A VICTIMS DETECTION APPROACH FOR BURNING BUILDING SITES USING CONVOLUTIONAL NEURAL NETWORKS

by

Farah Bilal Jaradat, B.S.

A thesis submitted to the Graduate Council of
Texas State University in partial fulfillment
of the requirements for the degree of
Master of Science
with a Major in Engineering
December 2019

Committee Members:

Damian Valles, Chair

William Stapleton

George Koutitas

DEDICATION

To my mother...

ACKNOWLEDGEMENTS

I am thankful to each of the members of my thesis committee who has provided me extensive personal and professional guidance and taught me a great deal about both scientific research and life in general. I would especially like to thank Dr. Damian Valles, the chairman of my committee as well as my thesis supervisor. As my mentor, with his time, effort, guidance and vision, he has taught me more than I could ever give him credit for. He consistently allowed this paper to be my work but steered me in the right direction whenever he thought I needed it. I would also like to thank him for providing me with the technology required to accomplish this work through HiPE servers.

The most important part during the span of this thesis was the support and love of my family. I would like to thank my parents; whose love and care gave me strength and who are always supportive of what I pursue. Also, I would like also to thank my supportive husband who was always there when I need inspiration and power to move on. Finally, big thanks to my baby “Dahlia” for being my strongest motivation to finish this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

In this work, an approach is proposed to detect people and pets trapped in burning sites. This will be done in a manner that will ensure the safety of firefighters and accelerate the rescue process of victims. The proposed work suggests detecting victims in fire situations autonomously, through a deep learning technique using the Convolutional Neural Network (CNN) model. Research work in the field of firefighters' assistance suggest using infrared (IR) sensors, which are widely used in this field due to their functionality in fire situations, since such cases are associated with high levels of smoke, that limit the vision as well as the feature of white light cameras since the smoke particles, reflect the light. Thus, thermal sensors found to be a convenient choice for gathering data for detecting victims in the burning site without distortion of smoke. Despite the huge progress is the field of firefighting assistance due to the huge technological advancement, however, deciding if victims exist inside the burning structure without human interference has remained a challenge. Here deep learning model is introduced to identify victims' appearances in IR images. What distinguishes this work is leveraging IR imaging combined with deep learning models, which were proved to be the state-of-the-art techniques in the field of people detection. The proposed method is extended to detect pets in the burning sites. The thesis work provides a deep learning approach that speeds up and facilitates the process of rescuing victims in vague burning sites layouts; hence, more lives would be saved. The objective of the CNN development is to classify input images sent from the burning site into one of three outcomes classes: “*people*,” “*pets*,” and “*no victims*.” First responders can leverage this information to define their priorities regarding the locations they should target first. Deep learning is a popular method for human detection in the IR image, and it is applied extensively to the field of pedestrian detection and surveillance. However, the literature scarce for applications in fire assistance fields. One reason behind this, is the unavailability of public IR datasets, including infrared images of people and pets. Therefore, a dataset was developed to accomplish the task of human detection in IR images. Furthermore, the dataset was processed to mimic high-temperature environments as in building on fire situations. In this thesis, a cascaded CNN architecture approach was implemented and benchmarked, and the cascaded model consists of two stages to accomplish the task of victim detection in IR images.

I. INTRODUCTION

Problem Statement

Structure fires are common type of incidents, where U.S. fire departments responded to an estimated average of 355,400 home structure fires per year during 2012-2016. These fires were responsible for about 2,560 deaths per year in average. Firefighting techniques are constantly evolving due to the technological advancement. Indeed, no aspect in firefighting has remained untouched, and this advancement caused firefighting to adopt new methods to put out fires and save lives of both victims and firefighters.

Moreover, this technological revolution introduces the smart city concept, where a smart city can be defined as an urban area with structures equipped with sensors network and communication means between these structures. Inspired by the smart city concept, smart firefighting model was developed. The current smart firefighting concept relies on stationary sensors placed into buildings, existing data from building plans and building inspection. However, this model can be further enhanced by incorporating artificial intelligence (AI) methods to firefighting.

