12-2020

Automatic target recognition with convolutional neural networks.

Nada Baili
University of Louisville

AUTOMATIC TARGET RECOGNITION WITH CONVOLUTIONAL NEURAL NETWORKS

By

Nada Baili
B.E. Polytechnic Engineer, Tunisia Polytechnic School, 2019

A Thesis
Submitted to the Faculty of the
J.B. Speed School of Engineering
in Partial Fulfillment of the Requirements
for the Degree of

Master of Science in Computer Science

Department of Computer Engineering and Computer Science
University of Louisville
Louisville, Kentucky

December 2020

AUTOMATIC TARGET RECOGNITION WITH CONVOLUTIONAL NEURAL NETWORKS

By

Nada Baili
B.E. Polytechnic Engineer, Tunisia Polytechnic School, 2019

A Thesis Approved On

11/24/2020
Date

By the following Thesis Committee:

Hichem Frigui, Ph.D., Thesis Director

Olfa Nasraoui, Ph.D.

Amir Amini, Ph.D.

ACKNOWLEDGEMENTS

This thesis would not have been possible without the support of many people. I would like to express my deepest gratitude to my supervisor Prof. Hichem Frigui for giving me the precious opportunity to carry out my master thesis under his guidance. His intellectual creativity and analysis have been of great inspiration for my work.

Thanks to the jury members who spared some time to read my thesis and who showed interest to come and attend the presentation.

I address, likewise, my thanks to my lab-mates at the Multimedia Research Lab for their support and friendship.

Last but not least, I would be forever grateful for the support and the unconditional love I have had from my parents, my siblings and my friends.

ABSTRACT

AUTOMATIC TARGET RECOGNITION WITH CONVOLUTIONAL NEURAL NETWORKS

Nada Baili

November 24, 2020

Automatic Target Recognition (ATR) characterizes the ability for an algorithm or device to identify targets or other objects based on data obtained from sensors, being commonly thermal. ATR is an important technology for both civilian and military computer vision applications. However, the current level of performance that is available is largely deficient compared to the requirements. This is mainly due to the difficulty of acquiring targets in realistic environments, and also to limitations of the distribution of classified data to the academic community for research purposes.

This thesis proposes to solve the ATR task using Convolutional Neural Networks (CNN). We present three learning approaches using WideResNet-28-2 [1] as a backbone CNN. The first method uses random initialization of the network weights. The second method explores transfer learning. Finally, the third approach relies on spatial transformer networks [2] to enhance the geometric invariance of the model. To validate, analyze and compare our three proposed models, we use a large-scale real benchmark dataset that includes civilian and military vehicles. These targets are captured at different viewing angles, different resolutions, and different times of the day. We evaluate the effectiveness of our methods by studying their robustness to realistic case scenarios where no ground truth data is available and targets are automatically detected. We show that the method that uses spatial transformer networks achieves the best results and demonstrates the most robustness to various perturbations that can be encountered in real applications.

TABLE OF CONTENTS

LIST OF TABLES

TABLE **Page**

LIST OF FIGURES

FIGURE **Page**

CHAPTER 1

INTRODUCTION

Over many decades, Automatic Target Recognition (ATR) has been one of the key components of autonomous vehicle missions in the context of defense systems [8]. The ATR system performs automatic target detection, identification, and tracking by processing a sequence of images captured from sensors, commonly being infrared. The goal is to perform these functions in real time and to be able to adapt to dynamic situations and different environments.

The ATR system has successfully and effectively removed the man from the process of target acquisition and recognition, whose intervention is generally slow, unreliable, vulnerable, and may limit the performance of the overall system or mission in real situations. The applications of ATR systems are numerous. They can be used to identify objects such as ground and air vehicles, as well as living targets such as animals, humans, and vegetative clutter. Possible military scenarios can take advantage of this application when attempting to identify an object on a battlefield. There has been more and more interest shown in using ATR for domestic applications as well, such as for border security, safety systems to identify objects or people on a subway track, automated vehicles, and many others.

ATR systems feed on infrared images as they offer fundamental advantages over regular imaging solutions [9]. Thermal imaging has been a boon to the ATR research community because of its ability to perform well in all weather conditions and all times of the day and night. Infrared imaging systems can perform well in low-light and low-visibility situations. This is critical for outdoor applications, such as the ATR task, where light and visibility are a constant variable, and especially useful in security applications where no camouflage can fool a thermal camera.

Despite their numerous advantages, infrared images pose several challenges that render the construction of a reliable and robust ATR system a tricky task. In fact, the infrared radiation is highly sensitive to several factors, such as meteorological conditions and sensor calibration. Targets are also captured from different viewpoints, different ranges and different times of the day. Moreover, cold or stationary vehicles may blend with the background or be partially masked by vegetation or

engine smoke, which makes the ATR task more difficult. Therefore, ATR systems are expected to be robust against various perturbations related to the object variabilities and the environment conditions.

Most early work [10–14] include model-based approaches that can typically be decomposed into five main steps: preprocessing, detection, feature extraction, feature selection and classification. These classic approaches require significant human intervention and supervision. Considering the challenges posed by the infrared imagery, hand-crafted features become too shallow and too tricky to extract in order to build a reliable and robust ATR system. Recent research on ATR has shifted to rely on deep learning [15–18].

Deep Learning is a new field of machine learning that has recently emerged to become one of the most popular scientific research trends. This new field has witnessed rapid progress since its appearance and it has already revolutionized a variety of domains, especially computer vision [19]. Deep learning based algorithms use neural networks to mainly address the issue of shallow learning algorithms, and to improve and automatize the step of feature extraction. The reason behind their success is that they attempt to automatically unfold hierarchies of abstraction embedded in observed data by elaborately designing the layers of neurons in depth and width. By doing so, deep neural networks outperformed their classic counterparts by a huge margin [20].

Deep learning architectures have different variants. In particular, the Convolutional Neural Network (CNN) attracted the most interest from the research community for its outstanding classification performance on large-scale datasets, such as ImageNet [21]. The power of CNNs reside in their ability to learn hierarchical image representations using its multi-layer feed-forward structure. This results in the extraction of high quality features that can describe the image at different levels. The low level features describe generic information about the image, such as the edges and corners. As opposed to high-level features that describe peculiarities in the image related to the task in hand, such as object parts.

Similar to many other research areas, the ATR field has been highly impacted by the deep learning predominance. Several works have been recently published attempting to apply CNNs on the ATR task [15–18]. Despite the promising results that have been achieved, many limitations are hindering the progress of deep learning based models in the ATR context. These challenges are mainly related to the lack of ground-truthed large-scale datasets suitable for the ATR task. This can be explained by the fact that most ATR systems have been developed for confidential military missions where the acquisition and the free publication of the appropriate data can be

either strenuous or require authorization from the responsible parties.

In this thesis, we investigate the application of a state of the art CNN (*WideResNet-28-2* [1]) to address target identification and recognition in infrared images. We explore three different CNN-based learning approaches. The first method uses the CNN initialized with random weights. The second approach uses transfer learning. Finally, we propose a novel approach that consists of a Spatial Transformer Network (STN) [2] block inserted within the CNN. We evaluate the three suggested CNN variants on the Defense Systems Information Analysis center (DSIAC) benchmark data set. Given the targeted operational contexts, we analyze and compare the robustness of the three suggested CNN variants against possible perturbations introduced by the detection stage. To simulate such behaviour, we train the three CNNs on images with centered targets and test them on translated or scaled inputs, and also on targets detected by the state of the art object detector YOLO [7].

Our analysis is guided by the following Research Questions (RQs) that we investigate through our experiments:

- **RQ1:** Does preprocessing of the input images improve the robustness of the ATR?
- **RQ2:** How well can the learned models generalize to targets captured at different resolutions?
- **RQ3:** Are the learned models invariant to small scale and shift variations?
- **RQ4:** Can the models reject the non-targets identified by the detector as potential targets?

The remainder of this thesis is structured as follows. Chapter 2 illustrates the concept of deep learning and lays out different successful architectures in the context of our ATR application. It also reviews the previous work conducted in the ATR field. In chapter 3, we describe our different CNN-based learning approaches, as well as the dataset we will use in our experiments. In chapter 4, we present and analyze our experimental results by answering the outlined research questions. Finally, in chapter 5, we conclude by summarizing the work, and suggesting possible future directions of research.

CHAPTER 2

BACKGROUND AND LITERATURE REVIEW

In this chapter, we review existing approaches in areas that are highly relevant to our proposed ATR application. We start with presenting the area of deep learning. Then, we go over successful neural network architectures that are relevant to our scope of research. We also present learning concepts in the context of deep learning that will be adopted in our experiments. Finally, we describe the task of Automatic Target Recognition, its major challenges, and the previous works that have been conducted to address these challenges.

2.1 Deep Learning

For many decades, the task of image classification in the field of computer vision was solely relying on hand-engineered features such as SIFT [22], HOG [23], and Fisher Vector [24]. However, these techniques require a considerable amount of pre-processing work and usually result in capturing shallow features. These low-level extracted features may not be effective for most real applications.

Recently, neural networks witnessed significant advances which led the path to the development of Deep Learning [25], thus revolutionizing the field of computer vision. Deep learning is a branch of machine learning that focuses on training deep neural networks, which are neural networks that encompass multiple hidden layers (Fig. 2.1). The introduction of deep learning in the task of image classification enabled it to take on several challenges and significantly improve its classification performance on large-scale data.

The interesting aspect about deep learning is that it cancels the manual feature engineering because neural networks can learn features on their own from training data. The success behind deep learning models is mainly due to the way they represent observed data. In fact, these algorithms perform hierarchical abstraction at many levels, by building on lower level features to generate more abstract information as the level moves further up. This process enables a sophisticated data representation in high dimension.

There are numerous achievements accomplished by deep neural networks in the context of

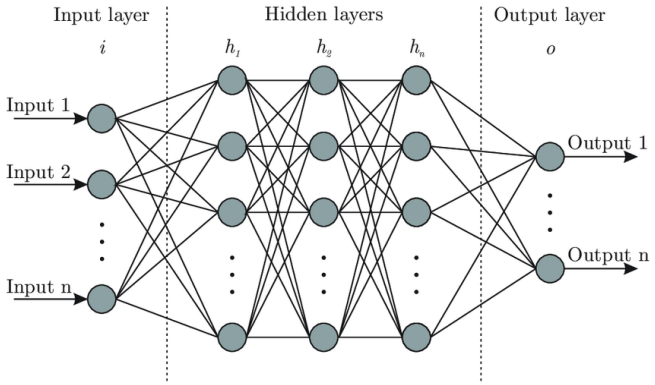

Figure 2.1: Artificial neural network architecture (source: [3])

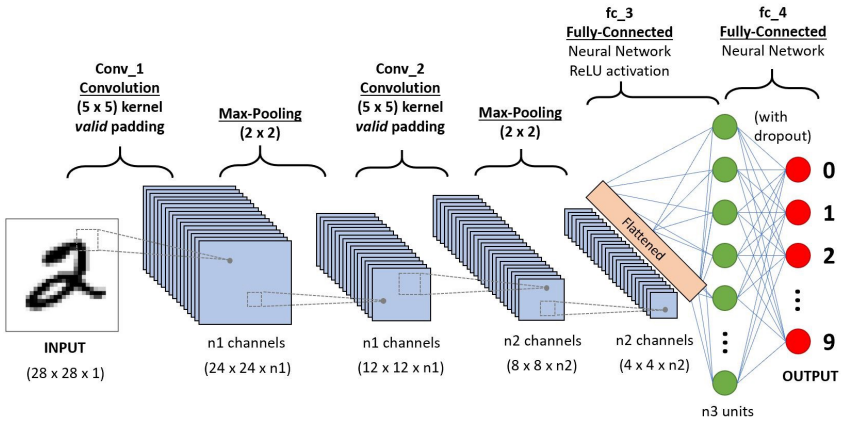

Figure 2.2: Architecture of a convolutional neural network (source: [4])

image classification, among which we can cite the large scale image classification record with the ImageNet database [26] and the DeepFace face recognition method by Facebook [27].

2.1.1 Convolutional Neural Network (CNN)

The first challenge encountered when dealing with deep neural networks is the necessity to learn a huge number of parameters, thus the need for large scale training datasets. A convolutional neural network (CNN) is a neural network architecture that is able to reduce the number of parameters by making small regions share the learnt weights, also called *filters* or *kernels*. While in traditional learning filters are hand-engineered, CNNs have the ability, with enough training, to learn filters that capture the spatial and temporal dependencies in an image.

The typical architecture of a CNN usually consists of convolutional layers, pooling layers and fully-connected layers. An example CNN architecture is displayed in Fig. 2.2 and outlined in the following subsections.

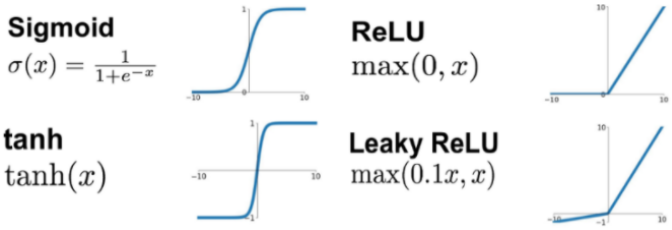

Figure 2.3: Example activation functions (source: [5])

2.1.1.1 Convolutional Layer

A convolutional layer is what distinguishes the CNN from the rest of the artificial neural networks. The input data is convolved using multiple kernels to extract features, and then the output is passed to an activation function to generate feature maps. These feature maps serve as the input for the next hidden layer.

Let x_i^l denote the i^{th} input feature map of layer l, k_{ij}^l the kernel connecting the j^{th} feature map of the output layer to the i^{th} feature map of the input layer and b_j^l an additive bias. Hence we have

$$x_j^l = f(\sum_i x_i^{l-1} * k_{ij}^l + b_j^l) \tag{2.1}$$

By repeating this process multiple times across the convolutional layers of the CNN, we are able to learn progressive levels of features. The extracted features can be categorized into two types: low-level and high-level. The low-level features result from the bottom layers of the CNN, and they usually describe edges, lines, corners, ...etc. The high-level features come from the top layers of the CNN, and they describe fine details and more peculiar information like object parts.

2.1.1.2 Activation Function

An activation function is a mathematical equation that determines the output of a convolutional layer. It is attached to each neuron in the neural network and decides whether it should be activated or not, depending on its value. Activation functions also help normalize the output of the neurons to a specific range. An important characteristic of activation functions is that they have to be computationally efficient since they will be calculated across thousands of neurons for each input sample. There are several activation functions that can be used in the context of CNNs. Fig. 2.3 illustrates the most commonly used activation functions: Sigmoid, TanH, ReLU, and Leaky ReLU.

2.1.1.3 Pooling Layer

Pooling layers work on progressively reducing the spatial size of the output feature map, thus reducing the amount of parameters in the network. The operation can be formally denoted as

$$x_j^l = f(x_j^{l-1}) \tag{2.2}$$

where f is the down-sampling function. The most commonly used pooling operations are max pooling and average pooling.

2.1.1.4 Fully Connected Layer

Unlike convolution layers where the neurons are connected only to a local region in the input, the neurons in the fully connected layer are connected to all the neurons of the input. Fully-connected layers act as linear classifiers in a CNN. They proceed by flattening the features extracted from the CNN, and then using this new representation, they are able to predict the probabilities of the mutually exclusive categories, in which the highest one is corresponding to the predicted category.

2.1.2 CNN Architectures

While the early CNN network architectures simply consist of stacked convolutional layers, more recent architectures explore new and innovative ways for constructing convolutional layers in a way which allows for more efficient learning. Architectures, like ResNet [28], are a breakthrough idea that enables building very deep neural networks.

2.1.2.1 Residual Neural Network (ResNet)

ResNet is a CNN that was introduced in 2016 to solve the issue of vanishing or exploding gradients. This problem arises during backpropagation in deep networks and it has been brought to the attention by noticing that deep networks often result in a higher error rate than their shallower counterparts. He et al. [28] proposed a remedy to this degradation problem by introducing *residual blocks* in which intermediate layers of a block learn a residual function with reference to the block input (Fig. 2.4). The goal of the residual block is to learn how to adjust the input feature map for higher quality features. In case there are no new features to learn, the intermediate layers can simply adjust their weights toward zero such that the residual block can at least represent the

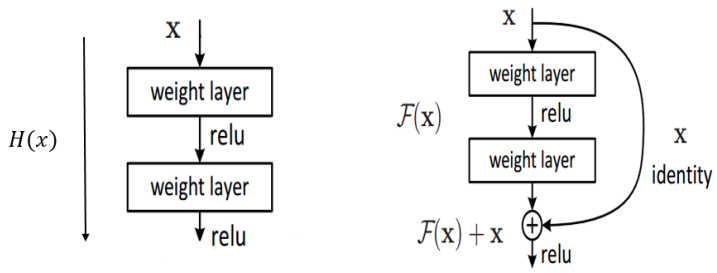

Figure 2.4: Classic CNN vs. residual block

identity function. In such scenario, classic deep networks fail because their corresponding layers are always expected to learn new and distinct features.

2.1.2.2 Wide-Residual Neural Network (WRN)

A Wide-ResNet [1] is an improved version of a ResNet that has been introduced in 2017. The motivation behind WRNs relies on the observation that scaling up ResNets leads to a slow training along with an insignificant gain in performance. Therefore, the authors in [1] suggested to widen the regular ResNet, by increasing the number of feature maps (Fig. 2.5), in order to obtain shallow networks that can achieve a better accuracy than the regular ResNet while taking less training time. The goal of WRNs is mainly to boost the representational power of ResNet blocks by adding more convolutional layers per block, widening the convolutional layers by increasing the number of feature maps, and also increasing the filter sizes. What makes WRNs faster to train is that the GPU becomes more efficient on parallel computing with wider layers.

A WRN is characterized by two parameters l and k. l is a deepening factor that describes the number of convolutions in a block, while k is the widening factor that multiplies the number of features in convolutional layers. Therefore, a WRN can be entirely described by the notation *WRN-l-k*. In the rest of the thesis, we use the notation *WRN-n-k*, where n characterizes the total number of convolutional layers.

