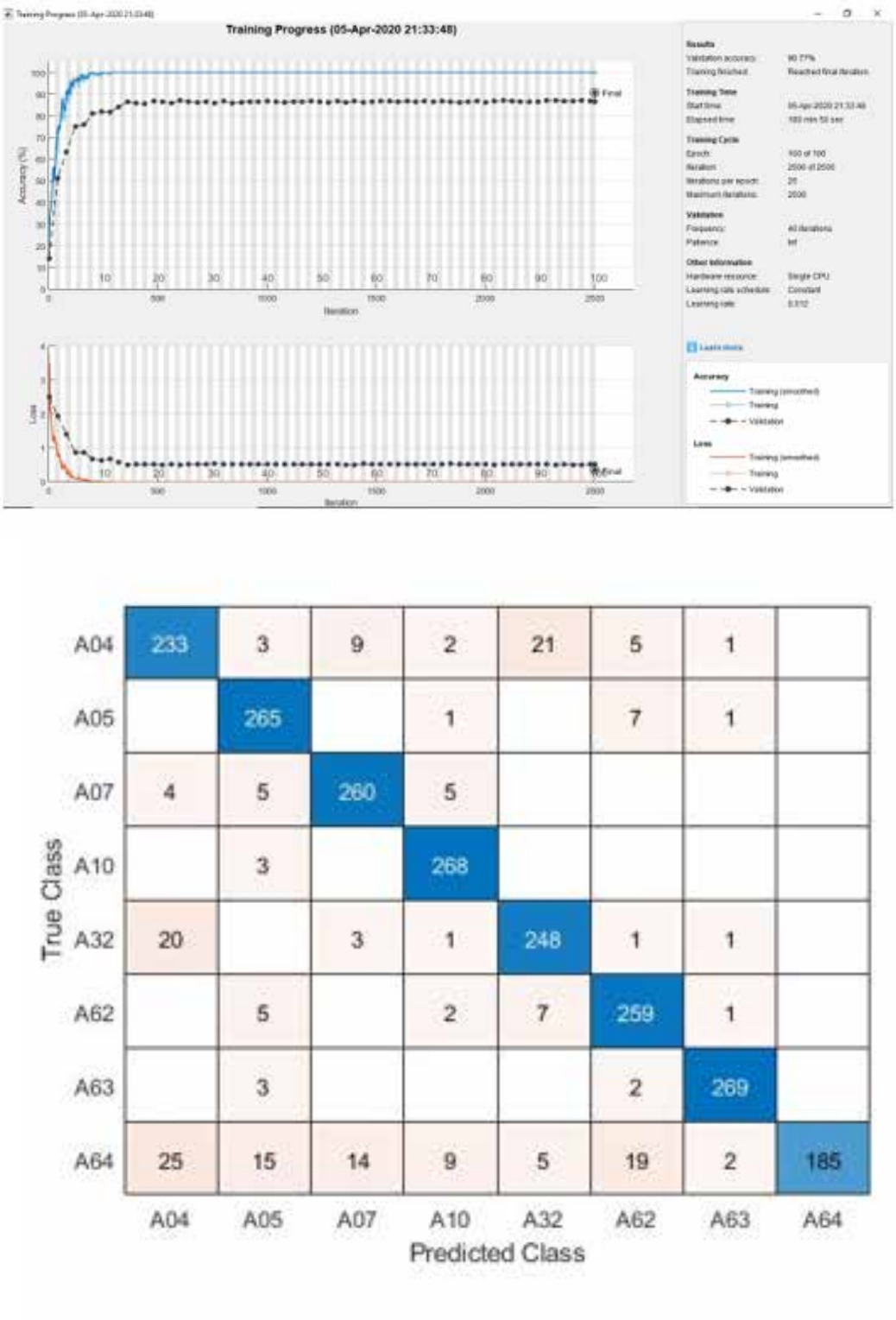

Figure 17. Training Progress and Confusion Matrix for 90.77% Accuracy for T72 Variants

3. Increased T72 Variants Training Data

In order to increase the accuracy level for T72 variants, the training data is increased by taking images from the test data. In this part of the experiment, the purpose is not only to increase the accuracy but also to examine how data is affecting the accuracy level. With this objective in mind, the training data is gradually increased and tested using the same CNN structure and training parameters. The first simulation is started with 299 training data for each target except the A64 type, which has 1,143 images. Then the dataset is increased gradually by adding 50 for the first iteration, then 25 for rest of the iterations. Results for each iteration are given in Table 5. As seen in the table, increasing the training dataset contributed to boosting the validation accuracy level, eventually reaching 98.49%. Compared with the results of the first run, which is 90.77%, it is a huge improvement in terms of accuracy. However, while the training dataset is increased, the testing dataset is decreased. Therefore, less testing data may lead to fewer errors in the results since the chances of seeing an error decreases when checking a smaller number of data.

Table 5. Results from Increased Training Dataset Size for T72 Variants. Adapted from [32].