In this work, a solution is being researched to introduce AI concept to fire situations in order to increase the efficiency of firefighters' role in saving victims by detecting humans and pets in ambiguous burning sites autonomously through classification of thermal images using deep learning techniques. Providing a Convolutional Neural Network (CNN) with a thermal Infrared (IR) images with position information from the burning would result in a detection whether humans or pets are present at any location. The CNN would be trained to do this task by feeding it with a large number of labeled thermal images collected by an IR camera.

In a fire situation, high levels of smoke would hinder the vision of human eyes, as well as the white light cameras. Thermal IR cameras are found to be suitable for this kind of missions since they are not affected by smoke and detect relative differences in the amount of thermal energy emitted from objects. Thermal properties of a foreground object are slightly different either higher or lower from the background radiation, and the corresponding region in a thermal image appears at a contrast from the environment.

IR images would be captured from the burning site and transmitted to a base computer station through an Autonomous Embedded System Vehicle (AESV) [1]. A CNN model will run on the base computer to indicate if a human or a pet is detected in the IR images. The CNN model will analyze each IR image to determine one of three classes: “people,” “pets” or “no_victims.” The proposed CNN model design would enhance firefighters’ safety and performance when evacuating victims from burning sites by helping them set their priorities for rescue protocols.

The AESV is another aspect of the project that is worked on by other member in High-Performance Engineering (HiPE) research group. The AESV design equipped with wireless communication system in which transmits gathered data to a base computer station located in a firetruck outside the burning site [2]. The purpose of the AESV is to navigate itself through a burning site and broadcast critical data to be analyzed in real-time. The AESV is equipped with several sensors to collect environmental, positional, and video data from the site, through temperature sensors, gas sensors, an IR camera, and other peripherals.

Usually, when firefighters arrive to the fire location, they navigate themselves in the burning building, then they will start looking for victims inside the burning building. The expectation is both the responder and victim experience very dangerous temperatures and carbon monoxide that threatens their lives. Furthermore, this way of victims' detection is time consuming. However, the proposed method in this work suggests providing the firefighters with guidance before they get into the burning structure so they spend less time in the hazardous environment, and the victims' rescue process would be performed in more effective manner. The next figure 1 shows the proposed smart firefighting model in this work.

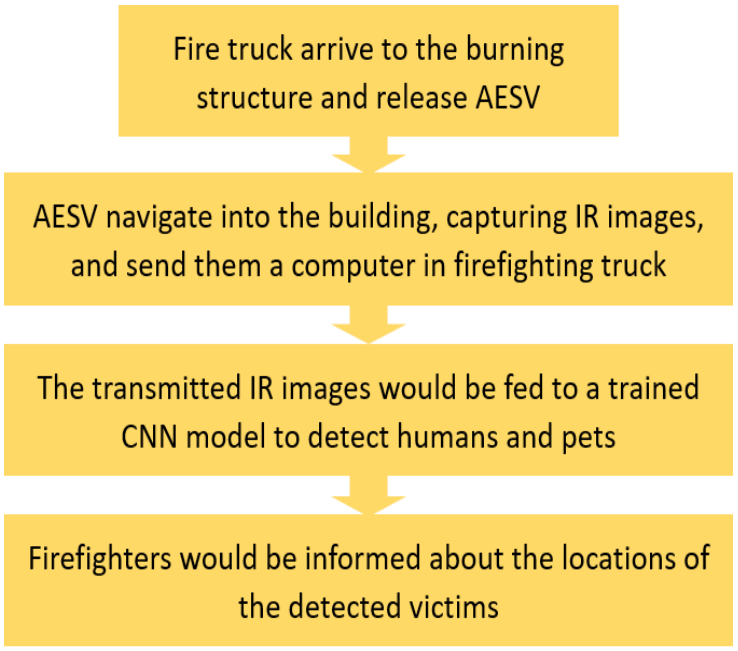

Fire truck arrive to the burning structure and release AESV
AESV navigate into the building, capturing IR images, and send them a computer in firefighting truck
The transmitted IR images would be fed to a trained CNN model to detect humans and pets
Firefighters would be informed about the locations of the detected victims

Figure 1. Flowchart demonstrating the proposed solution for victims' detection

The CNN is a type of Neural Network (NN) architecture based solely on the convolution principle. Where compared to standard feed-forward NNs with similarlysized layers, CNNs have fewer connections and parameters making them easier to train [3]. CNNs also provide outstanding results of the feature-learning approach based on deep learning techniques, and this is the reason behind CNNs' popularity in computer vision and recognition system specifically.