2.1.3 Transfer Learning

One of the most powerful ideas in deep learning is that you can take knowledge that a neural network has learned from one task, and apply it to a separate task. For example, if one neural network has learned to recognize objects, this knowledge can be used to improve the performance of an algorithm that detects and recognizes regions of interest in X-ray scans. This process is called

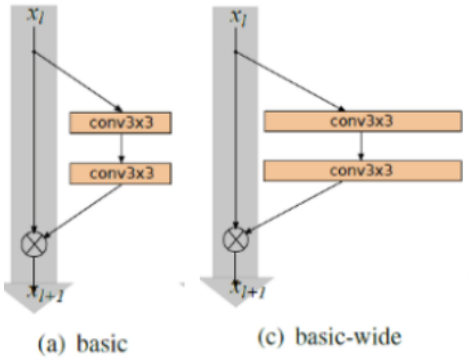

Figure 2.5: Residual block vs. Wide residual block. The basic wide residual block displays more feature maps than the basic residual block.

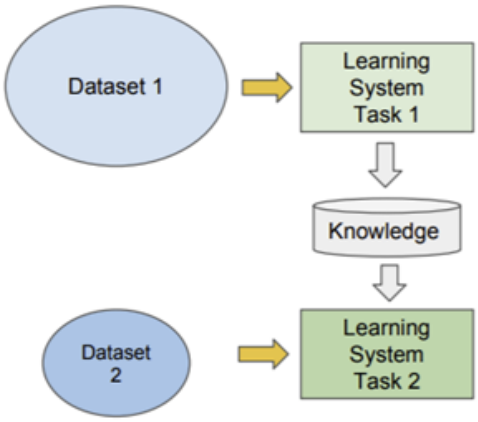

Figure 2.6: Illustration of the transfer Learning concept [6]

Transfer Learning [29].

When building a computer vision application, rather than training the network from scratch using random initializations, it is often faster and more effective to train the network using a model trained on a different domain, in order to transfer the knowledge from one task to a new one (Fig. 2.6). In fact, training a deep neural network from scratch requires a very large training data, takes a long time to train, and requires multiple GPUs. So, using open-source weights that have already been gone through the long and demanding learning procedure, is an efficient way to initialize the training process of a new network.

In practice, transfer learning simply consists in omitting the last fully-connected layer in the pretrained model, since this layer is responsible for the classification. A new fully-connected layer should be created to adapt to the peculiar classes of the new task. Afterwards, there are three different approaches to carry out the training process:

- The first approach consists of having all intermediate layers frozen, which means the parameters will not be retrained and the network will keep the transferred pre-trained weights. Therefore, only the last newly-created fully-connected layer will be trained. In this case, the pre-trained model is used as a *feature extractor*, which means that the new task uses the features extracted by the original pre-trained model, to perform a classification on a new set of classes.

- The second approach consists of having all the layers of the network retrained and the transferred weights are only used as an initialization. In this case, the pre-trained model is said to be *fine-tuned* on a different dataset to serve a new task.

- The third approach combines the first two methods by partially freezing the layers of the network and retraining the remaining ones. Usually, the bottom layers are frozen because they are dedicated to extracting low-level features, which are common to different classification tasks. However, the top layers, which extract more specific and task-related features, are the ones that are often retrained.

2.1.4 Spatial Transformer Network (STN)

Spatial Transformer Networks (STNs) were introduced in 2015 by Google DeepMind [2]. A spatial transformer is a differentiable module that can be inserted anywhere between the layers of a CNN, allowing the spatial manipulation of data within the network. STNs fall within the category of attention models since networks which include spatial transformers are able to select the regions of the image that are most relevant (attention). STNs can also help crop out and scale-normalize the selected regions, which can simplify the subsequent classification task, leading to improved classification performance. Since the transformation is differentiable, gradients are able to flow through the STN during backpropagation in order to update the weights of the transformation parameters as well as the feature map input. Fig 2.7 presents an example of how an STN performs a geometric transformation to an input image.

STNs are composed of three main building blocks (Fig. 2.8): *localization network*, *grid generator* and *sampler*. The localization block is a neural network that can either be a stack of fully-connected layers or convolution layers. Its goal is to learn the parameters of the transformation to apply to the input feature map. Let f_{LocNet} denote the localization network. As illustrated in (2.3), the function takes as input the feature map $U \epsilon \Re^{H \times W \times C}$ with width W, height H and C channels. It outputs $\theta = \{\theta_i\}_{i \epsilon (2N)}$, the parameters of the transformation, where $2N$ is the number of the

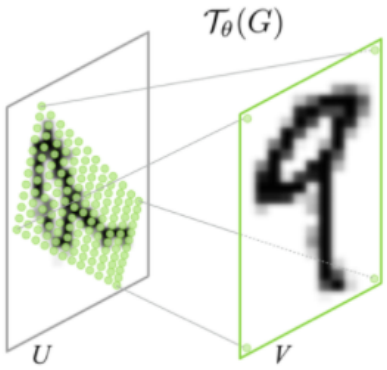

Figure 2.7: Demonstration of the spatial manipulation of the input image by a spatial transformer network. [2]

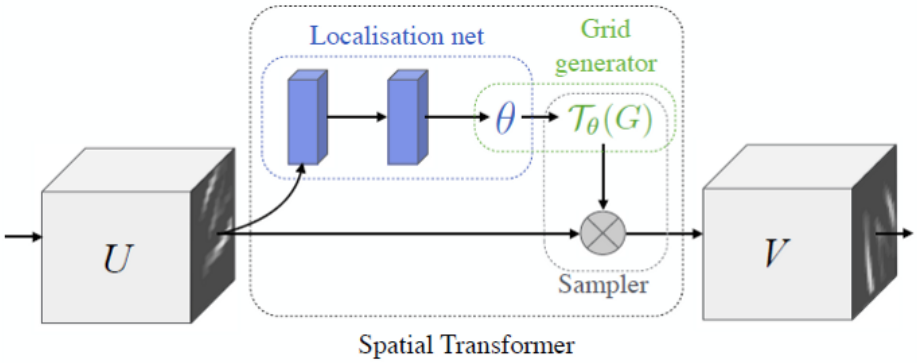

Figure 2.8: Spatial Transformer Network. [2]

transformation parameters to be learned by f_{LocNet}. This term is expressed as an even number because the transformation is applied to a bidimensional input feature map.

$$\theta = f_{LocNet}(U) \tag{2.3}$$

The second part of a spatial transformer is the grid generator. The grid generator starts by creating a regular grid $G = \{G_i\}$ of pixels $G_i = (x_i^t, y_i^t)$ which represent the pixel location coordinates in G. The created grid forms an output feature map $V \epsilon \Re^{H' \times W' \times C}$, where H' and W' are the height and width of the grid, and C is the number of channels. As illustrated in (2.4), the goal of the grid generator is to map the target pixel coordinates (x_i^t, y_i^t) of the created grid G to the source pixel coordinates (x_i^s, y_i^s) using the transformation T_θ . For clarity of exposition, T_θ is assumed to be a 2D affine transformation described in the matrix A_θ. This affine transformation allows cropping, translation, rotation, scale, and skew to be applied to the input feature map, and requires only 6 parameters (the 6 elements of A_θ) to be produced by the localisation network. It can be noted that the generated target grid does not necessarily have the same height and width as the input source

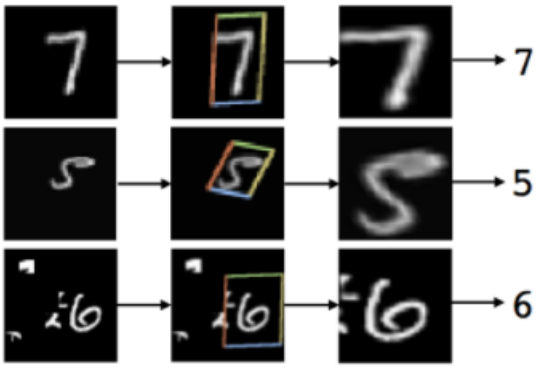

Figure 2.9: Application of STNs on Distorted MNIST dataset. [2]

grid, which allows to scale up or down the input feature map.

$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = T_\theta(G_i) = A_\theta \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} \tag{2.4}$$

The third block of the spatial transformer is the sampler. Given the input feature map and the parametrised sampling grid, the sampler uses a sampling kernel to compute the pixel intensity of the output feature map V. The only important characteristic that the sampling kernel must have is differentiability so that it can allow the loss gradients to flow all the way back to the localization network. A bilinear sampling kernel is shown in (2.5), where V_i^c is the output value for pixel i on channel c, U_{nm}^c denotes the input pixel at coordinate (n, m), and the two max functions determine the relative weight assigned for each pixel to contribute to V_i^c.

$$V_i^c = \sum_n^H \sum_m^W U_{nm}^c max(0, 1 - |x_i^s - m|) max(0, 1 - |y_i^s - n|) \forall i \forall c \tag{2.5}$$

Spatial Transformers proved to be successful in improving the classification results on several tasks. They have mostly shown their power in classifying the Distorted MNIST dataset (Fig. 2.9) by selecting the part of the image that is most relevant (attention aspect), thus eliminating the irrelevant background. STNs also succeeded to correct the distortion and standardize the input to the classifier, thus making it robust to the geometric perturbations.

2.2 Automatic Target Recognition (ATR)

One of the key components of defense weapon systems to be used on autonomous vehicle missions is the automatic target recognition (ATR) system. Basically, the ATR system performs

automatic target acquisition, identification, and tracking by processing a sequence of images obtained from sensors. Two sensors are commonly used to capture data for ATR: Synthetic Aperture Radar (SAR) and Infrared (IR). IR imagery is the conventional imaging modality for defense application. ATR systems have proven to be a necessity in the context of military applications, by enhancing intelligence and removing the human intervention that may limit the overall system or mission in real situations.

2.2.1 Challenges

Building a robust and reliable ATR system is a difficult task that highly depends on the quality of the input images, considering the challenges that the data might pose. The infrared radiation received by the camera sensor can vary significantly depending on the meteorological conditions, sensor calibration and time of the day. This leads to large intra-class variability , where targets from the same class, captured under certain conditions, may look different. Moreover, ATR systems are expected to be invariant to different viewpoints, since targets are usually captured at various orientations. These challenges can be more severe to handle when the distance between the sensor and the target being captured increases, reducing the resolution of the targets as they get represented by only few pixels. In particular, at low resolutions, different targets might look similar, leading to large inter-class similarities. The state of targets can also affect the performance of ATR systems. For example, stationary and cold vehicles blend with the background or plumes of engine smoke partially mask the target.

2.2.2 Previous work in ATR

Most early work in the ATR task adopted a model-based learning approach [12–14] that relies on manual feature extraction, accompanied with classification. However, recent successful ATR applications shifted toward the use of convolutional neural networks [15–18].

2.2.2.1 Model-based Approaches

Traditional model-based approaches proceed by first extracting features and then performing the classification. For instance, the authors in [12] proposed to automatically identify four different ships in simulated infrared imagery by considering Scale Invariant Feature Transform (SIFT) [30] features that they matched to a database of features corresponding to labelled images using the euclidean distance metric. Similarly, the authors in [13] used Histogram of Oriented Gradients

(HOG) [23] features followed by a Support Vector Machine (SVM) [31] classifier trained for a binary object/non-object classification task. When video sequences are available, other methods can be proposed. For example, the authors in [14] used background subtraction to detect and identify moving objects in thermal images. Even though successive frames can increase the robustness to background variations, such techniques may miss cold or stationary targets that blend in the environment.

2.2.2.2 Deep learning-based Approaches

Most CNN applications for object detection and recognition have been developed for optical images. Using other imaging sensors (radar, sonar, and infrared) may lead to more challenging problems due to the intrinsic image characteristics described earlier in Sec.2.2.1. In this thesis, we focus on mid-wavelength infrared (MWIR) imagery. Some CNNs have been proposed for this kind of images for target classification. For example, in [15], the authors present a CNN based deep learning framework for automatic recognition of civilian targets in infrared images. The authors of [16], on the other hand, used deep representations to address the problem of IR object classification by dividing the object appearance space hierarchically with a binary decision tree structure that consists of the object features. Another study [17] combines a CNN for object classification with an automatic target detection (ATD) algorithm to perform Automatic Target Recognition (ATR) on mid to long-wavelength infrared imagery. This approach enables to separate the training and validation of each stage of the ATR chain, which is regarded as a key feature when dealing with real-world applications. The most recent study in [18] proposed a compact and fully convolutional neural network with global average pooling (GAP), called cfCNN. The authors used high-quality synthetic data for training and evaluated their network on three real datasets with increasing quality. The paper also analyzed the robustness of the proposed CNN against possible perturbations introduced by the detection stage, such as shifted and scaled bounding boxes. The authors claim that the use of GAP is responsible of the geometric invariance of their suggested network.

Despite the advances in deep learning, specifically the recent great success of Convolutional Neural Networks, the number of neural network based ATR systems is still minimal, and this is due to the lack of very large ground-truthed datasets in the defense context, a paramount factor when trying to deal with deep neural networks.

In this thesis, we investigate the application of CNNs to automatic target recognition using infrared images. To this end, we use a backbone CNN architecture of WideResNet 28-2 [1]. We

explore three distinct learning approaches. The first method is based on randomly initializing the weights of the CNN. The second method aims to improve the network initialization by using transfer learning. Finally, the third method is a novel approach that introduces a block of a spatial transformer [2] into the CNN. We conduct a thorough evaluation, comparison and analysis of the performances of the three suggested CNNs.

CHAPTER 3

AUTOMATIC TARGET RECOGNITION WITH CONVOLUTIONAL NEURAL NETWORKS

In this chapter, we present our adaptation of three CNN variations to automatic target recognition using infrared imagery. We start by introducing our three models: randomly initialized CNN, finetuned CNN, and CNN combined with a block of spatial transformers [2]. Then, we describe the data we used to evaluate our models, along with its preprocessing. Finally, we present our experiments that were designed to investigate four important research questions.

3.1 Proposed models

For all our models, we use a baseline CNN of WideResNet 28-2 (WRN-28-2) [1]. It is an architecture that inherits the concept of residual blocks from Residual Neural Networks [28], with 28 convolution layers in depth and twice the number of channels of a regular ResNet. Fig. 3.1 describes the architecture of WRN-28-2 used in our experiments.

The CNN uses a global average pooling (GAP) layer instead of fully connected (FC) layers, for its capability to enhance the viewpoint-invariance, mainly toward translation perturbations, by summing up the spatial information of the feature maps. WRN-28-2 also uses Leaky-ReLU activation function for its ability to solve the "dying ReLU" problem since it doesn't have zero-slope parts, and for its ability to speed up the training [32].

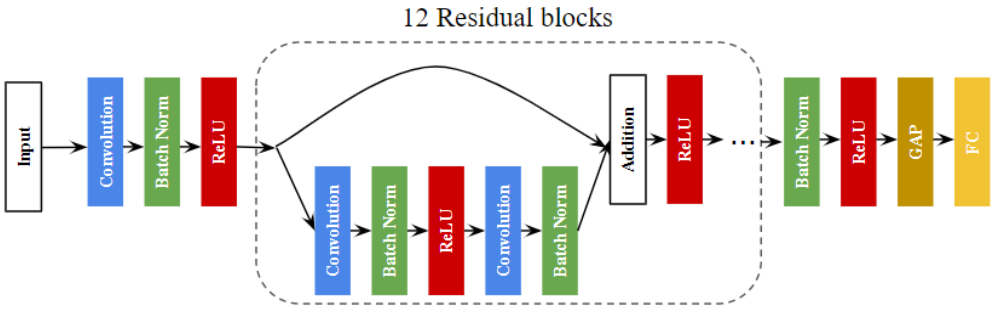

Figure 3.1: Architecture of WideResNet 28-2.

3.1.1 Randomly initialized CNN

Our first model is based on the state of the art WRN-28-2 , with random initial weights. The convolution layers are initialized using Kaiming initialization [33], the batch normalization layers are initialized with constant values of 1 for the weight layer and 0 for the bias, and finally the linear layer is initialized using Xavier initialization [33].

3.1.2 Finetuned CNN

Our second model uses transfer learning for a better initialization of the weights of the CNN. We start with training a WRN-28-2 model on a vehicle dataset, called VMMRdb [34]. It is a diverse large-scale dataset of vehicles, containing 9,170 classes, consisting of 291,752 images, and covering models manufactured between 1950 and 2016. Each car image is associated with a label that specifies the make, the model and the manufacture year of the vehicle. The images are of high resolution, taken by different users, different imaging devices, and multiple view angles. The vehicles are also not well aligned, and some images contain irrelevant background, thus ensuring a wide range of variations to account for various scenarios that could be encountered in a real-life scenario.

In the context of the proposed ATR application, we use the VMMRdb dataset to build a model that recognizes the vehicle type, given a vehicle image, by classifying it as a Sedan, SUV or Pickup. We build the appropriate database for this model by regrouping the fine-grained car classes of VMMRdb into only 3 categories: Sedan, SUV and Pickup. We proceed by merging the different manufacture years of the same model of a vehicle, and mapping its original label to one of the 3 proposed labels. We only consider the vehicles that have more than 50 images per class of makes and models combined. Finally, we sample the three classes to balance the resulting dataset. This results in a data collection that contains 35,000 sample images for each class, and a total of 105,000 images for the three classes. Fig. 3.2 displays sample training images from each class. We convert the images to grayscale to simulate the infrared aspect of the ATR imagery.

3.1.3 CNN with spatial transformers

Our third model integrates a spatial transformer network [2] in the WRN-28-2 CNN architecture [1]. As illustrated in Fig. 3.3, we insert one module of STN right before the first layer of the CNN. Therefore, the STN block takes as input the preprocessed target patch, learns and applies an appropriate transformation that enhances the global accuracy and generates a transformed input that will be passed to the CNN for feature extraction and classification. The entire network, that

Figure 3.2: Sample preprocessed training images from the constructed car dataset with 3 classes.

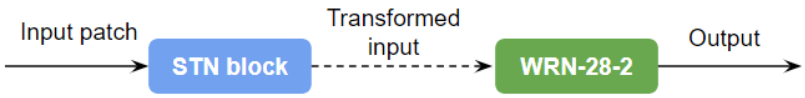

Figure 3.3: CNN architecture of the third model: WRN-28-2 with a block of STN.

consists of one block of STN and a WRN-28-2, will be trained in an end-to-end fashion.

For the localization network of the STN block, we use a VGG-16 [35] CNN architecture (Fig. 3.4) that takes as input the target patch and aims to regress the transformation parameters. We choose to apply an affine transformation on the target inputs, which means that we only consider rotation, translation or scaling transformations. Therefore, the localization network attempts to learn 6 parameters that characterize the affine transformation. Finally, we use bilinear interpolation as a sampling kernel, to compute the pixel values of the transformed input, given the input values and the pixel locations provided by the grid generator. The bilinear interpolation is specifically chosen because it takes into consideration weighted pixel values of the source image in order to compute pixel intensities of the transformed image. This interpolation step will smooth the image patch by eliminating pixel intensity jumps.