Number of Training Data	**Number of Testing Data**	**Accuracy Level**
299 (A04, A05, A07, A10, A32, A62, A63) and 1143 (A64) Total: 3236	274 for each type Total: 192	90.77%
350 (A04, A05, A07, A10, A32, A62, A63) and 1192 (A64) Total: 3642	224 for each type Total:1792	94.55%
375 (A04, A05, A07, A10, A32, A62, A63) and 1217 (A64) Total: 3842	199 for each type Total:1592	94.37%
400 (A04, A05, A07, A10, A32, A62, A63) and 1242 (A64) Total: 4042	174 for each type Total:1392	95%
425 (A04, A05, A07, A10, A32, A62, A63) and 1267 (A64) Total: 4242	149 for each type Total:1192	97,21%
450 (A04, A05, A07, A10, A32, A62, A63) and 1292 (A64) Total: 4442	124 for each type Total:992	98%
475 (A04, A05, A07, A10, A32, A62, A63) and 1317 (A64) Total: 4642	100 for each type Total:792	98.49%

4. Masking Input Image

Masking is used to focus on a specific part of the image. To make a region of interest clearer or sharper, the background or unwanted parts of the image are darkened [33]. Currently, this technique is widely used in medical areas to find the broken part of a bone, an anomaly in tissues, and so forth, from MRI or X-Ray images [34], [35].

This simulation aims to improve image quality or make it easier to classify images by darkening background noise from the SAR images, using a masking method. The images used in this study are grayscale images. Therefore, pixel values in the image, which indicate the brightness of the pixel, are between zero and 255. Generally, zero is considered as black and 255 as white. Targets in the images are brighter; thus, their pixel values are higher than those for pixels located in the background. Masking the background of images follows this idea.. Firstly, the mean value of the image is calculated; secondly, the background of the image is masked by darkening the pixels one, two, and three times over the mean value, respectively. In Figure 18, the original and masked images are illustrated. In the figure, the top left one is the original image. On the right side, it is the image masked one time over the mean value. The bottom left one is masked two times over the mean value, and the last one on the right is masked three times over the mean value.

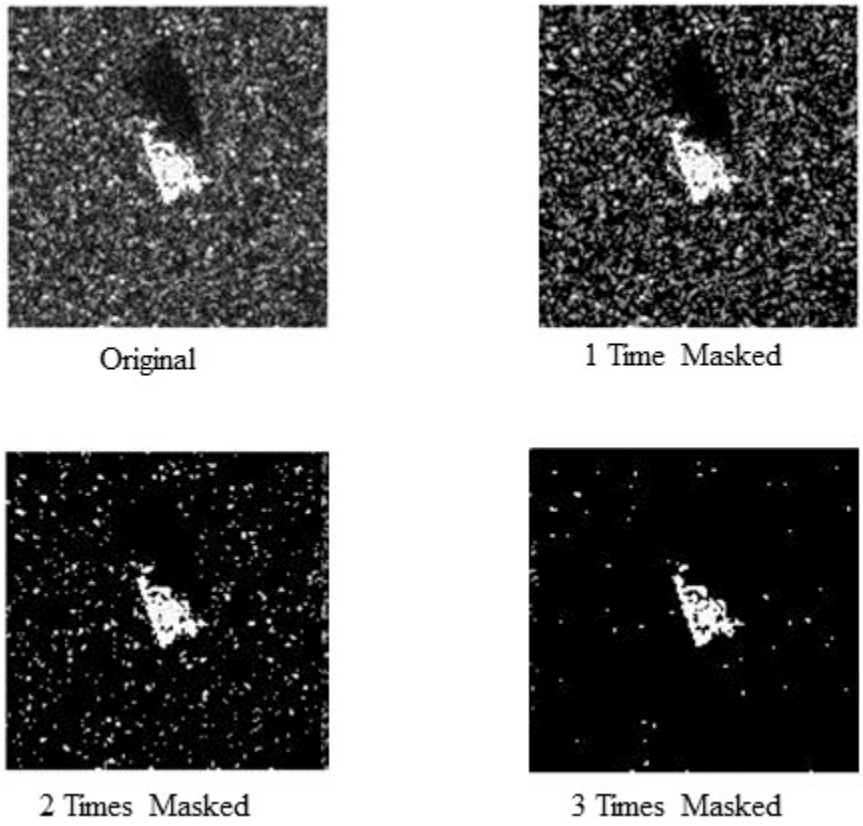

Figure 18. Image Masking

After masking the images, the simulation is run for all three masking cases with the same CNN structure and the same training and testing dataset size. Among all the masking cases, the best results are obtained from the one-time masking over the mean value of the image. With one-time masking, 96.35% accuracy is achieved; two-times masking reaches 92.74%; and three-times masking reaches 88.62% accuracy. The results gathered from this case are given in Table 6.

Table 6. Masking Results

Accuracy Level Without Masking	Accuracy Level with Masking 1 Time	Accuracy Level with Masking 2 Times	Accuracy Level with Masking 3 Times
90.77%	96.35%	92.74%	88.62%

As seen in the results from Table 6, a certain level of masking significantly improved the level of accuracy without changing the size of the dataset or the CNN structure. On the other hand, applying too much masking on the image background may remove shadows entirely, making this feature no longer available to assist in the ATR or classification. Thus, masking three times over the mean value led to a reduced accuracy level.

In Figure 19, the results of the simulation from one-time masking over the mean value of the images are given. In the bottom figure, it is seen that classification confusion between targets is reduced, but most of the confusion still occurs between A32 and A04, just like the previous case.