Firefighters assistance is an active field of research, researchers and engineers develop work in the field of enhancing firefighters' safety and performance. Different approaches are proposed for assisting firefighters, such as providing them with specially designed cameras to help them see through smoke, and some work suggests giving support to the first responders with robots equipped with sensors. Deep learning techniques were recommended for object recognition problems in burning sites and proven to be successful.

Human detection is a challenging task for computers due to the various poses of the human body, the changes in illumination, and the complicated background [4]. In the case of mobile robots, the task of detecting humans become more challenging since the robot is moving, and the environment is unpredictable. The nature of the proposed work puts challenges that the CNN model needs to handle, such as the variety of camera angles, distances from that the AESV would capture the IR images, and the low-resolution of the IR camera used by the AESV. Hence, a well-designed dataset would be gathered using an IR camera to train the proposed CNN model, and suitable architecture of CNN would be investigated. The dataset would include a variety of categories in an attempt to ensure covering all possible cases and looks of humans in the IR images transmitted from the burning site.

Contributions

The contributions of this thesis can be summarized as follows:

• Development of an IR dataset that displays human's and pets' appearances in high temperature environments, where such type of datasets is not available from open online sources.

• Object detection using deep learning is applied to IR images of people in different poses and with varying backgrounds. An application that is rarely appearing in current literature.

• Proposing solution to deal with the imbalanced classes sizes, where cascading two different CNN models were suggested.

Thesis Structure

The thesis begins with background chapter that discusses the motivations behind this research work is reviewed. The following chapter is intended to show the relevant literature and methods in related fields to this work, including firefighter assistance and human detection in thermal images. Chapters 4 and 5 are two theoretical chapters discussing convolutional object detection as a combination of several fields of computer science. Chapter 4 begins with a short introduction to artificial intelligence and machine learning. Also, NN structure and learning algorithms used in training them were discussed. In chapter 5, discussion of computer vision and the role of CNNS in image recognition is covered.

The details of the developed dataset for this research and the various processing steps applied to get the final dataset version are appraised in chapter 6. In chapter 7 proposes methodology for victims' detection system, and further details of each module are introduced. In chapter 8, selected experiments results would be presented and discussed, in an attempt to justify the choices and mechanisms adapted to implement the deep learning model for this work.

Chapter 9 is devoted for providing comprehensive study of a two-step cascaded CNN model that is developed in the course of this thesis as an extended work. Details of this model along with the results achieved are viewed and assessed. This research work is concluded in chapter 10. Finally, in chapter 11 some implications for further research are described.

II. BACKGROUND

This section is intended to provide an overview of the factors and considerations that contributed to this thesis project. It also reviews the challenges and obstacles that firefighters face on the ground and discuss the necessity of an engineering solution for helping the firefighters to accomplish their tasks efficiently and effectively.

Firefighters are among the frontline workers who serve and protect the public by responding to fires and other emergencies and regularly work in deplorable conditions, so they are more likely to be injured at fire ground operations than at different types of duties. According to the National Fire Protection Association (NFPA), it reports firefighter injuries in the United States estimate that 58,835 firefighter injuries occurred in the line of duty in 2017. In addition to injuries, there were 44,530 exposures to hazardous conditions. Indeed, inhalation of toxic smoke is a significant cause of firefighter death. Smoke inhalation causes acute life-threatening injuries and results in long-term lung and neurological damage. Many toxic products are released during a typical room-and-contents fire, and even though firefighters' protective equipment includes protective masks with filters, but a gas mask will offer minimal protection and is not enough to save fireman life in a fire. Another problem associated with dense smoke areas is the limitation in visual perception. The smoke particles reflect the light preventing the clear vision of both human and white light cameras as well, which in turn would impair firefighters’ ability to explore the burning site searching for victims. Thus, smoke existence is a major factor affecting the imaging method adapted for investigating the burning areas. Therefore, mechanisms that can help explore dense smoke areas without the intervention of human has the potential to speed up and facilitate the process of rescuing victims in a burning site while reducing the risk levels to humans.