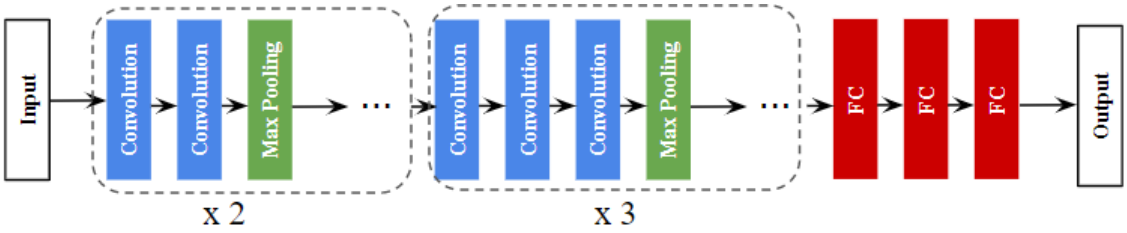

Figure 3.4: CNN architecture of VGG-16.

3.2 Dataset

To validate and compare the three CNN architectures, we use the data set provided by the Defense Systems Information Analysis center (DSIAC) [1]. The DSIAC data contain a large collection of Middle Wavelength Infrared (MWIR) imagery collected by the US Army Night Vision and Electronic Sensors Directorate (NVESD), intended to support the ATR algorithm development community.

The DSIAC data are captured and stored as videos in ARF[2] file extension. Each video file has a ground truth data stored in JSON [3] format.

Each video is stored as a set of imagery and is labeled using a unique scenario identifier that contains meta data. The scenario identifier is composed of the sensor identifier (*cegr*), a five digit scenario number, and a four digit look number. An example of a scenario identifier is "cegr01923_0001". The scenario numbers correspond to time of day and range of the target. For instance "01923" is a scenario number that describes targets captured during the night at the slant range 1000. The look numbers correspond to the particular type of target. For instance, the look number "0001" refers to the target PICKUP.

Each frame of a given video has 512 rows and 640 columns. Fig. 3.5 displays an example frame from the DSIAC database. Each ARF file is associated with a ground truth JSON file that describes various information about the target, such as, the target type, the slant range, the aspect orientation of the vehicle, the time at which the frame was collected, and also the coordinates of the bounding box that locate the target within the frame.

The targets described in the database include people, foreign military vehicles, and civilian vehicles. In our experiments, we aim to classify 10 vehicle classes:

- 2 civilian vehicles (Fig. 3.6): *PICKUP, SUV (Sport Utility Vehicle).*

[1] https://www.dsiac.org/
[2] Advanced Recording Format [36]
[3] JavaScript Object Notation [37]

Figure 3.5: Example of a target frame from the DSIAC database.

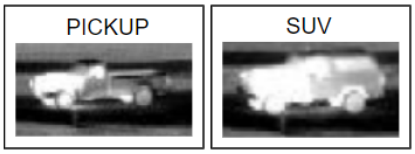

Figure 3.6: Example images of civilian targets in the DSIAC database.

- 8 military vehicle targets (Fig. 3.7): *BTR70 (Armored Personnel Carrier), BRDM2 (Infantry Scout Vehicle), BMP2 (Armored Personnel Carrier), T72 (Main Battle Tank), ZSU23 (Anti-Aircraft Weapon), 2S3 (Self-Propelled Howitzer), MTLB (Armored Reconnaissance Vehicle), D20 (Artillery Piece towed to MTLB).*

These targets were captured at different aspect angles and different slant ranges, from 500 to 5000 meters with steps of 500, during both day and night. This was accomplished by marking a

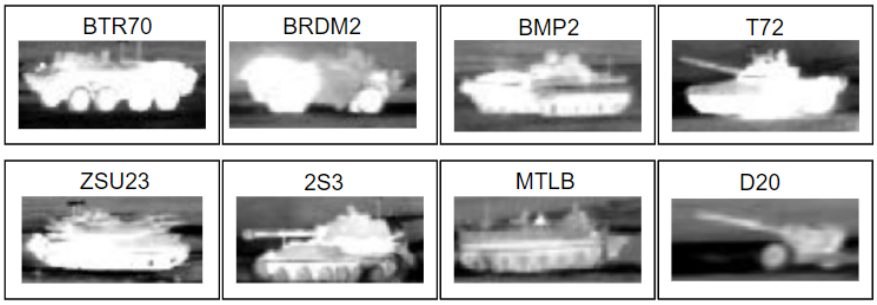

Figure 3.7: Example images of military targets in the DSIAC database.

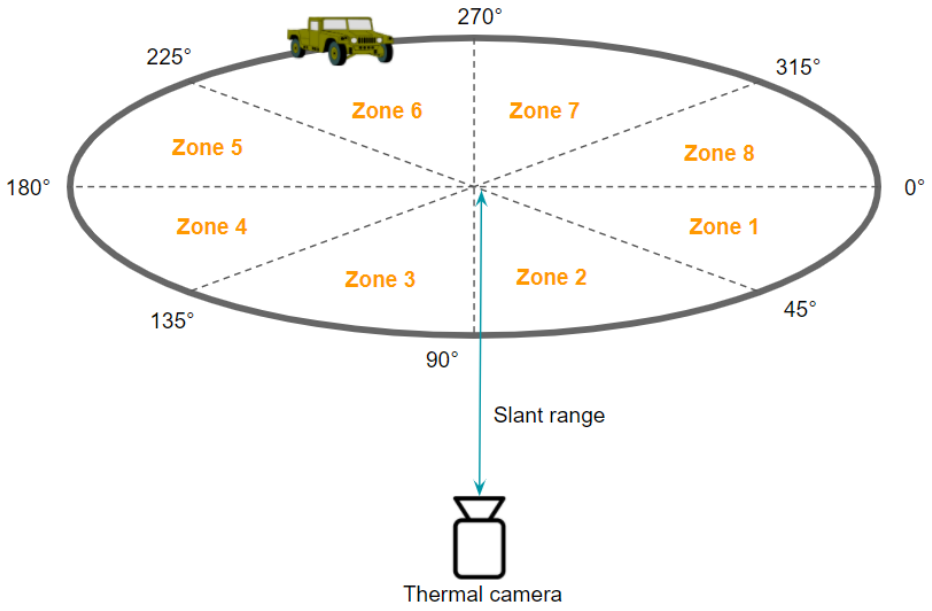

Figure 3.8: Illustration of the quantization of the trajectory of a moving vehicle into 8 zones.

circle with a diameter of about 100 meters at each range that allowed the targets to be consistently imaged at the same locations and aspect angles. Drivers were instructed to drive in a continuous circle at a constant low velocity (about 10 mph). Therefore, successive frames of the same video can have similar content.

In our experiments, we focus on targets that are captured within a minimum range of 1000 and a maximum range of 3500. In fact, targets at higher ranges are captured by only few pixels, and therefore they do not include sufficient information for training or testing.

We quantize the trajectory made by the moving target in a video into 8 zones, as illustrated in Fig. 3.8, where each zone has a range of 45°. The zone quantization of the vehicle trajectory enables to delimit the different viewpoints of a single target, as shown in Fig. 3.9. This will allow us to evaluate the performance of the trained models on the different viewpoints of the targets.

For each range, there are 2 videos per target vehicle: one during the day and one during the night. Considering that each video has 1800 frames, in total there are 21600 images for each target across all ranges of interest.

3.2.1 Data preparation

Each target vehicle can be delimited within the image frame using the coordinates of the corresponding ground-truth bounding box, which takes into account the position of the target and the viewpoint. As illustrated in Fig. 3.10, for a given bounding box $dx \times dy$, we consider a squared

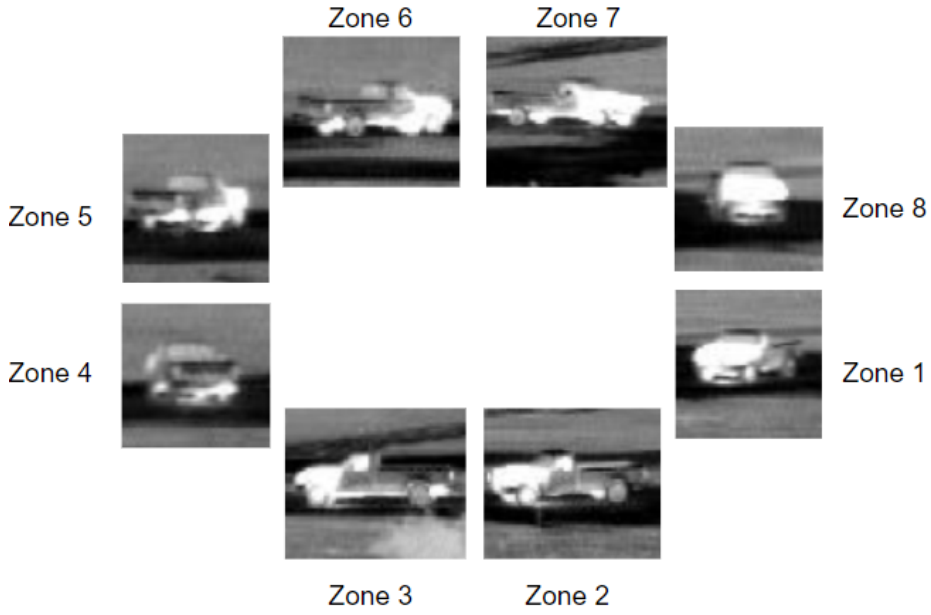

Figure 3.9: Illustration of the different viewpoints of a civilian target PICKUP in all 8 zones.

patch of size $1.1 \times max(dx, dy)$ centered in the center of bounding box. Using a bilinear interpolation, we rescale this patch to a 32×32 patch. We adopt this process because it does not affect the shape and aspect ratio of the target within the image. It also enables us to include some of the background in the cropped image patch.

The amount of background included in the extracted image patch varies according to the target orientation. For example, as shown in Fig. 3.11, minimal background is included as the original bounding box and the extracted square are identical since the target is already detected within a square box. This mainly happens to targets positioned at certain zones (zones 1, 4, 5 and 8) that capture either their back or front views. It also happens to targets captured at high ranges, where they become so small that their true dimensions can no longer be clearly discerned, and thus they can be delimited by small square bounding boxes.

3.2.2 Training design

The classical training design that typically consists of shuffling the data and splitting it into training, validation and testing sets is not the best choice for the DSIAC data. In fact, the targets of this data set are not solely characterized by their class types. There are also other important information to account for, such as the range of the target, its viewpoint and the time of the day at which it was captured. Moreover, data of the same video cannot be used for both training and testing since all target occurences share the same background, which rises an overfitting problem.

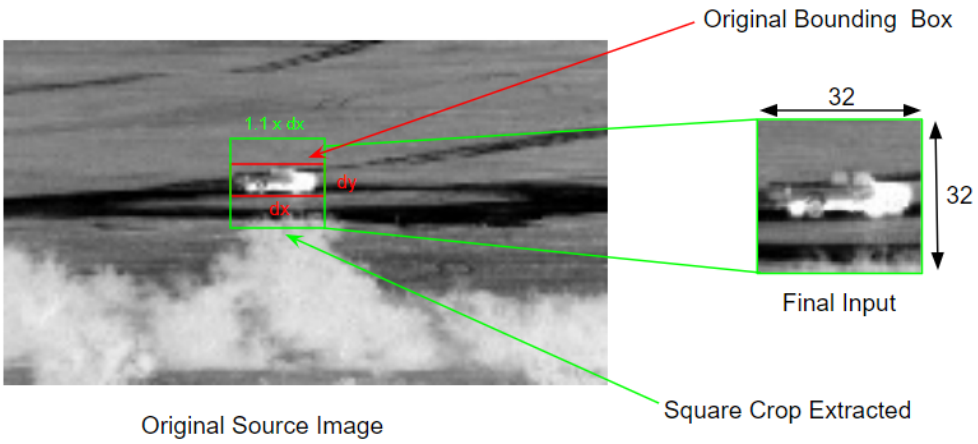

Figure 3.10: An extracted patch image (in green) from the original infrared image using the original bounding box information (in red). The target is a PICKUP captured at zone 6 and at a range of 1000 meters.

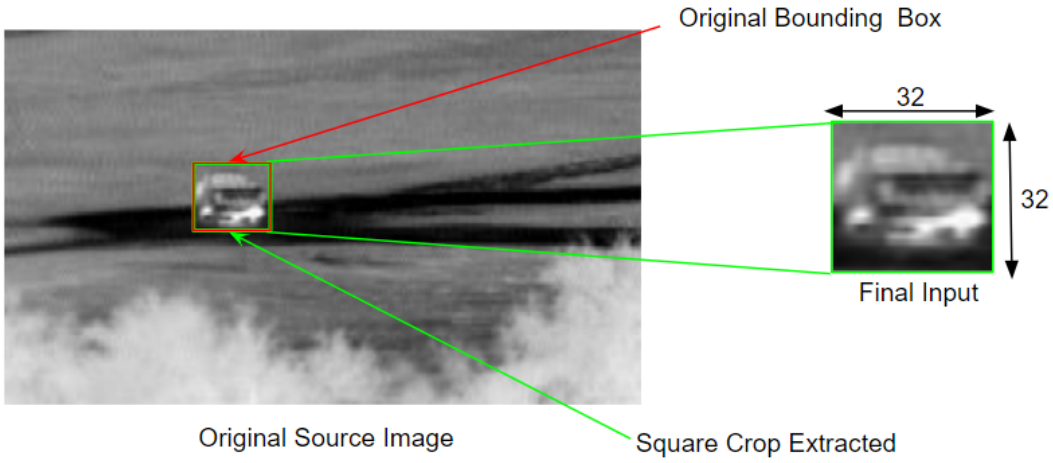

Figure 3.11: An example of an extracted patch image where the original bounding box (in red) is a square. In this case, no additional background is needed to pad the extracted patch (in green). The target is a PICKUP captured at zone 4 and at a range of 1000 meters.

To minimize the risk of overfitting, we train and test on different ranges by adopting a 6-fold cross validation training design. We split our folds based on the range, meaning that, at each time we train on only 5 of the 6 ranges and we test on the excluded range. Using this data partitioning, we ensure that both the training and testing subsets include all target types collected during the day and the night and at all viewpoints. We also ensure that we test on targets captured at ranges not used for training.

For each fold, the test set is divided into two subsets. The first one is used for validation during the training to select the best model. The second one is used for testing and evaluation after the model is fixed. The validation and testing subsets are obtained by splitting the excluded range

TABLE 3.1

Experimental setting: data split into training, validation and testing

Folds	Train		Validation		Test	
	Range	# samples	Range	# samples	Range	# samples
Fold 1	[1500,2000,2500,3000,3500]	176400	1000	17980	1000	18020
Fold 2	[1000,2000,2500,3000,3500]	176400	1500	18408	1500	17592
Fold 3	[1000,1500,2500,3000,3500]	176400	2000	16787	2000	19213
Fold 4	[1000,1500,2000,3000,3500]	176400	2500	17125	2500	18875
Fold 5	[1000,1500,2000,2500,3500]	178200	3000	17369	3000	16831
Fold 6	[1000,1500,2000,2500,3000]	178200	3500	17501	3500	16699

based on the zones. The validation subset includes zones 1, 3, 5 and 7 of each target video, while the testing subset includes the remaining zones 2, 4, 6 and 8. We distributed the zones between the validation and testing subsets this way because the frames that describe targets in consecutive zones encompass similar contents. Therefore, we can still assume that the validation and testing subsets include all target viewpoints. Both the validation and the testing subsets include all target types and both day and night images for each target. Table. 3.1 summarizes our experimental setting.

3.3 Evaluation strategy

The objective of this thesis is to develop a robust ATR system for infrared imagery based on CNN. We evaluate, analyze and compare three variations of WideResNet-28-2 [1]. We guide our research by formulating and answering four important research questions (RQ). These RQ's are described and explained in the following sections.

3.3.1 RQ1: Does preprocessing of the input images improve the robustness of the ATR?

Motivation: The pixel values of infrared images have a wide dynamic range and their distribution can have very long tails. Therefore, using raw data in the training can make the learned models highly sensitive to the sensor calibration.

Approach: To normalize the pixel values to a [0, 1] range and maintain the image contrast, we propose to preprocess each input frame by transforming its pixel values using the following two steps:

- Clipping: To eliminate the long tails of the pixel intensity distribution, we identify a lower threshold that corresponds to the 0.1 percentile of the distribution and an upper threshold that

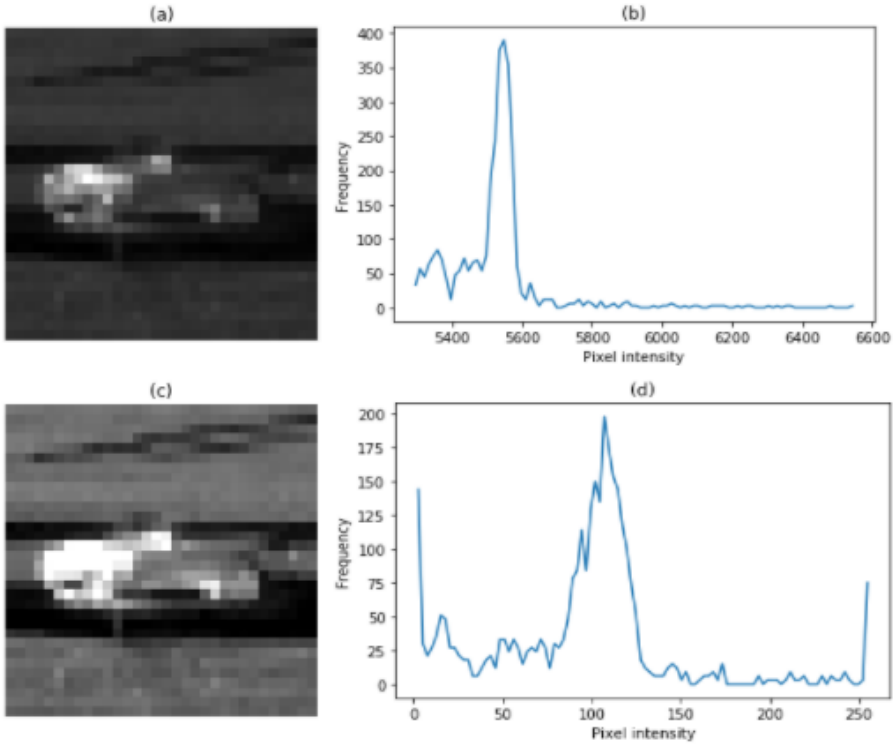

Figure 3.12: (a): Original target patch; (b): Histogram of pixel intensity of (a); (c): Target patch after preprocessing; (d): Histogram of pixel intensity of (c)

corresponds to the 99.9 percentile. Pixel values below or above these thresholds are mapped to these values.