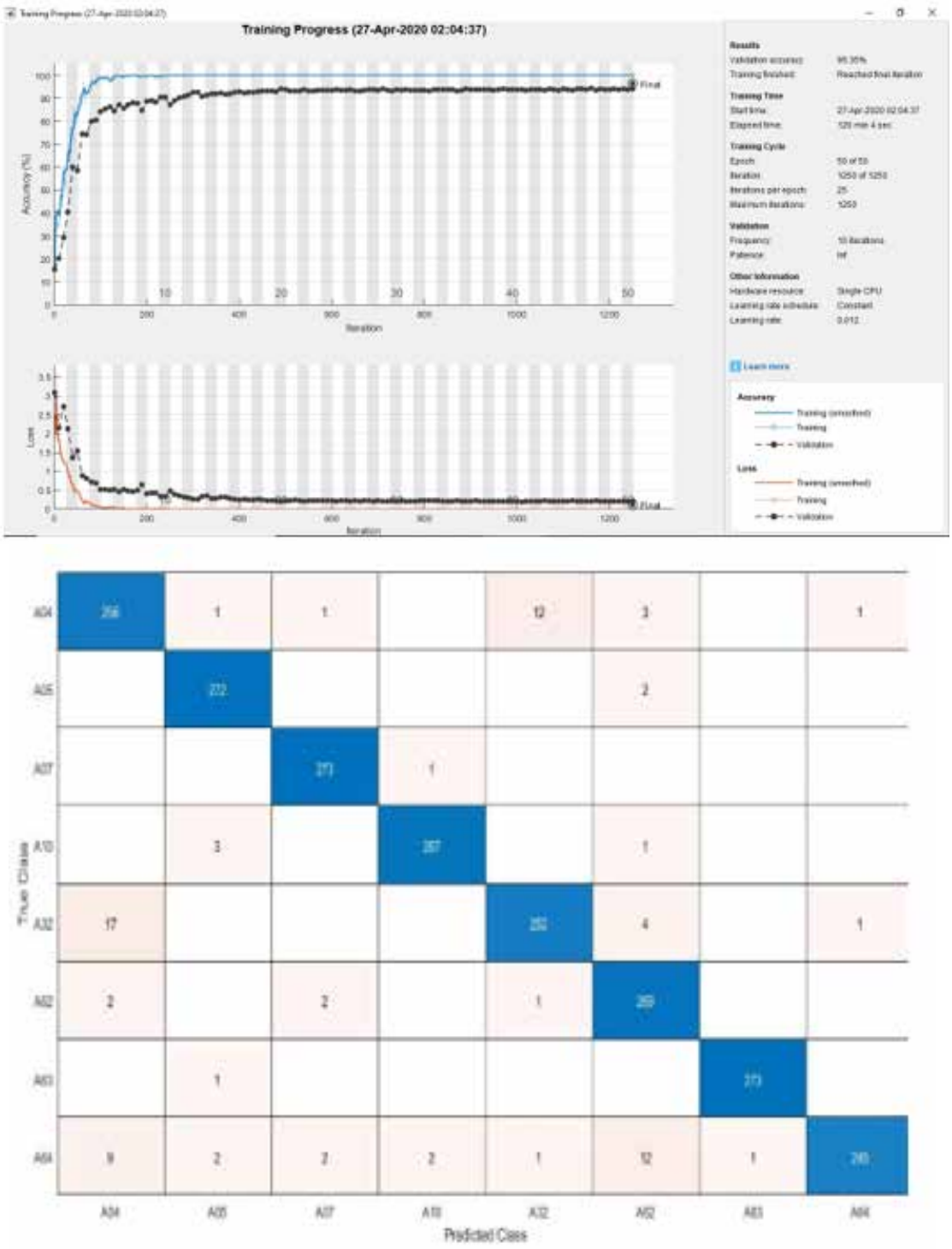

Figure 19. One-Time Masking Training Progress and Confusion Matrix

5. Mixed Targets and T72 Variants Combined Data

In the last part of the experiment, all data, mixed targets and T72 variants, are combined and tested in the CNN structure. In this part, 99.31% accuracy is achieved. The training progress and confusion matrix for this experiment are given in Figure 20.