Modern technology provides the means to collect and analyze a huge amount of data. As a result, fire service is on the cusp of a new era, and smart firefighting is paving the way for smarter, safer, and more effective firefighters of the future. Smart firefighting is a result of the technological revolution in communication and the sensor-rich environment available nowadays, and it has the potential to enhance the performance of firefighters, through incorporating sensors, communication means, and other technologies into the fire protection and alarm systems. However, despite the vast advancement in the field of firefighting, there are still several issues that need to be addressed. These include the management of the data and information received by the firefighters on the ground, so they do not get overwhelmed, where smart firefighting is responsible for receiving too much data from networks of sensors. Also, in situations when the burning site lacks the infrastructure of smart firefighting, or not being equipped with sensors or communication means to support firefighters in making decisions regarding the presence of victims inside. In such cases, the firefighters have to navigate themselves in the burning building, searching for people and pets in need of rescue. This procedure would not be time-efficient since no information is available regarding the locations that first responders should target first and would expose firefighters to hazards.

In an attempt to overcome the stated challenges and improve the performance of firefighters in evacuating and saving victims trapped in burning sites, while considering the safety of firefighters, a large scope project aims to provide the first responders with information regarding the location of victims inside different layouts of burning buildings. The scope of the project consists of two efforts. The first research work entails the hardware design of an Autonomous Embedded System Vehicle (AESV) that is equipped with sensors, a navigation system, and an Infrared (IR) camera. The second research effort entails the machine learning that consist of a CNN architecture trained to recognize humans and pets with the thermal IR images that would be captured and received from the AESV. The focus of this research work is focused on solving the second effort of the overall firefighting project.

III. LITERATURE REVIEW

Firefighters Assistance

In this section, relevant literature would be reviewed in the field of firefighters assisting, and human detection in thermal images with a particular focus on methods leveraging deep learning techniques, which proved to be exceptionally reliable.
IR cameras usage as firefighting assistive equipment is common, due to their ability to detect IR radiation even in dense smoke or dark environments. The authors in [17], [6], and [7] enhancing methods are investigated to increase the reliability of IR cameras performance in burning sites.

Fritsche et al. [17] propose using a mobile robot equipped with LiDAR sensor, a radar sensor, and a thermal imaging camera to detect hazards that are potentially harmful to the robot or firefighters in low visibility environments. In an attempt to overcome the limitations of optical sensors, such as laser scanners and RGB cameras, in low visibility environments. The research work implemented its solution by fusing LiDAR and radar data to generate two-dimensional obstacle scans that allow environment perception under low visibility conditions. The proposed methodology showed a distinct advantage over using pure laser scan.

Bianco et al. [6] suggest providing the firefighters with portable, lightweight IR laser system suitable for on field applications, to help them in looking for people through smoke and flames in fire-scenes. The proposed system aims to enhance the functionality of IR cameras, as flames existence in the scope of the infrared cameras impair their functionality, by equipping the IR sensor along with a narrow band-pass filtering.

Experiments conducted successfully proved the functionality of the proposed system when flames exist in the scene.

The authors in [7], advocate supplementing the IR camera with a compact W-band radar sensor is proposed, to enhance depth perception, since the IR sensors don't provide the firefighter with an estimation of the depth, distance of the obstacle, or a victim. The radar module has successfully provided measurements of the length to different targets in a smoked filled closed fire training system, where two marks to varying distances from the radar were set up.

However, the proposed methods above cannot determine if victims are present in the burning site or not in an autonomous manner. Thus, computer vision-based human detection would be more effective in detecting human in burning sites autonomously using thermal images.

Human Detection

This section presents proposed ideas for detecting human in thermal images using image processing techniques and pre-trained neural networks to decide human detection in the site.

1. **Using Hand Crafted Algorithms and Image Processing Techniques:**

   The goal of the work proposed by Ćirić et al. in [8] is to enhance tracking and detecting people using mobile robot platform. The system relies on fuzzy segmentation and Fuzzy Neural Network (FNN) classification. The motive behind going with the fuzzy segmentation is that the basic segmentation algorithms have problems in determining the Region of Interest (ROI) in real-world scenarios. The fuzzy segmentation creates fuzzy membership, which would be an indication of how strongly each RGB color belongs to the background or to the foreground.