- Normalization: The median and the Median of Absolute Deviation (MAD) are statistical measures that are more robust to outliers than the mean and standard deviation. Thus, we normalize each frame X_{frame} using these measures:

$$X_{normalized\,frame} = \frac{X_{frame} - median(X_{frame})}{MAD(X_{frame})} \tag{3.1}$$

The normalized values are clipped again to remain within the range of [-5, 10].

After normalizing the image frame, we extract the target using the corresponding coordinates of the bounding box. Then, we standardize the pixel values of the resulting target patch, X_{patch}, by normalizing it using its mean and standard deviation, as follows:

$$X_{normalized\,patch} = \frac{X_{patch} - \mu(X_{patch})}{\sigma(X_{patch})} \tag{3.2}$$

Fig. 3.12 illustrates the preprocessing effect on a typical image frame.

To answer RQ1, we train and test our models using the raw data and the preprocessed data. We analyze and compare the accuracy of the two approaches.

3.3.2 RQ2: How well can the learned models generalize to targets captured at different resolutions?

Motivation: The ATR system should be able to identify targets captured at any reasonable resolution. Since it is not possible to consider all the possible ranges for training, an ATR model should be able to generalize to the unseen ranges.

Approach: In order to evaluate this generalization ability, we train our models using a 6-fold cross validation based on the range, as described in Sec.3.2.2. Using this approach, for each fold, we reserve one range for testing and validation and we train using all other ranges.

3.3.3 RQ3: Are the learned models invariant to small scale and shift variations?

Motivation: In real-world situations, ground-truth data is unavailable. Realistic scenarios consist in having the targets automatically detected using state of the art object detectors, e.g. You Only Look Once algorithm (YOLO) [7]. Despite their variety and wide success, the performance of even the most efficient object detector can be hindered by the many challenges of the ATR data (Sec 2.2.1). These object detectors could fail to provide precise bounding boxes that enable to delimit well the targets within the image frames. For instance, Fig. 3.13 illustrates some detection errors committed by YOLO. It can be noted that some of the detected boxes are not centered around the targets, while other boxes tightly delimit the targets to the point of causing the occlusion of some target parts.

Approach: To assess the robustness of the models to localization errors and their sensitivity to scaling, we conduct two experiments. The first experiment consists in evaluating the models on a test set we created, which includes shifted and scaled versions of the ground truth bounding boxes. The second experiment consists in evaluating the trained models on targets automatically detected by YOLO [7].

3.3.4 RQ4: Can the models reject the non-targets identified by the detector as potential targets?

Motivation: When using a detector, e.g YOLO [7], the ultimate goal is to minimize the number of missed targets. As a matter of fact, if a target is missed at the detection stage, the system will not be able to recover it at any subsequent steps. However, acquiring the maximum number of detections possible leads to the inclusion of false alarms in the process. Therefore, a robust ATR system should be able to detect anomalies and reject the maximum number of detections that

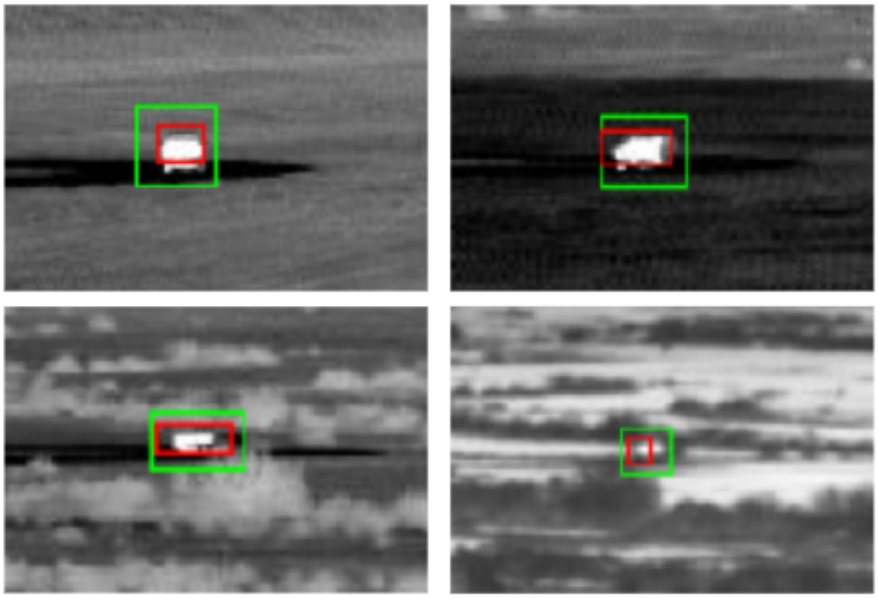

Figure 3.13: Example image frames where the YOLO does not delimit well the target. The ground truth box is shown as the green box, while the YOLO detected box is displayed as the red box. The sample image frames are extracted from different ranges and different target types.

correspond to non-targets.

Approach: We investigate an approach that expands the number of classes by introducing an additional class for the "non-target" detections. This class includes a variety of anomalies that do not describe true targets. The objective is to train models to become able to discriminate the targets of interest from any other surrounding, such as background, animals, vegetation ... etc.

3.3.5 Analysis measures

In our analysis, we evaluate the performance of the trained models with respect to various parameters that characterize the image patches. Some of these parameters are documented in the data collection. These include target range, time of collection (day or night), viewing angle (described in Sec.3.2), ...etc. We also define the following two measures:

- **Contrast**: The contrast can be defined as the difference in intensity between the object and its background. We calculate the contrast of the ground truth bounding box with respect to a larger box that contains more background, using:

$$contrast = \frac{|\mu_1 - \mu_2|}{\sigma_1 + \sigma_2} \tag{3.3}$$

where μ_1 and σ_1 are, respectively, the mean and standard deviation of the bigger box, while μ_2 and σ_2 are respectively the mean and standard deviation of the ground truth box. It is

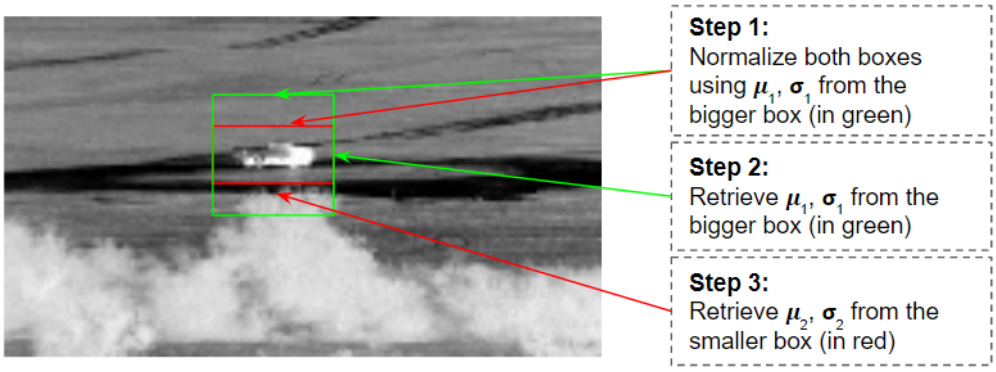

Figure 3.14: Steps to compute the contrast measure.

important to mention that we normalize the two boxes similarly using the mean and standard deviation of the bigger box. We illustrate the different steps to compute the contrast measure in Fig. 3.14.

The contrast of a target image, depending on its value, may affect the classification power of the network. In fact, when the contrast is high, which occurs when the target is either hotter or colder than its background, the target can be easily distinguished which should facilitate the classification task. However, when the contrast is low, which happens when the target blends in with the background, the classification task is expected to become more difficult.

- **Aspect ratio**: In order to have a better understanding of the dimension of a given target, we calculate the aspect ratio of the ground truth bounding box, as follows:

$$Aspect\ ratio = \frac{Height}{Width} \tag{3.4}$$

This measure is helpful in assessing the classification power of the trained models with respect to the dimension of the ground truth box. As a matter of fact, since we consider a square patch to delimit the target within the frame (Sec.3.2.2), if the $Height << Width$ or $Width << Height$, the square patch would consequently include irrelevant background which may affect the classification power.

CHAPTER 4

EXPERIMENTAL RESULTS

In this chapter, we evaluate, analyze and compare the models presented in Chapter III. We start by presenting our experimental setting. Then, we outline our hyperparameter tuning process. Finally, we address each RQ by describing the conducted experiments and analyzing the results.

4.1 Experimental Setting

In our experiments, we use random sampling [38] to load and process the training data by randomly permuting the list of indices of the training set for each epoch. We choose to proceed this way in order to minimize the risk of overfitting the training data and memorizing the data by the order it was presented to the network. This is likely to happen in our case, mainly because our data is originally extracted from videos, thus consecutive frames encompass highly similar content.

We define a number of iterations instead of a number of epochs. The number of iterations corresponds to the number of sampling batches of training data points. In other words, we can express the number of epochs using the number of iterations as:

$$nb_{epochs} = \frac{nb_{iter} \times k}{l_{train}} \tag{4.1}$$

where nb_{epochs} is the number of epochs, nb_{iter} is the number of iterations, k is the sampling batch size, and l_{train} is the size of the training set.

Finally, we create checkpoints during the training to evaluate the trained model on the validation set. The checkpoint with the best performance is saved and used for testing.

4.1.1 Experimental platform

We ran our experiments on a computer equipped with an Intel Core i7-5930K CPU (12 CPUs), an NVIDIA GeForce GTX TITAN X GPU with 12 Gb of VRAM and 128 GB of RAM.

TABLE 4.1

Experimental settings and their investigated hyperparameters values

	Number of iterations	**Learning rate**	**Learning rate decay**
Setting 1	50k	4e-4	No
Setting 2	100k	4e-5	No
Setting 3	100k	4e-4	decay factor = 0.1 after 50k iterations
Setting 4	100k	4e-4	No

4.2 Hyperparameter tuning

The hyperparameter tuning process is an empirical approach that aims to find the hyperparameters that optimize the performance of the model. Since we are using a state of the art CNN, we fix the hyperparameters that are related to the architecture, such as the kernel size of the filters and the number of feature maps in the convolutional layers , to their default values as suggested in the original Wide ResNets [1] paper. We also fix the batch size and the optimizer. Therefore, our hyperparameter tuning experiments focus only on the *learning rate*, the *number of iterations*, and whether we should apply a *learning rate decay*.

Table. 4.1 summarizes the 4 experimental settings and their hyperparameter values. For this experiment, we only use the randomly initialized CNN. We assume that, since the other models (finetuned CNN and CNN with STN) share the same backbone CNN, they should agree on the hyperparameter values.

Since we are adopting a 6-fold cross validation training design as discussed in Sec.3.2.2, we choose to train the considered CNN for only three folds, which correspond to three representative testing ranges: range 1000 that represents the low ranges, range 2500 that represents the medium ranges, and range 3500 that represents the high ranges. For each experiment, we train the models 3 times using different random initializations. In Fig. 4.1, we report the mean and standard deviation of the results. We note that settings 3 and 4 equally achieve the best mean classification accuracy for the three considered testing ranges. However, setting 3 yields a less consistent performance (larger standard deviation) over the different runs when compared to setting 4.

Fig. 4.2 displays the evolution of the validation accuracy during training for a model trained with setting 3 versus a model trained with setting 4. As it can be seen, training with setting 3 appears to be more stable, mainly during the second half of the training when we apply the learning rate decay. Based on the above observations, we select setting 4 as the best configuration

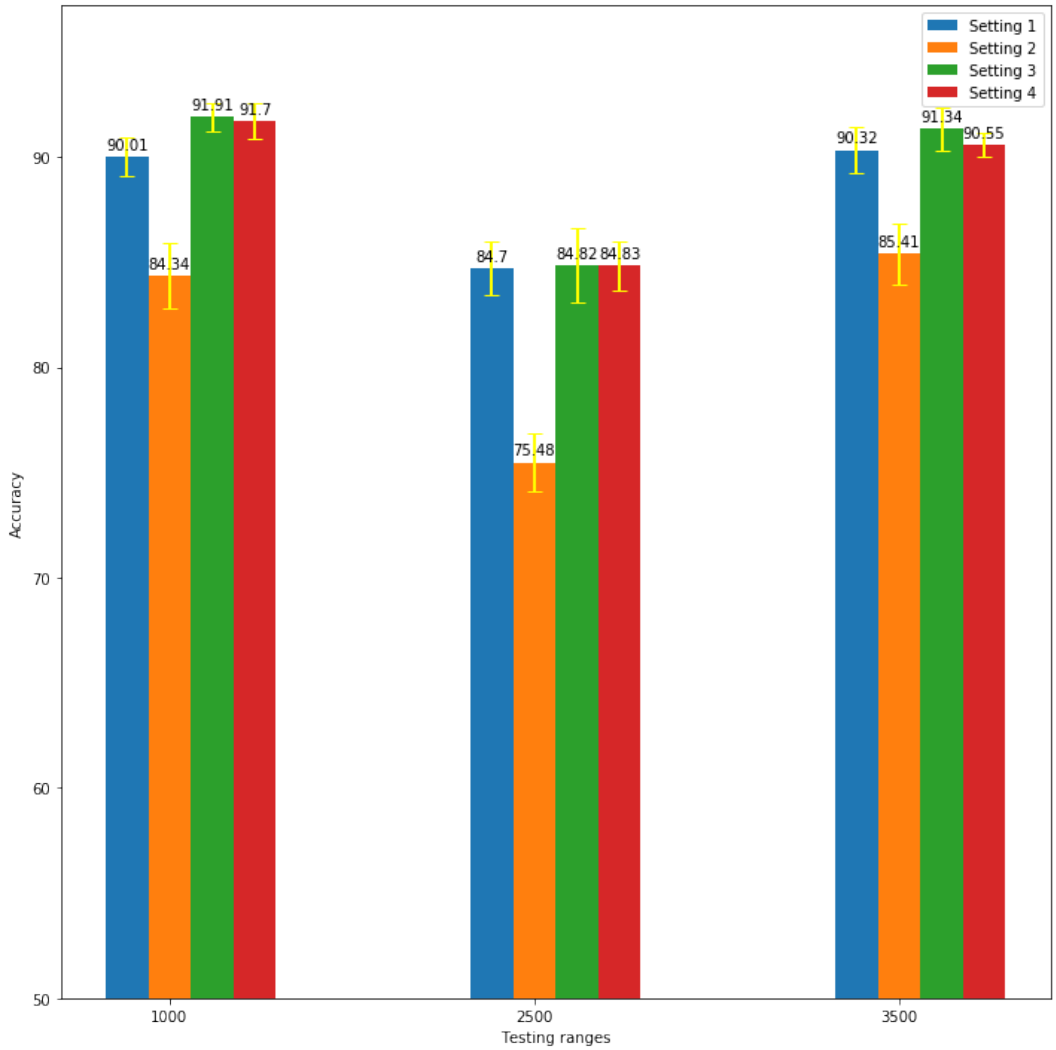

Figure 4.1: Mean and standard deviation of the classification accuracy for 3 runs with different random initializations.

of hyperparameters to train our models with.

Table. 4.2 summarizes the final hyperparameter setting considered for the rest of the experiments.

4.3 Experiments designed to investigate the research questions (RQ1-RQ4)

Our objective is to build a CNN-based ATR system that can efficiently and reliably classify targets captured by an infrared sensor. Most importantly, the proposed model must exhibit robustness against several perturbations imposed by real-life scenarios. To ensure that our objectives are satisfied, we guide our investigation by defining the four important research questions, introduced in Sec.3.3. In this section, we investigate each research question by conducting the appropriate

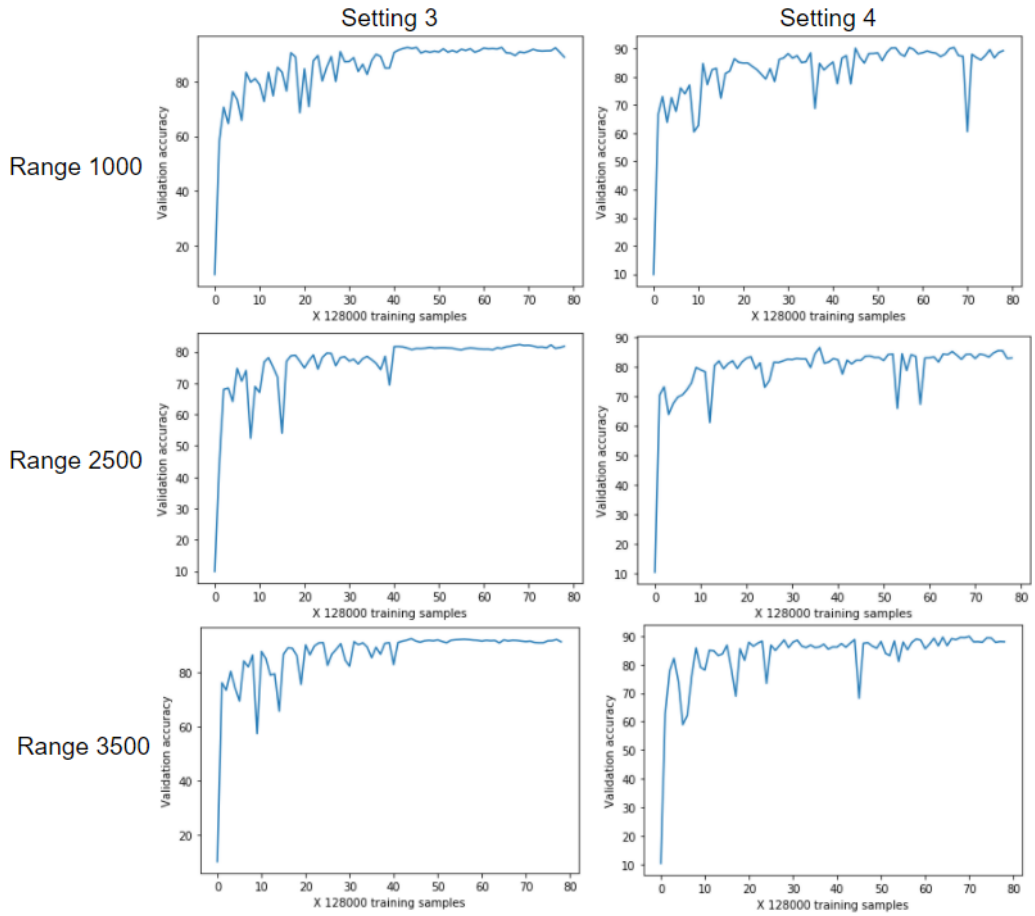

Figure 4.2: Evolution of the validation accuracy during training of one model trained with setting 3 and one trained with setting 4.