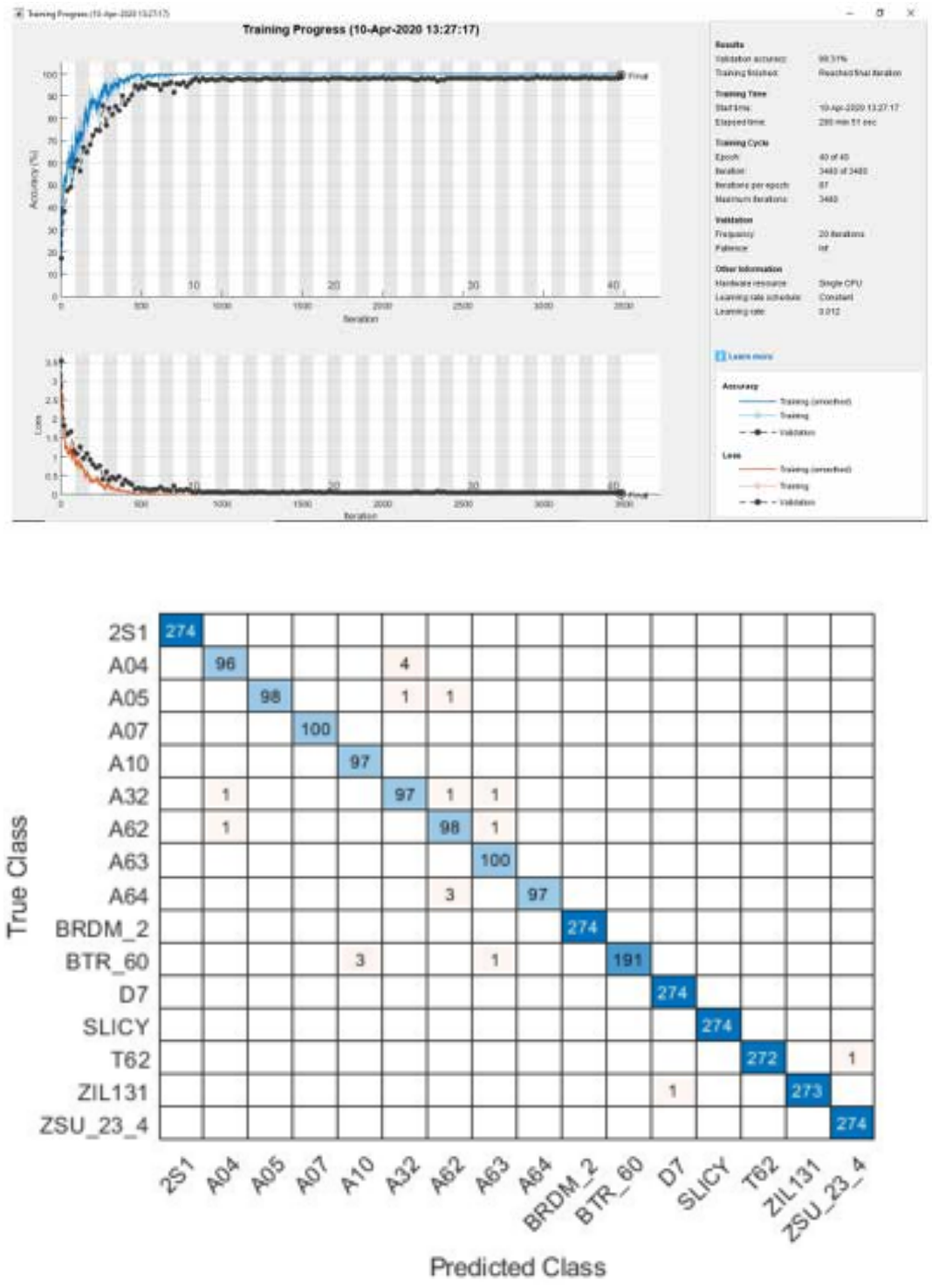

Figure 20. Training Progress and Confusion Matrix for Combined Data

When the confusion matrix is examined thoroughly, it is seen that most of the confusions occur between A32 and A04, just as in the previous experiment. Also, it is evident that most of the confusions occur between T72 variants according to the matrix. A lack of data and the similarities between T72 variants may cause these confusions.

B. ANALYSIS

In this part, simulation results obtained from this study are analyzed first, and then they are compared to each other. Then, in the second part of this section, similar or previous studies about CNN are explained. Lastly, results from this thesis and previous works are compared.

1. Analysis of Results

This study deals with using the prepared CNN algorithm to recognize targets from images available for public access in the MSTAR dataset. Five separate cases are implemented to test the CNN structure created for this experiment. From these experiments, results having a validated accuracy of 90% to 99.76% are observed.

In the first case, the eight targets shown in Table 2 are tested with the CNN structure. These targets, which are military vehicles and tanks, look relatively different from each other. In this part of the experiment, structures composed of different numbers of layers are tested, and the most accurate test result (99.76%) is taken from six, seven, and eight layers. Among these, the CNN structure with seven layers has the best average result, which is 99.68%. Although it achieves a better result than structures with fewer layers, it requires more processing time because every additional layer costs more processing time. For example, while it takes 290 minutes to complete the simulation with a six-layer structure, it takes 310 minutes with seven layers, and 350 minutes with eight layers.

In the second case, the eight targets shown in Table 3 are tested with the CNN structure. Due to the differences between the first and the second cases, the second case has less training data and the targets look alike as all of them are variants of the T72 type tank; only 90.77% of accuracy is achieved in the second case. To increase the accuracy

level, some improvements are tried with the CNN structure, but they did not give better results.

In the third part of the experiment, it was decided to increase the training data to improve the accuracy level and add some data from the test set to the training data. After increasing the training data, shown in Table 4, a 98.49% accuracy level is achieved. It is still less than the first case, but a significant level of improvement is observed. When the confusion matrix related to this test case is examined, it is observed that confusion is concentrated on the A04 and A32 types of tanks. A better result can be obtained with more training data based on these targets. This experiment revealed that the amount of data for CNN is very crucial.