The difficulty of detecting humans in thermal images considering temperature environment variations are discussed in [9] and [10]. In [9], the authors present an approach based on the method of background subtraction. The proposed method can be summarized as follows: 1) generating a background image by image averaging, detection, and erasing methods of human areas from the input images, 2) obtaining the difference between the background generated image and an input image using brightness thresholds, and 3) detecting humans in the difference between the images. The experimental results confirmed that the detection accuracies of the proposed method were higher than other methods that leverage background generation but without adjusting parameters of detection based on background information.

Whereas, in [10], Trofimova et al. present a method based on leveraging an adaptive background algorithm and noise removal technique based on Kalman filter to detect people in the two-dimensional thermal area. Low-Resolution Thermal Array Sensor: LR-TASs, composed of an array of sensing elements are used to produce the two-dimensional thermal area. Final results show an improvement in the human detection accuracy compared with prior work in [11].

2. Using Deep Learning Techniques:

Although hand-designed heuristics such as discussed in part a "Using Hand Crafted Algorithms and Image Processing Techniques" could achieve high accuracy levels in detecting people, better human detection systems can be built by relying more on automatic learning. Deep learning techniques are made possible by recent progress in machine learning and computer technology.

One of the most influential works in the field of object recognition is by using the Convolutional Neural Network (CNN) found in [12] as the ImageNet Classification with Deep Convolutional Neural Networks (DCNN). In this paper, Alex et al. proposed using five-convolutional layers followed by three fully-connected layers network which uses large datasets to enhance object recognition CNNs performance when used for wide-range of classes. The proposed design achieved top-1 and top-5 error rates.

In [13], Girshic et al. proposed a simple and scalable detection algorithm that improves mean average precision by extracting region proposals in the input data point, then classify each region. This proposed design is called Regional Convolutional Neural Network (R-CNN). However, the cost of the regional proposal step in the proposed R-CNN is an expensive step since it is consuming as much time as the detection network. Therefore, in [14] Ren et al. proposed enhancing the R-CNN by adding additional convolutional layers that simultaneously regress region bounds and classifies objects at each location to reduce the cost of regional proposal step.

A significant application of human detection using CNNs is for pedestrian detection, which is essential for advanced driver assistance systems (ADAS). Chavdarova et al. [15] discussed the problem of occlusion and how to enhance pedestrian detection in general, by using the state-of-the-art object detection CNN used in monocular pedestrian detection after tuning them to be suitable for multi-view dataset.

John at al. [16] are proposing leveraging the deep learning techniques in pedestrian detection in thermal images. The proposed approach suggests using adaptive fuzzy C-means clustering to define candidates in the thermal image, then further pruning the resulting candidates relying on human posture characteristics, and second central moment ellipse to get bounding boxes of different sizes around the candidates. The approach should reduce the computational complexity associated with classifying sliding window-based object detection. Thus, the CNN job is to classify the candidate pedestrian bounding box as a pedestrian or non-pedestrian. Another classification work of real persons, mirrored-persons, and objects in thermal images was developed by Ulrich et al. [5]. In an attempt to help firefighters in distinguishing real from mirrored persons using IR cameras to explore and find victims in burning sites. The difference is accomplished by using different fusion architectures of IR sensor along with micro-doppler radar. The micro-doppler radar produces a μ-D spectrum of the target, and it is approved to be the suitable choice for this mission since the μ-D spectrum of mirrored standing person is different from the μ-D of a real standing person. Three architectures of fusing IR sensor and micro-doppler radar are investigated in the approaches bounding boxes of the upper body of real and mirrored persons:

1. Measurement level fusion: Classification of the radar μ-D and the associated image patches of the detected bounding boxes yields the class of the object.

2. Object level fusion: classifying the IR bounding box and the radar μ-D separately before association.

3. Performing a standard classification of the whole IR image (no bounding boxes) and the radar μ-D (scene analysis).

Testing the three architectures in different scenes shows that the measurement-level and object-level fusion perform best for distant targets. On the other hand, the joint classification performs best for near targets.