TABLE 4.2

Final hyperparameter setting

Hyperparameter	**Value**
Number of iterations	100k
Sampling batch size	100
Learning rate	4e-4
Batch size	128
Optimizer	Adam

experiments and analyzing the results.

4.3.1 RQ1: Does preprocessing of the input images improve the robustness of the ATR?

To investigate this research question, we train a randomly initialized WRN-28-2 model and test it on two variations of the data. The first one is the raw data, and the second one is the data preprocessed using the steps outline in Sec.3.3.1. We adopt the 6-fold cross validation described in Sec.3.2.2.

The classification accuracy results are reported in Fig. 4.3. As it can be seen, the model trained on raw data where no preprocessing was applied, performed better than the model trained with preprocessed data. The difference in performance is more significant for the high ranges (2500, 3000 and 3500), where it exceeds 10%.

This behaviour may be explained by the fact that all data used in our experiments have been captured by the same camera. Thus, data normalization is not critical. Moreover, the architecture of WRN-28-2 (Sec.3.1) contains multiple batch normalization layers, which are accomplishing the job of normalizing the pixel values of the input images. Consequently, we conclude that no preprocessing is necessary, and we opt for using unpreprocessed data as it yields higher accuracies.

4.3.2 RQ2: How well can the learned models generalize to targets captured at different resolutions?

To investigate this research question, we train our three models, which are the randomly initialized CNN (RD_CNN), finetuned CNN (FT_CNN) and the CNN combined with a block of STN (STN_CNN), using the 6-fold cross validation described in Sec.3.2.2. We report the classification accuracy results of the three models on the six testing ranges in Fig. 4.4.

As it can be seen, the FT_CNN outperforms the RD_CNN on all ranges, except for the high ranges of 3000 and 3500. This observation can be justified by the fact that FT_CNN is initialized with a pretrained model on the VMMRdb dataset [34]. This dataset contains, as described in Sec.3.1.2, high resolution car images that could resemble the targets of the DSIAC database captured at low slant ranges. Nevertheless, the FT_CNN still have a highly comparable performance to the RD_CNN even at higher ranges. Therefore, we can fairly conclude that the weight initialization of the CNN has a crucial impact on the classification performance.

The STN_CNN achieves similar classification performance as the RD_CNN at low ranges.

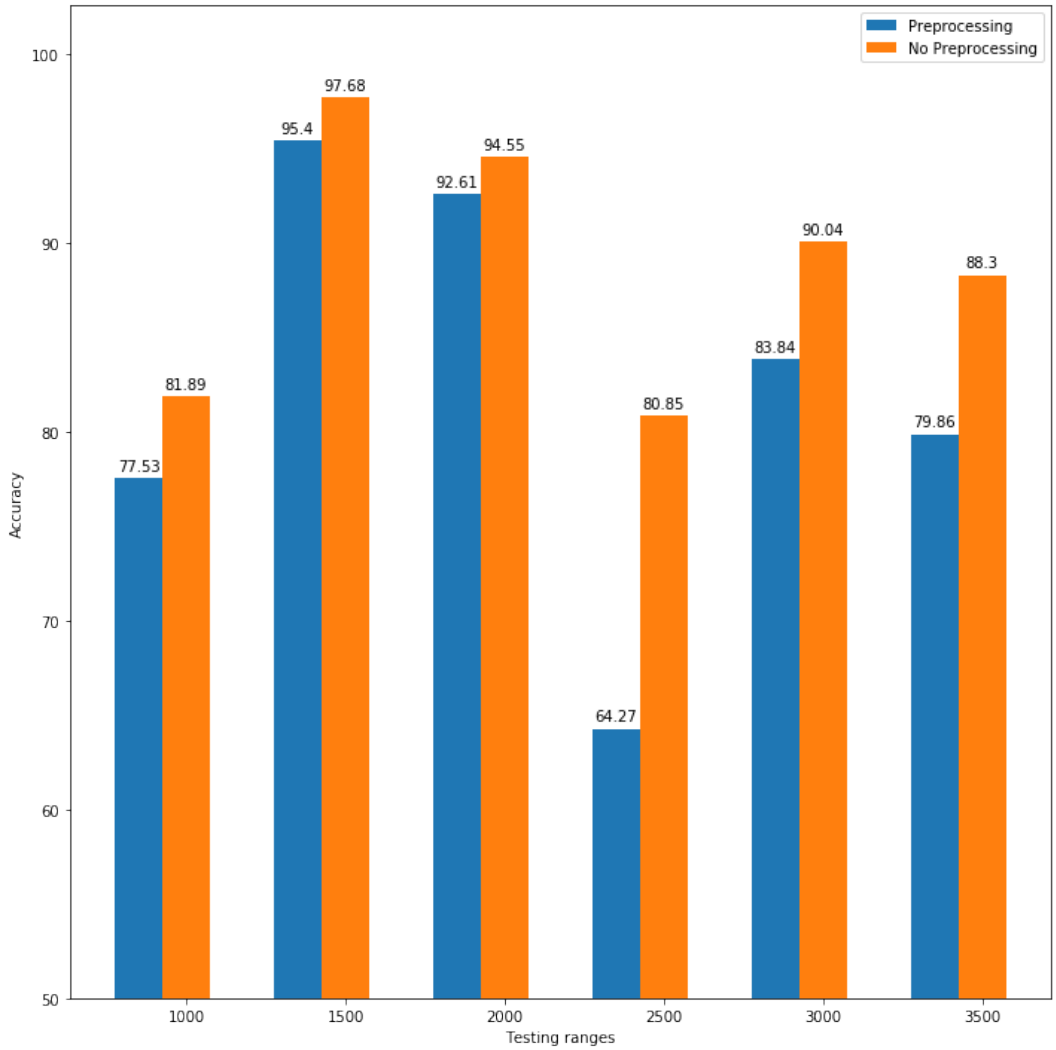

Figure 4.3: Accuracy of the randomly initialized WRN-28-2 model when raw and preprocessed data are used to train and test the model.

However, at higher ranges, it outperforms the other models significantly.

We examine more in depth the performance of the STN_CNN by comparing it to the RD_CNN, since the difference in performance between these two models is the most significant. We limit our analysis to 3 representative ranges: 1000, 2500 and 3500.

Fig. 4.5 displays the distribution of the aspect ratio values of the testing samples that are misclassified by the RD_CNN and correctly classified by the STN_CNN. We notice that the STN_CNN manages to correctly classify samples with low values of aspect ratio ≤ 0.5, unlike the RD_CNN which tends to misclassify them. A low aspect ratio means that the height and the width of the original bounding box of the target are very different, which leads to the inclusion of background when attempting to extract a square crop (refer to Sec.3.1.1).

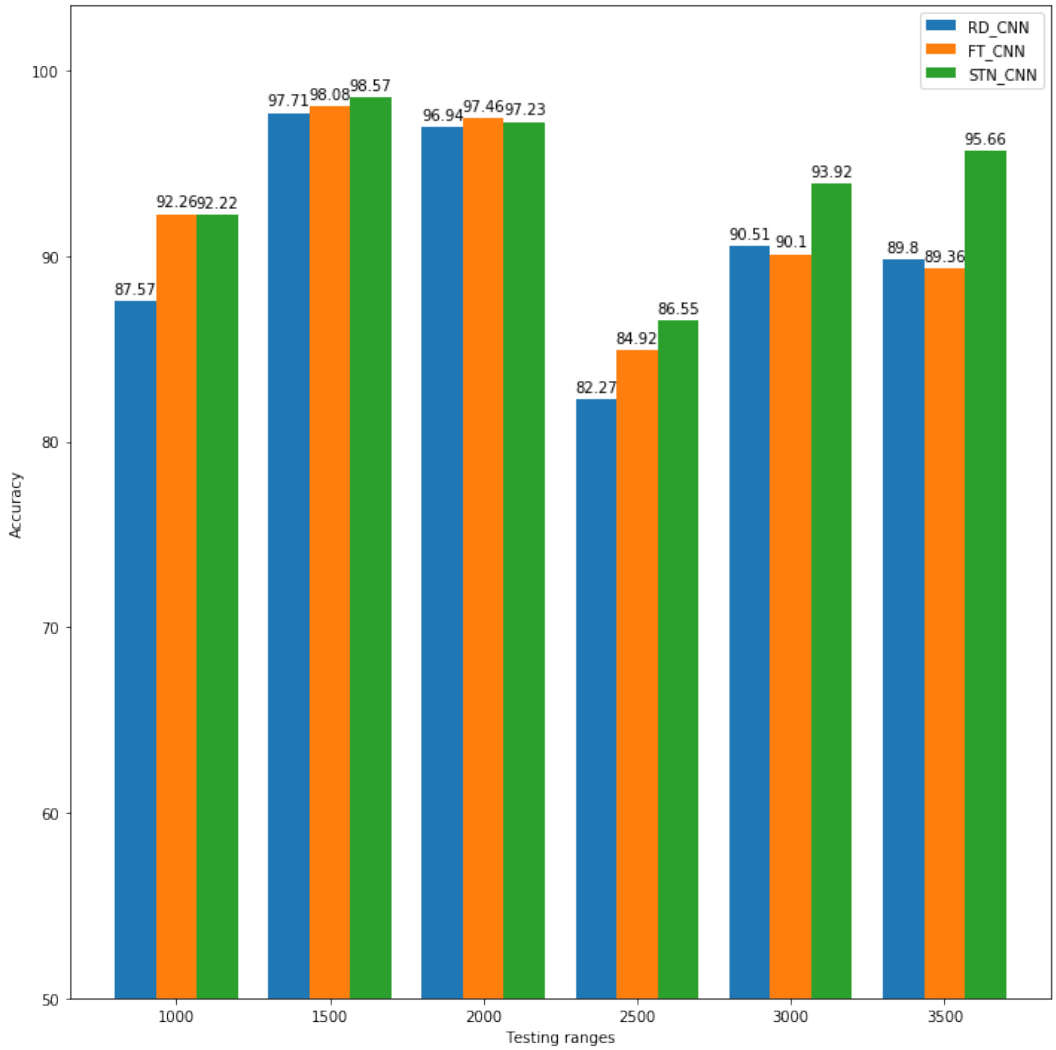

Figure 4.4: Comparison of the classification accuracy of the 3 trained models on the six testing ranges. RD_CNN: randomly initialized CNN ; FT_CNN: finetuned CNN (transfer learning) ; STN_CNN: CNN combined with a block of STN.

Targets captured from their side views are usually delimited with bounding boxes that have low aspect ratio values. This observation can be further supported by our visualization in Fig. 4.6, which confirms that the samples that are misclassified by RD_CNN and correctly classified by STN_CNN, are mainly captured at zones 2 and 6, where the targets appear from their side views.

The performance of RD_CNN drops when it encounters a target situated within irrelevant background. As for the STN_CNN, it spatially transforms the input patch by zooming in on the relevant parts, which results in eliminating the irrelevant background and preserving the target of interest. Therefore, the STN block feeds a transformed input to the CNN, that consists of only the target relieved of its background. This is the "attention" aspect of STNs, that basically represents

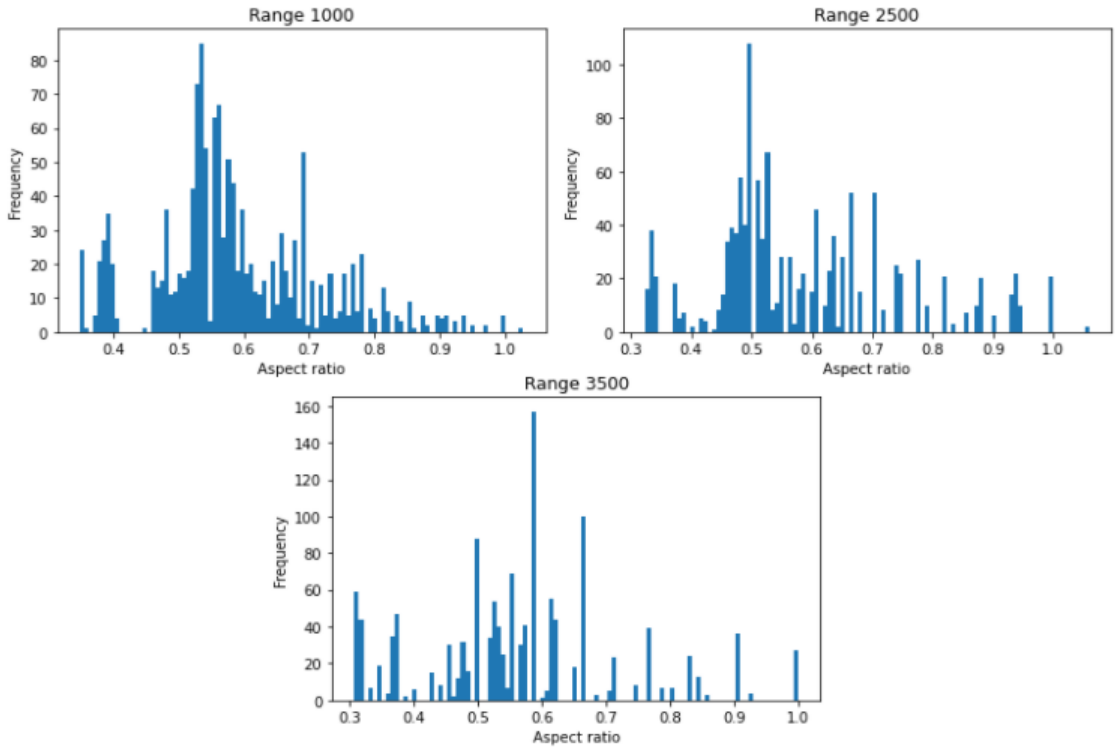

Figure 4.5: Histograms of aspect ratio values corresponding to samples correctly classified by STN_CNN and misclassified by RD_CNN for 3 testing ranges.

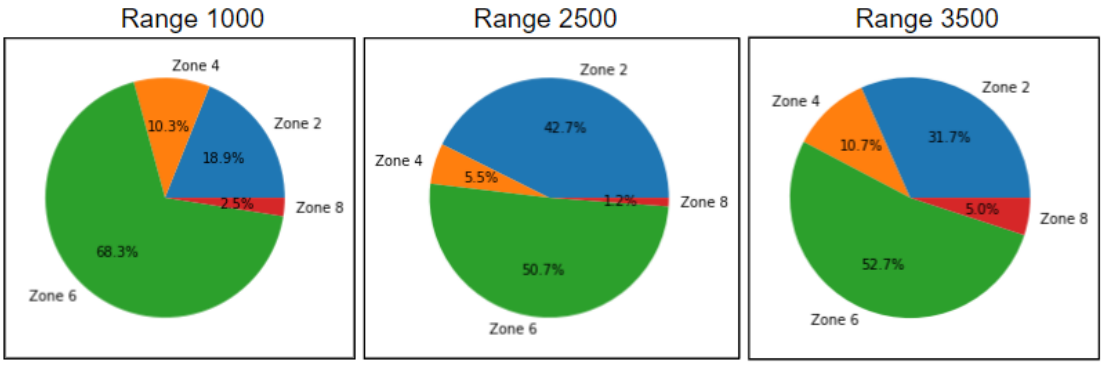

Figure 4.6: Distribution, by zones, of samples correctly classified by STN_CNN and misclassified by RD_CNN. We conduct our analysis on three testing ranges: a low range 1000, a medium range 2500, and a high range 3500.

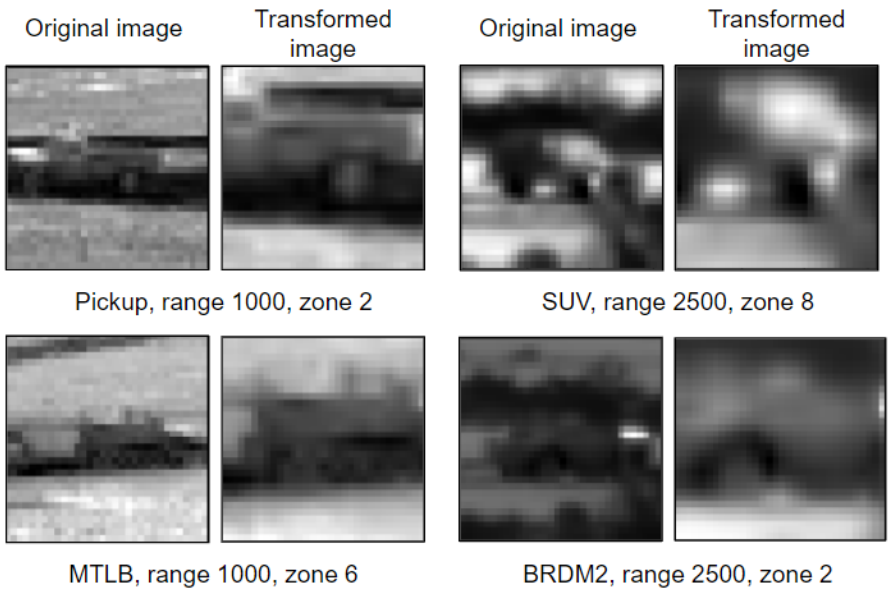

Figure 4.7: Example input patches that are misclassified by RD_CNN and correctly classified by STN_CNN. These samples originally contain irrelevant background (left image). They are transformed by the STN (right image) that highlighted the target in the image by eliminating the background.

its power of focusing on only what is important in the input, thus making the model invariant to noisy background. Fig. 4.7 illustrates examples of input patches that are misclassified by RD_CNN and correctly classified by STN_CNN. We show the original images that contains the irrelevant background and how STN_CNN transformed the patches by removing the background. The STN block can also be perceived as a way to improve the target localization within the patch without causing any distortion due to the rescaling.