In the fourth part of the study, masking is applied to try to improve the image quality and accuracy level of the test results for variants of the T72. In this section, three masking cases are implemented to the images, and a 96.35% accuracy level is achieved with one-time masking. When it is compared with the experiment done without masking, a major improvement is observed. The accuracy level increases from 90.77% to 96.35%.

In the last case, all 16 targets, shown in Tables 2 and 3, are tested with the CNN structure. In this part of the experiment, a 99.31% accuracy level is obtained. Although it seems better than the second case, the number of confused data is almost the same with the second case for the T72 variants. In this case, better accuracy is achieved because there is more data than in the T72 variants test data. Also, when the confusion matrix is examined, it is evident that A04 and A32 have more confusion than other targets, just like in the second case. Similar to the second case, it is obvious that the amount of data is important for better results.

In the first case, it is seen how important the CNN structure and the organization of it is, such as the number of convolution layers, pooling, and the classification layer. The number of epochs (iterations), learning rate, and validation frequencies are also other factors that affect the results, but these factors mostly affect training time. In this experiment, a personal computer with a single CPU is used.

The other cases showed how the amount and the quality of data is important for obtaining better CNN results. While there are many image databases for CNN experiments, it is difficult to find military SAR images for public use or CNN experiments. Therefore, these experiments are done with limited data. The easiest way to increase accuracy results is by increasing the amount of data. Yet, it is not easy to find military images; thus, some image processing techniques are used to increase the dataset. Masking noise or unwanted parts in the images is one of these techniques. In this study, both increasing the amount of data and masking of images gave better results and improved accuracy.

2. Comparison of Results with Similar and Previous Works

As already noted, CNN is very popular and a widely used method for automatic target classification and recognition. Additionally, the MSTAR dataset is one of the most well-known public datasets for military SAR images. Thus, there have been similar or previous experiments against which to compare the results of this study. Yet, every study or experiment has its own method. While some have tried to improve CNN structure, others have focused on increasing the dataset.

In [5], well-known CNN structures, LeNet, AlexNet, VGGNet16, Inception-3, ResNet18, ResNet101, ResNet152, SE-ResNet18, SE-ResNet10, and DenseNet161, are compared in terms of accuracy and operation time. The simulations are done with 50 epochs and 0.01 learning rate, and the MSTAR dataset is used. The accuracy levels and operation times for each network are given in Table 7.

Table 7. Accuracy Levels of CNN Models. Source: [5].

CNN Model	Accuracy Level	Operation Time (s)
LeNet	85.25 %	61.351
AlexNet	99.15 %	343.631
VGGNet16	99.03 %	2219.282
Inception-3	99.37 %	2055.555
ResNet18	99.95 %	521.882
ResNet101	99.59 %	2370.102
ResNet152	99.41 %	3386.744
SE-ResNet18	100 %	606.193
SE-ResNet101	99.79 %	3192.021
DenseNet161	99.92 %	3451.127

As seen in the table, except for LeNet, over 99% accuracy is reached with all of the CNN models. These accuracy levels correspond to the result seen in this study, which is 99.76%. In the same paper, the operation time of the network is observed, and it is stated that AlexNet, LeNet, and ResNet18 have the shortest operation time. While accuracy levels from that paper match this study, the operation times are far shorter than in this study. Operation times of these networks vary from one minute to 60 minutes; by contrast, it takes approximately 250 minutes in this study. Computer processors are the main contributor to the operation time, so using a GPU instead of a CPU can shorten the time of operation. Therefore, this difference may be caused by using a CPU in this study.

In the other study [36] about classification using CNN, five targets, which are BTR70, BTR60, D7, T62, and ZIL131 from the MSTAR dataset, are tested with an eight-layer CNN structure and a 99.48% accuracy level is achieved, which is similar to this study. In [37], the MSTAR dataset is tested with the ResNet-18 Network and achieved a 99% accuracy level, while in [38] a PCANet model CNN network is used to classify the targets and 99.22% accuracy is achieved. In the paper, the effect of the amount of data on accuracy also is examined. With 300 training data, only 73.15% accuracy is achieved, but with 2,700

training data, 97.40% is obtained. This thesis also showed that increasing the dataset provides a better accuracy level in the third test case.

In [39] the effect of the background of the image on accuracy is tested. In the paper, the background of the SAR image changes with road, farmland, and grassland. Among the experiments, a maximum 98.73% accuracy level is achieved with a road background. In [40], data enhancement is done by adding noise to the original SAR images. Then training data is increased by combining noisy images with the original images. This experiment is done with only three targets, BMP1, T64, and T72. After the simulations, 91% accuracy is achieved with data improvement while without it, only 76.85% accuracy is achieved. Those experiments are similar to the masking done in this study. In this study, masking over the mean value of the images was applied and the accuracy level increased from 90% to 96%.

According to those previous works and experiments, this study reached a state-of-the-art accuracy level of 99.76%, although operation time is higher than for some other experiments. Also, it is obvious that augmenting the images or dataset increases the accuracy level as seen in this study as well as similar works. Based on papers and studies done previously, it is evident that there are many ways to improve the dataset like by adding noise, changing the background, cluttering, and masking. In this study the masking affect is examined, and it showed that with masking, the accuracy level can be improved.

VI. CONCLUSION AND FUTURE WORK

A. CONCLUSION

Recently, the number of sensors in the military operational environment have increased with advances in sensor technology. Sensors are found on various platforms and systems, including enemy targets, air defense systems, airborne early warning radars, fighter jets, etc. As a result, in the operation area it is important to decrease decision and action time to complete the mission successfully. To decrease time of action, automatic target recognition and classification have become a hot and important topic, and consequently, the number of scientific studies on this topic is increasing. With the advent of artificial intelligence tools like machine learning and deep learning, studies about automatic target recognition have accelerated. The robustness and better accuracy of the convolutional neural network have gained the spotlight in these studies.

Regardless of which tool is used, data is the most important aspect of automatic target recognition and classification studies. Thus, many image databases have been created for this purpose. However, as noted earlier in this study, it is hard to find such databases for military images. Fortunately, MSTAR provides military SAR images for public use. This dataset is widely used for CNN applications as it was in this thesis.

This thesis contributes to the theoretical understanding of performance of CNN structures based on the number of layers in those structures. The optimum number of layers is found in terms of accuracy and processing time. In addition, the effect of image background on the accuracy level is investigated. It is seen that by partially darkening the background of the grayscale images, the accuracy level increases without raising the amount of training data.

This study has aimed to build a CNN structure that can classify targets with the highest level of accuracy as possible. First, brief information about synthetic aperture radar and convolutional neural networks was given. Then, using the MSTAR dataset, simulations were performed using CNN structures composed of different numbers of layers.

Several experiments were run to analyze the CNN structure. Among these experiments, a 99.76% accuracy level, which is a state-of-the-art value, was reached with eight-class targets. One of the purposes of this thesis is testing the effect of the quantity and quality of data on accuracy. Thus, the CNN architecture remained the same as the amount of data gradually increased, and as it increased the classification accuracy levels also increased. An 8% difference was observed in the results that are taken from the highest and lowest number of data. To see the effect of augmentation of image, masking was applied to images in the dataset. It is observed that without masking, an accuracy level around 90% is achieved, and with masking, that level rises to around 96%. Therefore, it is seen that in addition to CNN architecture, the quality and quantity of data play a crucial role in obtaining a higher accuracy level in classification problems.

In conclusion, this study deals with automatic recognition of targets available for public access from the MSTAR dataset, using the convolutional neural network algorithm. In this study the classification of targets reached a 99.76% accuracy level. Further, data augmentation is also examined and applied to improve the accuracy level. Therefore, this application can be used where automatic classification and recognition is desired.

B. FUTURE WORK

In this thesis, the CNN algorithm is developed and tested with MATLAB; therefore, these studies were implemented in a simulated environment only. To take this study further, this algorithm could be tested in the real environment with an unmanned or manned air vehicle. Also, the dataset used in this study was insufficient and outdated, so this study could be repeated with a larger and a relatively new dataset. Lastly, this experiment was done using only one CPU; hence, it took more time to finish the classification when compared to other studies. To decrease processing time or learning time, a more powerful processor like one or more GPUs should be used.

APPENDIX: MATLAB CODE

A. CODE FOR CONVOLUTIONAL NEURAL NETWORK [41]

```
close all; clear all;  clc;

%% creating two seperate data storeage. One for Train and other one for
test
imagepath = fullfile('Test');
imagepath1 = fullfile('Train');
ImgData = imageDatastore(imagepath,
'IncludeSubfolders',true,'LabelSource','FolderNames');
ImgData1=imageDatastore(imagepath1,
'IncludeSubfolders',true,'LabelSource','FolderNames');

%% display # of labels and count of each
 ImgData.countEachLabel
 label_class = unique(ImgData.Labels);
 nl = length(label_class);

 ImgData1.countEachLabel
 label_class1 = unique(ImgData1.Labels);
 n2 = length(label_class);

%% Creating CNN Layers
layers = [
    imageInputLayer([128 128 1])

    convolution2dLayer(2,8,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(2,16,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(2,32,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(2,64,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)
```

```
    convolution2dLayer(2,128,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(2,256,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(2,512,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    fullyConnectedLayer(8)
    softmaxLayer
    classificationLayer];


%% Setting input image sizes

imagesize = layers(1).InputSize;
outputSize = imagesize(1:2);
ImgData.ReadFcn = @(loc)imresize(imread(loc),outputSize);
ImgData1.ReadFcn = @(loc)imresize(imread(loc),outputSize);

%% Setting Training Options

options = trainingOptions('sgdm', ...
    'InitialLearnRate',0.01, ...
    'MaxEpochs',100, ...
    'Shuffle','every-epoch', ...
    'ValidationData',ImgData1, ...
    'ValidationFrequency',10, ...
    'Verbose',false, ...
    'Plots','training-progress');

%% Training part
net = trainNetwork(ImgData,layers,options);

%% Classification part
classification = classify(net,ImgData1);
validation = ImgData1.Labels;
accuracy = sum(classification == validation)/numel(validation)

%% Creating Confusing Matrix
cm = confusionchart(validation,classification);
```