Fig. 4.8 displays the distribution of the contrast values of the testing samples that are misclassified by RD_CNN and correctly classified by STN_CNN. We notice that the STN_CNN manages to correctly classify samples with low contrast ≤ 0.2, unlike RD_CNN which tends to misclassify them. Typically, a low contrast occurs when the patch contains a target that blends in with the background. This can happen when data is collected at night and the target is not moving (cold target) or when the data is collected during a hot day and the target is moving (hot target). Under these conditions, the spatial transformers have the additional advantage of being able to smooth the pixel intensity of the image thanks to the sampling kernel. As explained in Sec.2.1.4 and Sec.3.1.3, the STN computes the pixel intensities of the transformed image using a sampling kernel, which in our case is the bilinear interpolation. This sampling kernel generates the pixel intensity of a given location in the transformed image by considering the weighted effects of all the pixels of the

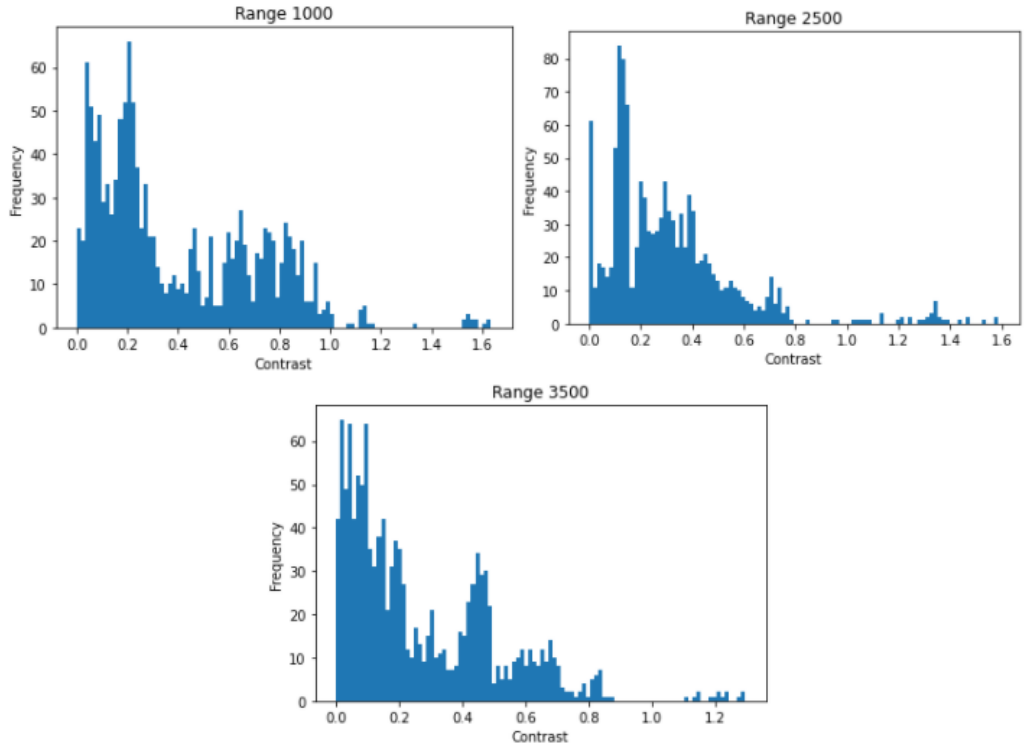

Figure 4.8: Histograms of contrast values corresponding to samples correctly classified by STN_CNN and misclassified by RD_CNN. We conduct our analysis on three testing ranges: a low range 1000, a medium range 2500, and a high range 3500.

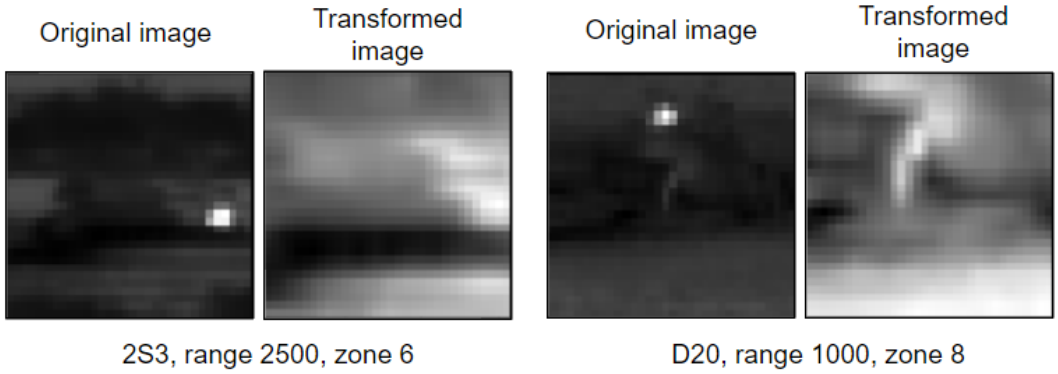

Figure 4.9: Example input patches that are misclassified by RD_CNN and correctly classified by STN_CNN. These samples are originally dark with bright blops (left image). They are transformed by the STN (right image) that helped smooth the pixel intensities.

source image. Therefore, if there are sudden jumps in pixel intensities, they will be smoothed and the resulting transformed image will display a more balanced distribution of pixel values. Fig. 4.9 illustrates example source images that have low contrast and how the STN managed to smooth them by enhancing their contrast.

TABLE 4.3

Number of shifted test boxes by testing range

Range	# samples
1000	360400
1500	351840
2000	384260
2500	377500
3000	336620
3500	333980

4.3.3 RQ3: Are the learned models invariant to small scale and shift variations?

To investigate this research question, we evaluate the robustness of the trained CNN models against potential target localization and scaling errors generated during the detection stage. To this end, we consider two experiments:

- Evaluate the performance of the trained models on shifted and scaled versions of the ground truth bounding boxes. These results are presented, respectively, in Sec.4.3.3.1 and Sec.4.3.3.2.
- Evaluate the performance of the trained models on targets automatically detected by YOLO [7]. These results are presented in Sec.4.3.3.3.

For both experiments, we train the models as described in Sec.3.2.2. We should note that we do not apply any data augmentation during the training stage. Thus, these translated or shifted exemplars are introduced only during the validation step.

4.3.3.1 Impact of localization errors on identification performance

we create a test set that includes horizontally and vertically shifted versions of the ground truth boxes. For a given translation direction, namely horizontal or vertical, we apply a translation of the bounding box. The amount of translation is a predefined fraction of the dimension of the bounding box along the considered translation direction. We apply this procedure both to vertical and horizontal directions. Fig. 4.10 illustrates an example of a vertical translation. We create a dataset with exemplars for translation factors ranging from 0.1 to 0.5 with a step of 0.1, for all four directions. Table. 4.3 summarizes the statistics of the shifted test data.

In Fig. 4.11 and Fig. 4.12, we compare the target identification performance of the considered models under horizontal and vertical localization errors. From Fig. 4.11, it can be noted that the models reach very similar performances for low ranges (1000, 1500 and 2500) where the classification

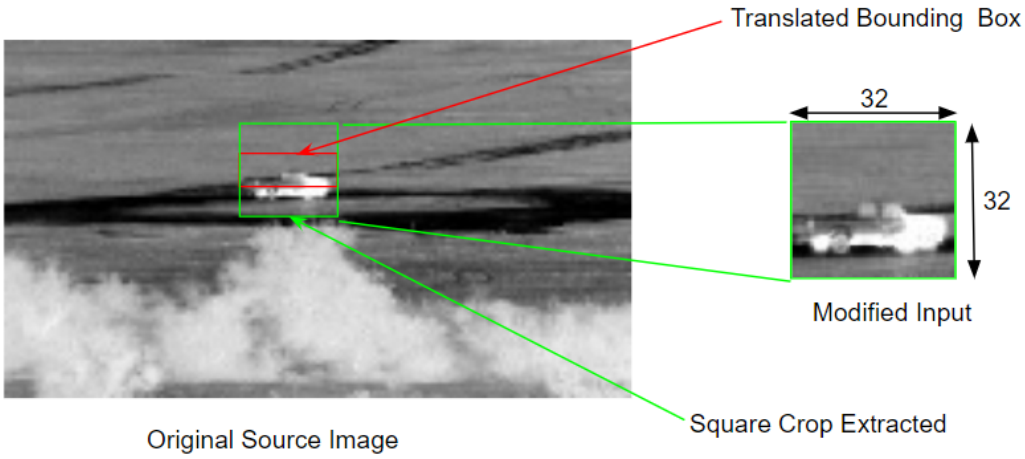

Figure 4.10: An extraction of target exemplars accounting for localization errors: We consider here an upward translation by a factor of 0.3 of the height of the bounding box; we depict both the original bounding box (red) and the translated bounding box (green).

accuracy remains above 80% for horizontal translations ranging from -20% to +20%. However, STN_CNN shows increased robustness over the other CNNs for high ranges (2500, 3000 and 3500). In particular, for range 3500, it preserved a high performance over 85% for horizontal translations ranging from -10% to 10%.

Vertical translations display more oscilations in the behaviour of the three CNNs. In fact, from Fig. 4.12, we can notice that STN_CNN outperforms the other CNNs over all the ranges, mainly for vertical translations ranging from -40% to 30%. However, STN_CNN exhibits a drop in performance for vertical translation above 40%. This misachievement can be even more glaring at the 1500 and 2000 ranges, where the performance drops below 40%. Meanwhile, RD_CNN and FT_CNN succeeded to keep a reasonable performance over all the ranges, specifically at ranges 1500 and 2000.

This phenomenon that causes the STN_CNN to fail over large vertical translations can be explained by the fact that, at a certain point, a large shifting of the position of the bounding box leads to the occlusion of important discriminative parts in the target of interest. This masking effect did not occur at range 1000, because targets captured at such a high resolution have big enough bounding boxes that would not cause major target masking when shifted horizontally or vertically. We also notice that large translations were relatively harmless to STN_CNN for higher ranges (2500, 3000 and 3500), because targets at these ranges fit inside small bounding boxes, and they are only described by few pixels. Therefore, no object part is particularly essential to the discrimination of the class. In this case, the models are insensitive to masked target parts. However, at medium

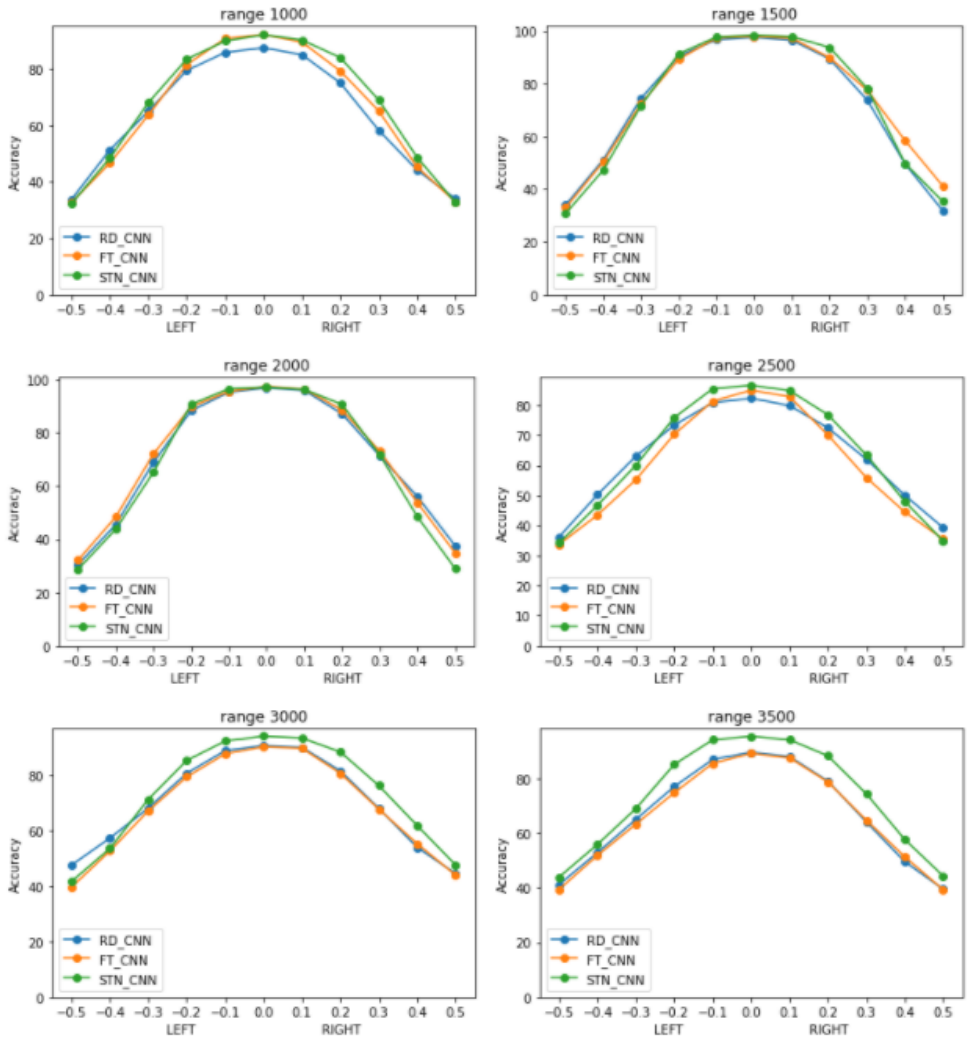

Figure 4.11: Performance results on test targets with horizontally translated boxes. LEFT denotes horizontal shifting of the box to the left, while RIGHT denotes horizontal shifting of the box to the right. We report the results on all testing ranges.

ranges such as 1500 and 2000, targets are large enough to display discriminative object parts, and small enough to be affected by the masking effect of the translated box.

This interpretation can be further supported by Fig. 4.13 where we display example test images with translated bounding boxes. These samples correspond to targets that were misclassified by STN_CNN and correctly classified by RD_CNN. We notice that applying a large vertical translation caused almost a complete disappearance of the target. As we demonstrated in the previous research question in Sec 4.3.1, STN_CNN spatially transforms the input patch to localize exactly the target within its background and focuses only on the relevant discriminative object parts. If these parts are masked due to shifting effects, STN_CNN fails to recognize the target.

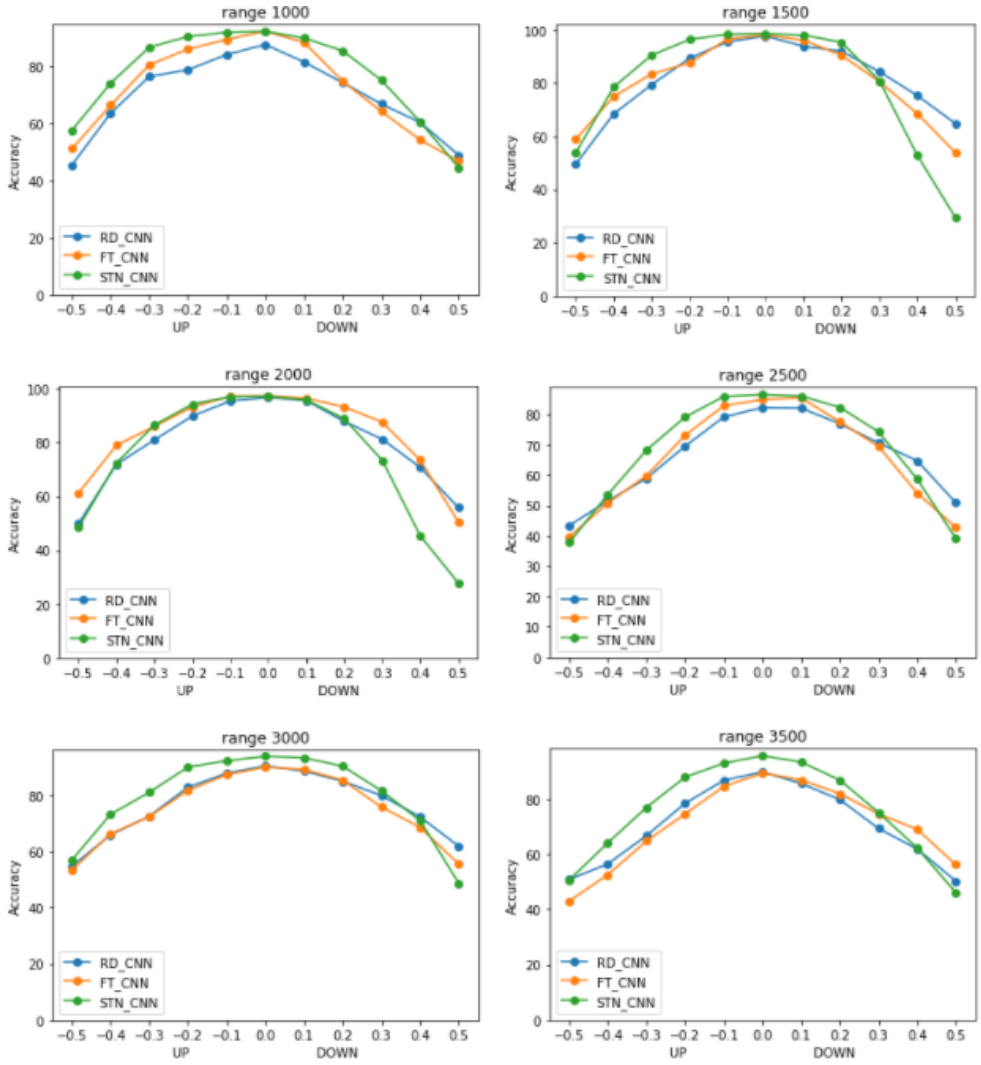

Figure 4.12: Performance results on test targets with vertically translated boxes. UP denotes upper box shifting, while DOWN denotes bottom box shifting. We report the results on all testing ranges.