B. CODE FOR MASKING

```
close all; clear all; clc

%%Creating Path

imagepath = fullfile('BTR_60');
ImgStore = imageDatastore(imagepath,
'IncludeSubfolders',true,'LabelSource','FolderNames');

%% Masking all images for BTR_60 class target

fld='C:\Users\serkan\Documents\MATLAB\BTR_60';
path = dir(fullfile(fld, '*.jpg'));

for i=1:numel(path)
    img = readimage(ImgStore,i);
    imgd= im2double(img); % converting image to double
    mn= mean(imgd(:));    % findind mean value of the image
    mask_matrix = (imgd > 2.*mn); %creating masking matrix
    imgn =mask_matrix.*imgd;  % new image
    filename = sprintf('imgn%02d.jpg', i);
    imwrite(imgn, filename)
end
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] C. Wiley, “Synthetic aperture radars,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. AES-21, no. 3, pp. 440–443, May 1985, doi: 10.1109/TAES.1985.310578

[2] S. Chen and H. Wang, “SAR target recognition based on deep learning,” in *2014 International Conference on Data Science and Advanced Analytics (DSAA)*, Shanghai, China, Oct. 2014, pp. 541–547, doi: 10.1109/DSAA.2014.7058124

[3] Y. Wang, Y. Zhang, H. Qu, and Q. Tian, “Target detection and recognition based on convolutional neural network for SAR image,” in *2018 11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, Beijing, China, Oct. 2018, pp. 1–5, doi: 10.1109/CISP-BMEI.2018.8633151

[4] T. Guo, J. Dong, H. Li, and Y. Gao, “Simple convolutional neural network on image classification,” in *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)(*, Beijing, China, Mar. 2017, pp. 721–724, doi: 10.1109/ICBDA.2017.8078730

[5] J. Shao, C. Qu, and J. Li, “A performance analysis of convolutional neural network models in SAR target recognition,” in *2017 SAR in Big Data Era: Models, Methods and Applications (BIGSARDATA)*, Beijing, Nov. 2017, pp. 1–6, doi: 10.1109/BIGSARDATA.2017.8124917

[6] K. Jaiswal and D. Kalpeshbhai Patel, “Sound classification using convolutional neural networks,” in *2018 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, Bangalore, India, Nov. 2018, pp. 81–84, doi: 10.1109/CCEM.2018.00021

[7] V. Swaminathan, S. Arora, R. Bansal, and R. Rajalakshmi, “Autonomous driving system with road sign recognition using convolutional neural networks,” in *2019 International Conference on Computational Intelligence in Data Science (ICCIDS)*, Chennai, India, Feb. 2019, pp. 1–4, doi: 10.1109/ICCIDS.2019.8862152

[8] I. Almakky, V. Palade, and A. Ruiz-Garcia, “Deep convolutional neural networks for text localisation in figures from biomedical literature,” in *2019 International Joint Conference on Neural Networks (IJCNN)*, Budapest, Hungary, Jul. 2019, pp. 1–5, doi: 10.1109/IJCNN.2019.8852353

[9] Z. Hu and E.-J. Lee, “Human motion recognition based on improved 3-dimensional convolutional neural network,” in *2019 IEEE International Conference on Computation, Communication and Engineering (ICCCE)*, Fujian P.R, China, Nov. 2019, pp. 154–156, doi: 10.1109/ICCCE48422.2019.9010816

[10] Q. Weilei, Z. Xinggan, and F. Ge, “An automatic target recognition algorithm for SAR image based on improved convolution neural network,” in *2017 12th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, Siem Reap, Jun. 2017, pp. 551–555, doi: 10.1109/ICIEA.2017.8282905

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017, doi: 10.1145/3065386

[12] E. R. Keydel, S. W. Lee, and J. T. Moore, “MSTAR extended operating conditions: A tutorial,” Orlando, FL, Jun. 1996, pp. 228–242, doi: 10.1117/12.242059

[13] M. Rajnoha, R. Burget, and L. Povoda, “Image background noise impact on convolutional neural network training,” in *2018 10th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, Moscow, Russia, Nov. 2018, pp. 1–4, doi: 10.1109/ICUMT.2018.8631242

[14] Y. K. Chan and V. C. Koo, “An introduction to synthetic aperture radar (SAR),” *Progress in Electromagnetics Research* B, vol. 2, pp. 27–60, 2008 .

[15] C. Wolff, “Synthetic aperture radar.” Accessed Jun. 9, 2020. [Online]. Available: https://www.radartutorial.eu/20.airborne/ab07.en.html

[16] F. Neri, *Introduction to Electronic Defense Systems*, 3rd ed. Boston: Artech House, 2018.

[17] M. I. Skolnik, Ed., *Radar Handbook*, 3rd ed. New York: McGraw-Hill, 2008.

[18] A. Moreira, P. Prats-Iraola, M. Younis, G. Krieger, I. Hajnsek, and K. P. Papathanassiou, “A tutorial on synthetic aperture radar,” *IEEE Geosci. Remote Sens. Mag.*, vol. 1, no. 1, pp. 6–43, Mar. 2013, doi: 10.1109/MGRS.2013.2248301

[19] E. Podest, “Basics of synthetic aperture radar (SAR),” NASA’s Applied Remote Sensing Training Program, pp. 1–56. Nov. 2017. [Online]. Available: https://appliedsciences.nasa.gov/what-we-do/capacity-building/arset