We argue that the success of RD_CNN and FT_CNN in correctly classifying occluded targets might be a sign that these models are overfitting. Fig. 4.14 supports this claim, where we show example test images that demonstrate the masking effect due to a large vertical shifting of the box. These samples correspond to targets that are misclassified by STN_CNN and correctly classified by RD_CNN. We also display three nearest neighbors from the training set that correspond to each testing sample. To compute the nearest neighbors, we use the learnt features extracted from the last layer of the CNN, right before the classifier. We can observe from Fig. 4.14 that RD_CNN associates, as neighbors, training images that exhibit a particular pattern in their pixel intensities: either a dark background with a bright blob or a bright background with a dark spot. Even if the

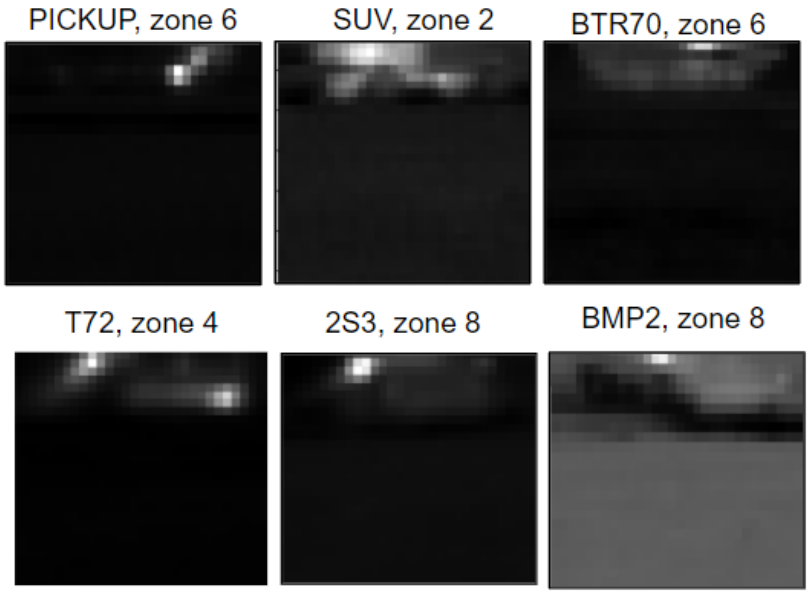

Figure 4.13: Example test images that illustrate the masking effect due to large shiftings of the box position. We consider a vertical translation of 50%. The displayed targets are misclassified by STN_CNN and correctly classified by RD_CNN.

neighbors come from a different class, they are still considered to be close to the test image because they display the same pattern. Therefore, in some cases, RD_CNN might be memorizing the targets by following a certain pattern in the intensity of the pixels, rather than learning and considering discriminative features during the target recognition process.

The overall robustness against translated inputs exhibited by the FT_CNN and RD_CNN, actually derive from the global average pooling layer, which has the ability to make neural networks insensitive to position shifting perturbations.

4.3.3.2 Impact of scale modification on identification performance

To assess the robustness of the models to scaling effects, we create a test set that includes scaled boxes given the ground truth boxes. We rescale the bounding box with a factor ranging from 0.5 to 1.5, keeping the center of the bounding box unchanged, as shown in Fig. 4.15. Table. 4.4 summarizes the details about this created test set.

We report in Fig. 4.16 the robustness performance of the considered models regarding scaling effects. The models achieve highly comparable results in the low ranges, then STN_CNN takes the lead in the higher ranges. STN_CNN also exhibits more robustness toward an increase in the scale of the box. Enlarging the box means that we will be including more background. Therefore, we can interpret the outperformance of STN_CNN in these case scenarios in terms of robustness to dealing

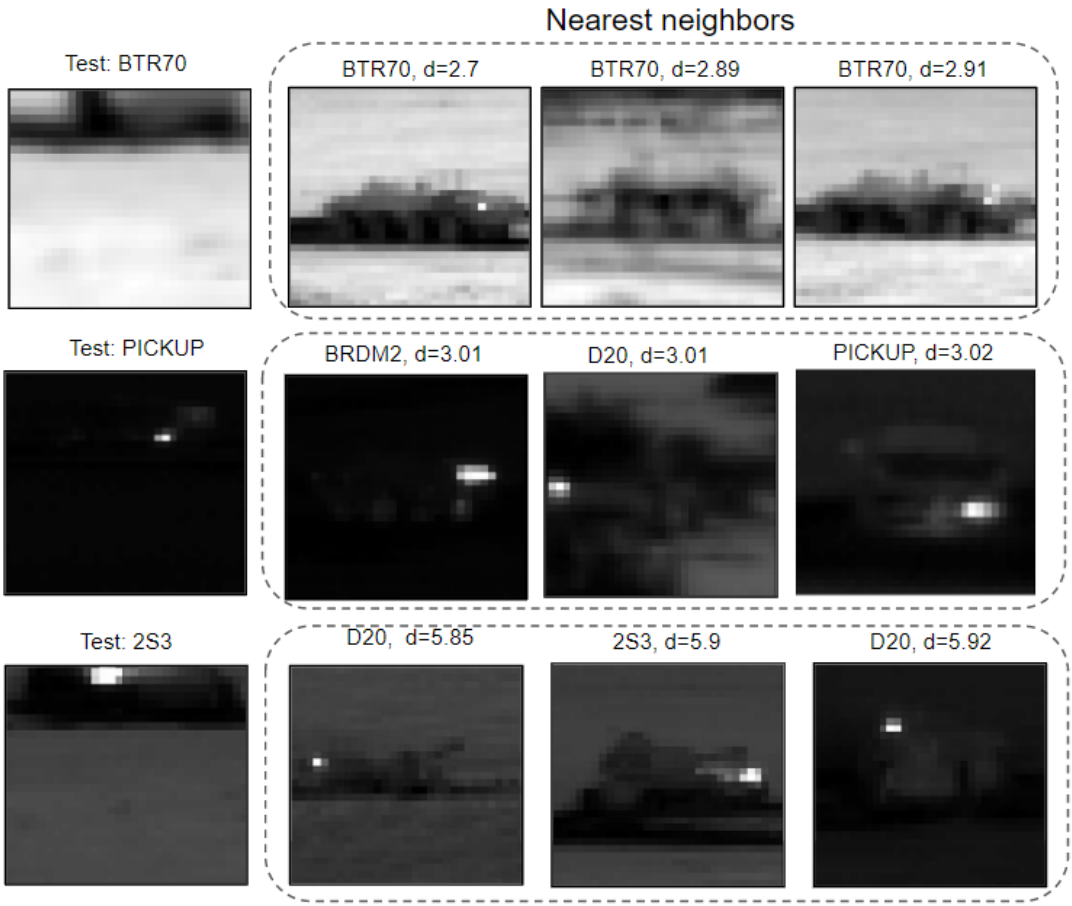

Figure 4.14: On the left we show example test images from range 1500 and 2000 that display the masking effect due to a large vertical box shifting of 50%. On the right, we present three nearest neighbors of the test patch from the training set. We use the features extracted by RD_CNN to compute the nearest neighbors algorithm. We annotate each neighbor by its class and the computed euclidean distance that separates it from the test image.

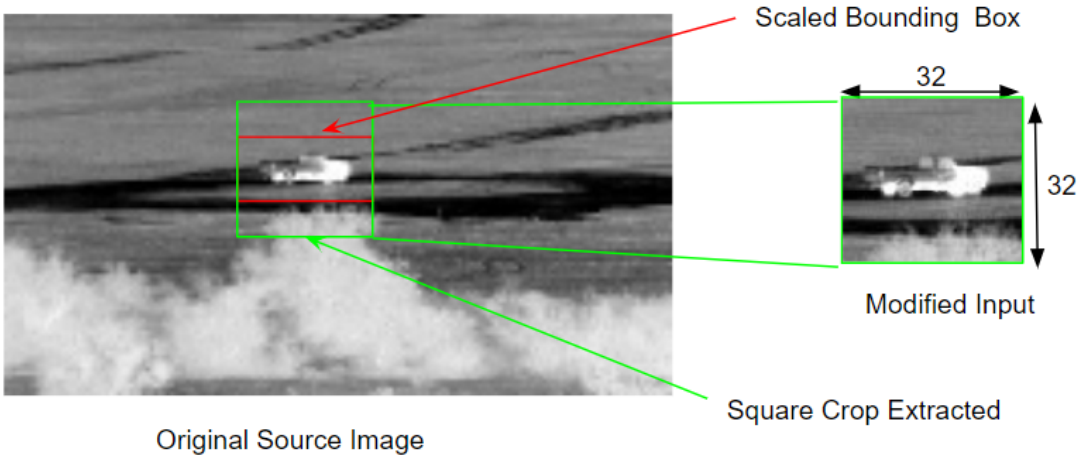

Figure 4.15: An extraction of target exemplars accounting for scaling errors: We consider here a scaling by a factor of 1.3 of the height and the width of the bounding box; we depict both the original bounding box (red) and the scaled bounding box (green).

TABLE 4.4

Number of scaled test boxes by testing range

Range	# samples
1000	360400
1500	351840
2000	384260
2500	377500
3000	336620
3500	333980

TABLE 4.5

Number of YOLO detections by testing range

Range	# detections
1000	37441
1500	37292
2000	40126
2500	38243
3000	33247
3500	25259

with targets situated within irrelevant background, as discussed in the previous research question (Sec 4.3.2).

We can also note that, when the bounding box size is reduced (with a scale factor below 1), some parts of the targets may be masked. Thus, these results may also be interpreted in terms of robustness to masking effects, as discussed in Sec 4.3.3.1.

4.3.3.3 Impact of automatic target detection on identification performance

We assess the ability of the models to handle real-world situations by testing them on targets detected by the object detector YOLO [7]. YOLO has been retrained from scratch on the DSIAC dataset using the annotated ground-truth boxes. Figure. 4.17 describes the different steps we adopt in order to finally generate the class predictions given YOLO detections. Table. 4.5 presents details about the test set generated using YOLO detections.

We analyze the performance of the models using receiver operating characteristic (ROC) curve. The ROC curves display plots of the probability of the detection (Pd) of a given class vs. the probability of false alarm (Pfa). Let "T1" be the event that a target is present, and "T0" the event that a target is not present. The probability of detection (Pd) is the probability of saying that "T1" is true given that event "T1" occurred. The probability of false alarm (Pfa) is the probability

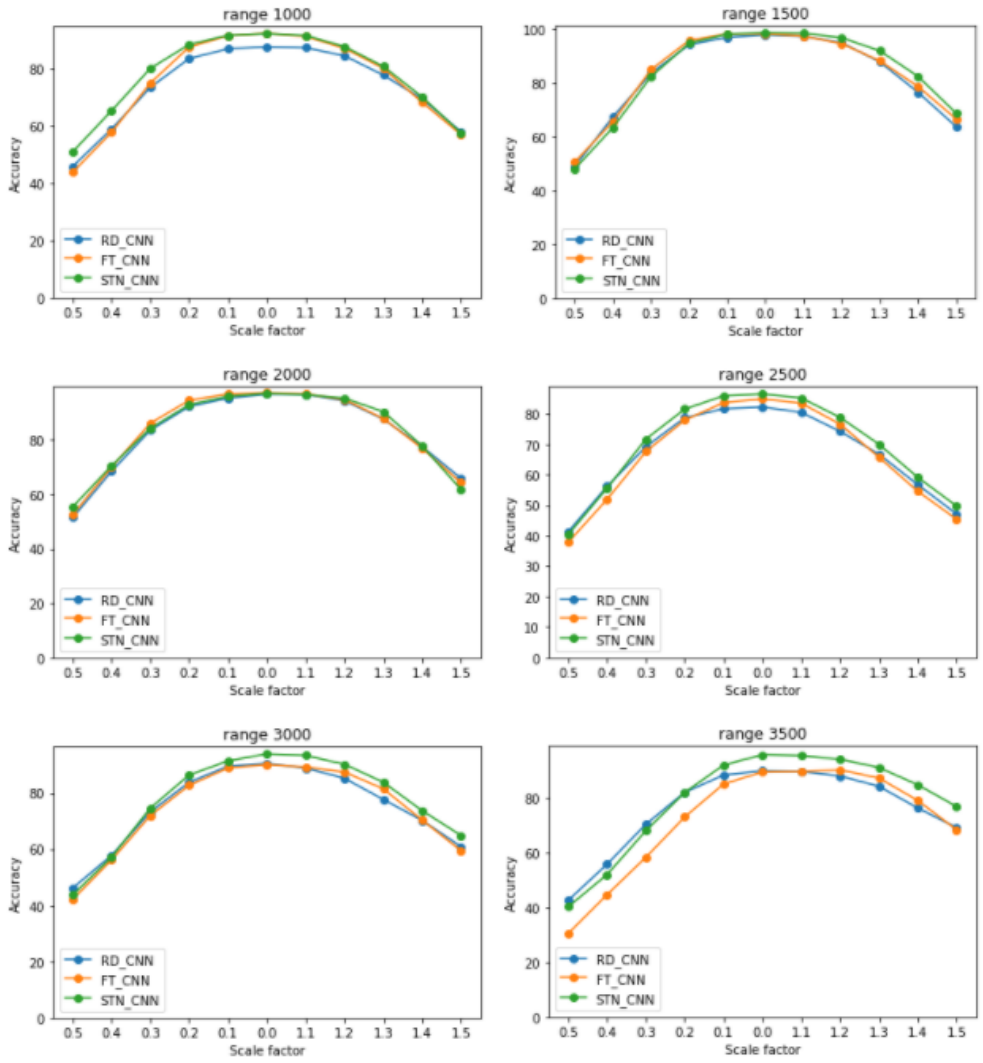

Figure 4.16: Performance results on test targets with scaled boxes. We report the results on all testing ranges.

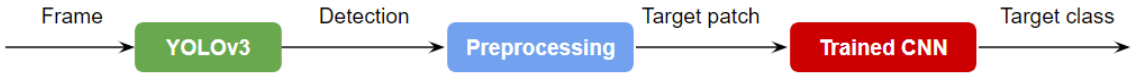

Figure 4.17: Different steps of model evaluation using detected targets by YOLOv3.

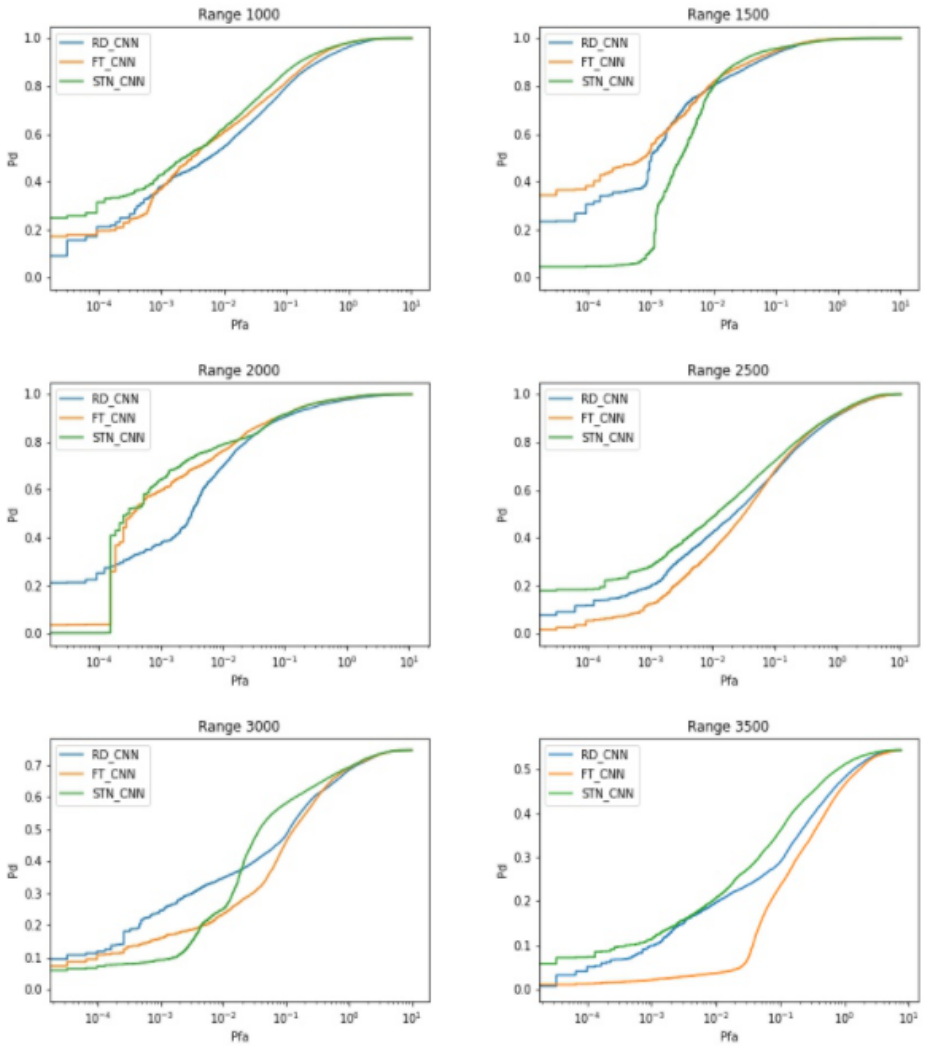

Figure 4.18: ROC curves of the models tested on automatically detected targets by YOLO [7]

of saying that "T1" is true given that the "T0" event occurred.

Fig. 4.18 shows the ROC curves of the tested models RD_CNN, FT_CNN and STN_CNN on the considered ranges. It can be noted that YOLO succeeded to detect 100% of the targets in almost all the ranges, except for the high ranges 3000 and 3500 where it reached a maximum PD around, respectively, 75% and 55%. At such long slant ranges, the targets become too small for the detector to capture.

At range 1000, the ROC curves of the three models are almost overlapping, which indicates that their respective perfomances are highly comparable. Although the ROC curve of STN_CNN is above the ROC curves of the other models, we can fairly judge that the three models perform similarly well on the detected targets at range 1000, by reaching a PD of 80% with a Pfa less than

0.1.

At range 1500, STN_CNN completely fails to recognize the targets for Pfa values less than 10^{-3}, while RD_CNN and FT_CNN score PD values of around 30%. However, STN_CNN, along with the other models, manages to reach a PD value of 80% at a Pfa of 10^{-2}.

At range 2000, STN_CNN and FT_CNN outperform RD_CNN by reaching a Pd around 60% at a Pfa less than 10^{-3}, while RD_CNN realized only a Pd of only 20%.

For the remaining high ranges 2500, 3000 and 3500, STN_CNN takes the lead, followed by RD_CNN and finally FT_CNN. The degradation of the performance of FT_CNN on the high ranges can be, again, traced back to the respective network weight initialization that uses a pretrained model on the VMMRdb dataset [34], where the vehicles resemble the targets of the DSIAC database captured at low ranges.

At range 2500, STN_CNN reaches a Pd of around 60% for a Pfa less than 0.1, while RD_CNN and FT_CNN score a Pd of around 50% for the same Pfa. As for the range 3000, STN_CNN scores a Pd of 60% , out of a maximum of around 75%, for a Pfa of 0.1, leaving RD_CNN and FT_CNN behind with a difference of 10%. Finally, in the range 3500 where the maximum Pd that can be reached is 55%, STN_CNN and RD_CNN are able to attain, respectively, Pd scores of 30% and 25% for a Pfa less that 0.1, while FT_CNN keeps a Pd of around 0%.

Overall, despite the slight advantage presented by STN_CNN, the models are not considered to be performing well enough on the automatically detected targets by YOLO. This misachievement is perceived to be even more severe for targets captured at low resolutions (ranges 2500, 3000 and 3500).

We justify the failure of our models to perform well on the automatically detected targets by the fact that the classification performance is highly dependent on the detection performance. Unfortunately, the ROC metric does not distinguish between these two steps. Therefore, the classifier is penalized for any flaw in the detection stage. In other words, if the detector fails to delimit well the target or generates a false alarm by detecting clutter, the resulting detection, no matter how relevant it is, will be passed anyways to the classifier for class prediction. This leads to an increase in the number of misclassifications, which will be translated in the ROC curve as a bad classifier. Our hypothesis can be further supported by observing that the models show severe failure mainly in the high ranges, where targets are captured at a very low resolution. In this case, the detector is most likely to either miss some targets, poorly delimit them or detect false alarms.

One way to mitigate this situation is to teach the models to identify the false alarms so that

they can be discarded in advance and not be considered for scoring, later, by the ROC.

4.3.4 RQ4: Can the models reject the non-targets identified by the detector as potential targets?

We attempt to boost the ability of the models to identify false alarms generated during the detection stage. To do so, we introduce a new class named "Non-Target". We build this class by collecting the false detections of YOLO. We consider a given detection a false alarm if there is no overlap (overlap=0) between the detected box and its ground truth counterpart. We augment this class in order to balance the total 11 classes by generating random boxes from the background of the frames. Finally, we train the models using the same design described in Sec.3.2.2, with 11 classes. We evaluate the trained models on the test set generated by YOLO detections. We use the ROC curves to report, analyze and compare the results.

We display the resulting ROC curves in Fig. 4.19. We can notice that the ROC curves of STN_CNN indicate an increase in performance compared to the results displayed in Fig. 4.18. This improvement can be observed in all ranges, mainly in range 1500.

However, this observation can be verified to be true for only STN_CNN. The models RD_CNN and FT_CNN do not demonstrate a similar improvement in performance. In fact, these models preserved the same behaviour of their ROC curves compared to their counterparts trained on 10 classes. Their performances even got worse in range 3500 where they scored almost a null Pd for all Pfa values less than 0.1.

We focus our analysis on the range 3500, since in this range the models display the biggest gap in performance. Fig. 4.20 presents the confusion matrices of the three models on range 3500. We can notice that the three models were able to successfully identify the false alarms by scoring an accuracy of 73% in the "Non-Target" class.

However, the three models also misclassified target-labeled detections as non-targets. The confusion is shown to be even more severe for the models RD_CNN and FT_CNN, particularly for the targets BTR70, BMP2 and T72. We investigate these misclassified samples by displaying in Fig. 4.21 the histogram of the overlap ratio. We define the overlap ratio as the ratio between the intersection area of a detected box and its ground truth counterpart, and the area of the ground truth box. It can be noted that STN_CNN misclassified target-labeled detections that mostly have small overlap ratios less than 50%. These detections do not delimit entirely the targets. FT_CNN, on the other hand, misclassified target-labeled detections that have high overlap ratios that range

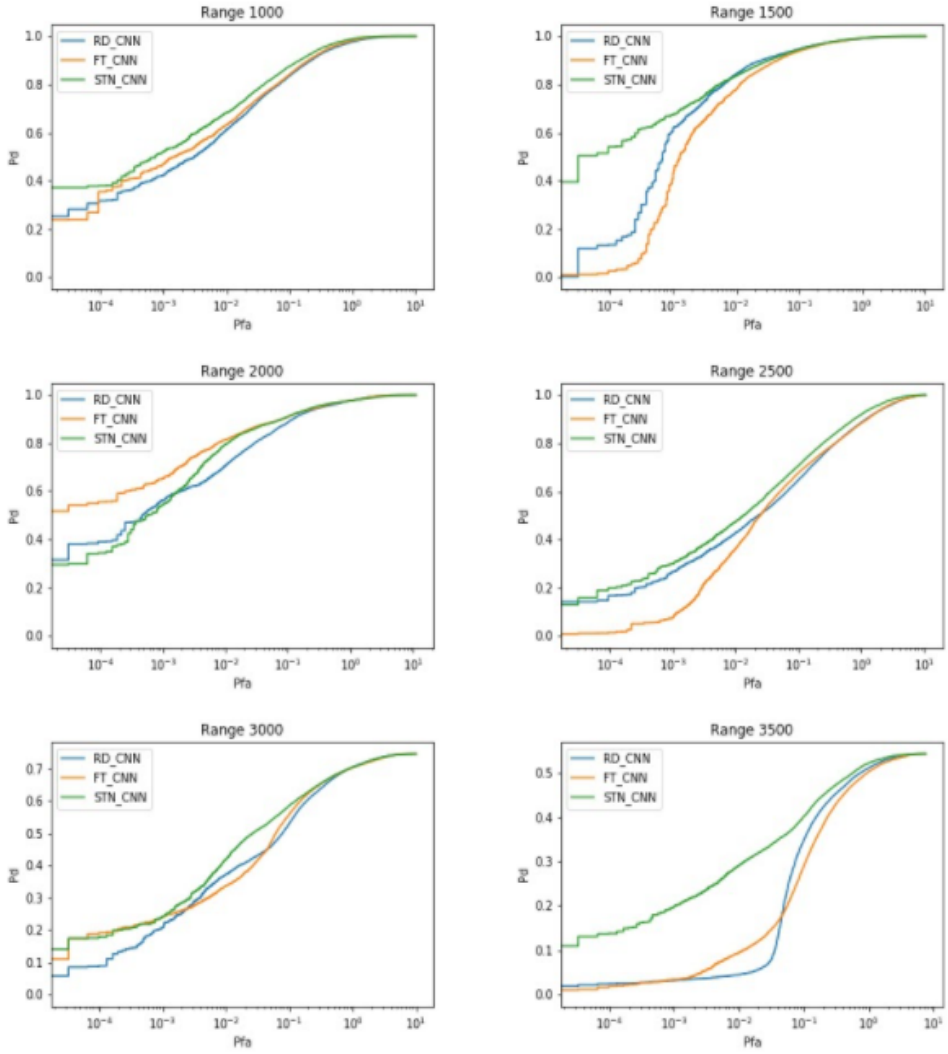

Figure 4.19: ROC curves of the models trained on 11 classes (with "Non-Target" class) and tested on YOLO detections.

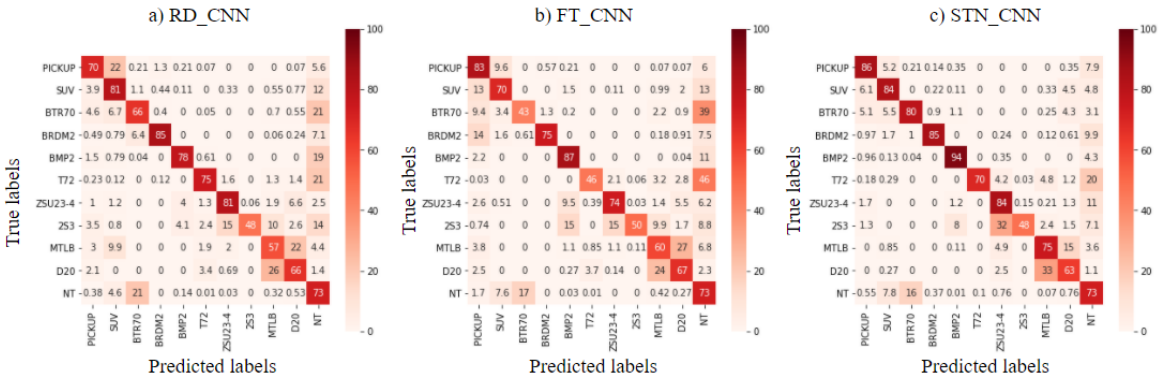

Figure 4.20: Confusion matrices of a)RD_CNN b)FT_CNN, and c)STN_CNN trained on 11 classes and tested on YOLO detections for range 3500. "NT" indicates the Non-Target class. We display the results in terms of percentages.

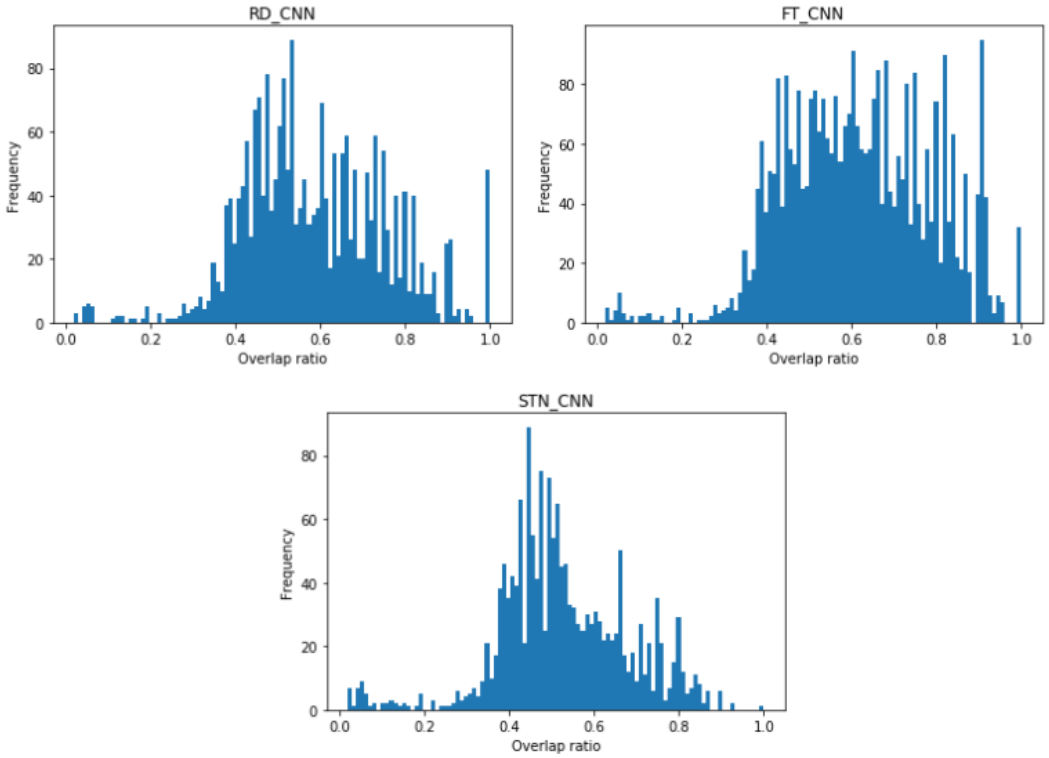

Figure 4.21: Histograms of the overlap ratio corresponding to target-labeled detections misclassified as Non-Target by, respectively, RD_CNN, FT_CNN and STN_CNN. The reported results correpond to range 3500.

from 40% to 100%, with a peak in 90%. This is similar for RD_CNN whose histogram shows high peaks for overlap ratios larger than 50%. This means that RD_CNN and FT_CNN are misclassifying actual targets as non-targets.

Therefore, even though RD_CNN and FT_CNN were able to eliminate the false alarms, they rejected actual target detections in the process, which hindered their classification performance. Meanwhile, STN_CNN succeeded to both eliminate the false alarms and the target-labeled detections that do not relate well enough to the ground truth, probably describing false detections too. Consequently, introducing the "Non-Target" class further boosted the performance of STN_CNN.

CHAPTER 5

CONCLUSIONS

In this thesis, we adapted and investigated the application of CNN models to ATR using infrared imagery. We explored three different learning approaches. The first method relies on random initialization of the network weights. The second method uses transfer learning. The third method introduces a block of STN to the network in order to learn a spatial transformation that attempts to enhance the classification performance.

We evaluated our models on a large-scale collection of infrared images containing civilian and military vehicles. The proposed models performed comparably well on targets captured at high resolution. However, the network equipped with STN displayed superior performance even for targets captured at low resolution. Transfer learning proved to be helpful in boosting the network classification capabilities over the randomly initialized CNN.

We also assessed the power of our models in handling realistic scenarios. The evaluation included testing the models on datasets that simulate the potential perturbations that could be encountered in real-life applications, as well as testing the models on automatically detected targets. In both experiments, the model with the STN block showed the most robustness and provided the best results. Finally, we further boosted the robustness of our models by training them not only to recognize targets, but also to distinguish them from false alarms. This experiment yielded the best results for the STN model.

We concluded that the most reliable, efficient and accurate ATR system would consist of a WideResNet-28-2 preceded by a block of STN. This model was shown to achieve superior classification results for targets captured at both low and high resolutions.

However, our optimal model is much slower to train. As a matter of fact, the STN is a neural network itself. When combined with a CNN, the entire network is trained end-to-end, thus taking longer time to converge than a standalone CNN.

Future work may include investigating efficient preprocessing operations to apply to the data, so that we can generalize the trained model to other test data captured by different sensors

with different calibration.

Another potential future work may include exploring more in depth spatial transformers by using the attention aspect to extract discriminative features in order to explain the model. To do so, the use of parallel blocks of STNs could be considered, so that each block learns a different region of interest.

Finally, in order to overcome the potential overfitting issue that rises from the redundancy of consecutive frames, additional data, captured under different conditions or even generated synthetically, should be considered for training.

REFERENCES

[1] Sergey Zagoruyko and Nikos Komodakis, "Wide residual networks," 05 2016.

[2] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu, "Spatial transformer networks," *CoRR*, vol. abs/1506.02025, 2015.

[3] Facundo Bre, Juan Gimenez, and Víctor Fachinotti, "Prediction of wind pressure coefficients on building surfaces using artificial neural networks," *Energy and Buildings*, vol. 158, 11 2017.

[4] Sumit Saha, "A comprehensive guide to convolutional neural networks-the eli5 way," Dec 2018.

[5] "Complete guide of activation functions," .

[6] "A comprehensive hands-on guide to transfer learning with real-world applications in deep learning," https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a, Accessed: April 2019.

[7] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015.

[8] James Ratches, "Review of current aided/automatic target acquisition technology for military target acquisition tasks," *Optical Engineering - OPT ENG*, vol. 50, 07 2011.

[9] "Infrared and thermal imaging system benefits and applications," Feb 2018.

[10] He Deng, Xianping Sun, Maili Liu, Ye Chaohui, and Xin Zhou, "Infrared small-target detection using multiscale gray difference weighted image entropy," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52, pp. 60–72, 02 2016.

[11] Seok Yoon, Taek Lyul Song, and Tae Kim, "Automatic target recognition and tracking in forward-looking infrared image sequences with a complex background," *International Journal of Control, Automation and Systems*, vol. 11, 02 2013.

[12] G. J. Gray, N. Aouf, M. Richardson, B. Butters, Roy Walmsley, and E. Nicholls, "Feature-based target recognition in infrared images for future unmanned aerial vehicles," *Journal of Battlefield Technology*, vol. 14, pp. 27, 2011.

[13] Zhi-Guo Cao, Xuan Zhang, and Wenwu Wang, "Forward-looking infrared target recognition based on histograms of oriented gradients," pp. 27–, 11 2011.

[14] Konstantinos Makantasis, Antonis Nikitakis, Anastasios Doulamis, Nikolaos Doulamis, and Ioannis Papaefstathiou, "Data-driven background subtraction algorithm for in-camera acceleration in thermal imagery," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. PP, pp. 1–1, 06 2017.

[15] Aparna Akula, Arshdeep Singh, Ripul Ghosh, Satish Kumar, and Hk Sardana, *Target Recognition in Infrared Imagery Using Convolutional Neural Network*, vol. 460, pp. 25–34, 12 2017.

[16] E. Gundogdu, A. Koç, and A. A. Alatan, "Object classification in infrared images using deep representations," in *2016 IEEE International Conference on Image Processing (ICIP)*, 2016, pp. 1066–1070.

[17] Iain Rodger, Barry Connor, and Neil Robertson, "Classifying objects in lwir imagery via cnns," 10 2016, p. 99870H.

[18] Antoine d' Acremont, Ronan Fablet, Alexandre Baussard, and Guillaume Quin, "Cnn-based target recognition and identification for infrared imaging in defense systems," *Sensors*, vol. 19, no. 9, pp. 2040, Apr 2019.

[19] "Computer vision: What it is and why it matters — sas," https://www.sas.com/enus/insights/analytics/computervision.html, (Accessed on 10/15/2020).

[20] "Deep learning achievements over the past year — by eduard tyantov — cube dev," https://blog.statsbot.co/deep-learning-achievements-4c563e034257, (Accessed on 10/15/2020).

[21] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[22] David G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[23] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005, vol. 1, pp. 886–893 vol. 1.

[24] Jorge Sánchez, Florent Perronnin, Thomas Mensink, and Jakob Verbeek, "Image classification with the fisher vector: Theory and practice," *Int. J. Comput. Vision*, vol. 105, no. 3, pp. 222–245, Dec. 2013.

[25] Yann LeCun, Y. Bengio, and Geoffrey Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–44, 05 2015.

[26] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei Fei Li, "Imagenet: a large-scale hierarchical image database," 06 2009, pp. 248–255.

[27] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1701–1708.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[29] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, Oct 2010.

[30] David G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[31] Theodoros Evgeniou and Massimiliano Pontil, "Support vector machines: Theory and applications," 01 2001, vol. 2049, pp. 249–257.

[32] Danqing Liu, "A practical guide to relu," Nov 2017.

[33] James Dellinger, "Weight initialization in neural networks: A journey from the basics to kaiming," Apr 2019.

[34] Faezeh Tafazzoli, Hichem Frigui, and Keishin Nishiyama, "A large and diverse dataset for improved vehicle make and model recognition," 07 2017, pp. 874–881.

[35] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[36] "Advanced recording format - just solve the file format problem," http://fileformats.archiveteam.org/wiki/AdvancedRecordingFormat.html, (Accessed on 11/02/2020).

[37] "Json," https://www.json.org/json-en.html, (Accessed on 11/02/2020).

[38] "torch.utils.data — pytorch 1.6.0 documentation," https://pytorch.org/docs/stable/data.htmliterable-style-datasets, (Accessed on 10/20/2020).

CURRICULUM VITAE

NAME: Nada Baili

ADDRESS: Computer Engineering & Computer Science Department
Speed School of Engineering
University of Louisville
Louisville, KY 40292

EDUCATION:

M.S., Computer Science
December 2020
University of Louisville, *Louisville, Kentucky*

B.Eng., Polytechnic Engineering
June 2019
Tunisia Polytechnic School, *Tunis, Tunisia*

PROJECTS AND INTERNSHIPS:

1. Graduate Research Assistant in University of Louisville, February 2019 - Today
2. Engineer intern in EURA NOVA, June 2018 - July 2018

HONORS AND AWARDS:

1. CECS Master of Science Award, April 2020
2. Tunisian National Scholarship for Engineering Studies, September 2016