[20] K. D. Foote, “A brief history of deep learning,” Dataversity, Feb. 7, 2017. [Online]. Available: https://www.dataversity.net/brief-history-deep-learning/#

[21] K. Fukushima, “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position,” *Biol. Cybern.*, vol. 36, no. 4, pp. 193–202, Apr. 1980, doi: 10.1007/BF00344251

[22] S. Loussaief and A. Abdelkrim, “Convolutional neural network hyper-parameters optimization based on genetic algorithms,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 10, 2018, doi: 10.14569/IJACSA.2018.091031

[23] D. Steinkraus, I. Buck, and P. Y. Simard, “Using GPUs for machine learning algorithms,” in *Eighth International Conference on Document Analysis and Recognition (ICDAR’05)*, Seoul, South Korea, 2005, vol. 2, pp. 1115–1120, doi: 10.1109/ICDAR.2005.251

[24] S. N. Aslan, A. Ucar, and C. Guzelis, “Fast object recognition for humanoid robots by using deep learning models with small structure,” in *2020 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, Novi Sad, Serbia, Aug. 2020, pp. 1–7, doi: 10.1109/INISTA49547.2020.9194644

[25] K. O’Shea and R. Nash, “An introduction to convolutional neural networks,” *ArXiv151108458 Cs*, Dec. 2015. [Online]. Available: http://arxiv.org/abs/1511.08458

[26] “Visual modeling of data using convolutional neural networks,” *Int. J. Eng. Adv. Technol.*, vol. 9, no. 1, pp. 4938–4942, Oct. 2019, doi: 10.35940/ijeat.A2084.109119

[27] S. Hijazi, R. Kumar, and C. Rowen, “Using convolutional neural networks for image recognition,” p. 12., 2015. [Online]. Available: https://ip.cadence.com/uploads/901/cnn_wp-pd

[28] M. Alawad and M. Lin, “Stochastic-based deep convolutional networks with reconfigurable logic fabric,” *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 2, no. 4, pp. 242–256, Oct. 2016, doi: 10.1109/TMSCS.2016.2601326

[29] Ujjwalkarn, “An intuitive explanation of convolutional neural networks,” The Data Science Blog, Aug. 11, 2016. [Online]. Available: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

[30] N. Cui, “Applying gradient descent in convolutional neural networks,” *J. Phys. Conf. Ser.*, vol. 1004, p. 012027, Apr. 2018, doi: 10.1088/1742-6596/1004/1/012027

[31] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, “Convolutional neural networks: An overview and application in radiology,” *Insights Imaging*, vol. 9, no. 4, pp. 611–629, Aug. 2018, doi: 10.1007/s13244-018-0639-9

[32] U.S. Air Force, “MSTAR public targets,” Sensor Data Management System. Accessed Jan. 21, 2020. [Online]. Available: https://www.sdms.afrl.af.mil/index.php?collection=mstar

[33] S. Eppel, “Classifying a specific image region using convolutional nets with an ROI mask as input.” [Online]. Available: https://arxiv.org/pdf/1812.00291

[34] J. Bullock, C. Cuesta-Lazaro, and A. Quera-Bofarull, “XNet: A convolutional neural network (CNN) implementation for medical x-ray image segmentation suitable for small datasets,” in *Medical Imaging 2019: Biomedical Applications in Molecular, Structural, and Functional Imaging*, San Diego, United States, Mar. 2019, p. 69, doi: 10.1117/12.2512451

[35] B. Kayalibay, G. Jensen, and P. van der Smagt, “CNN-based segmentation of medical imaging data,” *ArXiv170103056 Cs*, Jul. 2017. Accessed Sep. 04, 2020. [Online]. Available: http://arxiv.org/abs/1701.03056

[36] Q. Liu, S. Li, S. Mei, R. Jiang, and J. Li, “Feature learning for SAR images using convolutional neural network,” in *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, Valencia, Spain, Jul. 2018, pp. 7003–7006, doi: 10.1109/IGARSS.2018.8519159

[37] R. J. Soldin, “SAR target recognition with deep learning,” in *2018 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, Washington, DC, USA, Oct. 2018, pp. 1–8, doi: 10.1109/AIPR.2018.8707419

[38] B. Qi, H. Jing, H. Chen, Y. Zhuang, Z. Yue, and C. Wang, “Target recognition in synthetic aperture radar image based on PCANet,” *J. Eng.*, vol. 2019, no. 21, pp. 7309–7312, Nov. 2019, doi: 10.1049/joe.2019.0238

[39] Y. Zhou *et al.*, “Complex background SAR target recognition based on convolution neural networks,” in *2019 6th Asia-Pacific Conference on Synthetic Aperture Radar (APSAR)*, Xiamen, China, Nov. 2019, pp. 1–4, doi: 10.1109/APSAR46974.2019.9048279

[40] C. Belloni, N. Aouf, J.-M. L. Caillec, and T. Merlet, “SAR specific noise based data augmentation for deep learning,” in *2019 International Radar Conference (RADAR)*, Toulon, France, Sep. 2019, pp. 1–5, doi: 10.1109/RADAR41533.2019.171310

[41] MathWorks, Natick, MA, USA. 2020. “Deep learning toolbox.” [Online]. Available: https://www.mathworks.com/help/deeplearning/index.html?category=index&s_tid=CRUX_topnav

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California