The Scope of Thesis Research Work

The scope of this research work is to integrate human and pet detection through a thermal camera in an Autonomous Embedded System Vehicle (AESV). The AESV design imposes some challenges on the proposed CNN design in movements, angles, and distances. The AESV is not a static body that carries multiple sensors including a thermal camera. Also, victim's detection has to be successful in different poses and sizes due to the thermal camera angle facing upwards. Distance and resolution of the thermal camera come to consideration for the victim or object detection from the captured frames of the AESV. This work targets a high-temperature environment with high levels of smoke. The process of gathering data requires carefully selected tools to get useful data for the process of classification of the captured IR images.

IV. ARTIFICIAL NEURAL NETWORKS (ANN)

Deep Learning is a type of Artificial Intelligence (AI) that is based on the use of algorithms to learn from data. Where AI is a broader concept defined as intelligence exhibited by machines, and the use of computers to mimic the cognitive functions of humans. AI has many applications in today's society such as robotics, knowledge reasoning, and machine learning. Machine learning is the subset of AI that is responsible for providing systems the ability to automatically learn and evolve from experience from data, and observations available without being explicitly programmed. However, updating machine learning algorithms adjust as they learn more about the data being processed.

Training computers to behave like humans' brains is achieved partly through the use of Artificial Neural Networks (ANN). ANNs are a series of parametric algorithms where its design is inspired by the structure of biological neural networks that constitute the human brain. As the number of hidden layers in a neural network architecture increases, the model capability to handle more complex classification tasks increases, and it is referred as a deep network. Deep learning belongs to the family of ANN algorithms. However, Deep learning networks are distinguished from the more commonplace single-hidden-layer neural networks by their depth; that is, the number of node layers through that data must pass in a multistep process of pattern recognition. There is no consensus amongst experts on the depth of a network to be considered. For a graphical depiction of the relationship between AI, machine learning, neural networks and deep learning, please refer to Figure 2.

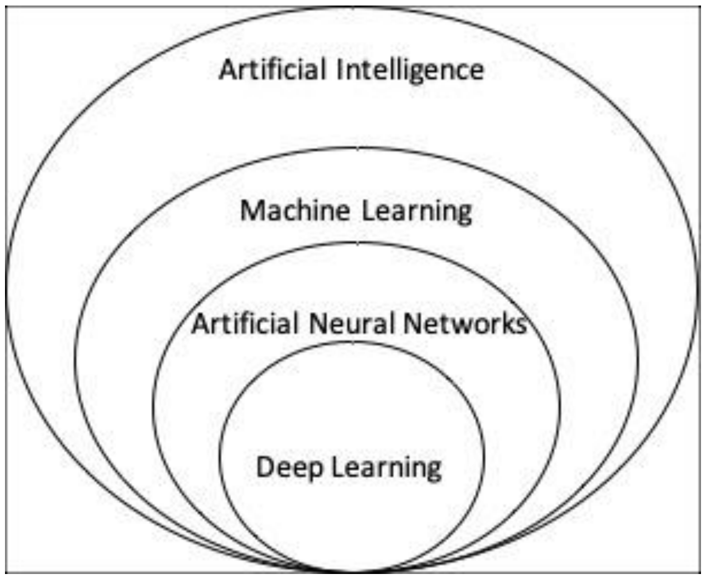


Figure 2. A Venn diagram describing deep learning as a subfield of neural networks which is a subfield of machine learning all included in AI

In the next sections the building blocks of a neural networks and the algorithms used to make the network able to learn would be discussed. Furthermore, techniques to speed up the learning process and enhance the performance of the deep learning model will be reviewed.

ANN Representation

As mentioned previously, ANNs are inspired by the human brain, where our brain contains billions of connected neurons forming a neural network, each neuron receives inputs from other neurons through dendrites, apply some kind of processing on the input, and the calculated result is passed to other neurons connected through axon. ANN has the same structure as the biological structure as shown in figure 3. In ANNs, a neuron is represented by a node associated to an activation function. The nodes are connected via weighted connections, and each node perform multiplication between the connection's weights and input data point, which are the pixels' intensities of training dataset images in the context of image classification. The weights of the connections indicating the extent to which the signal is amplified or diminished. The connections with large, positive weights amplify the signal, meaning that the signal is very important when making a classification. Others have negative weights, dampening the strength of the signal, thus implying that the output of the node is less important in the final classification.

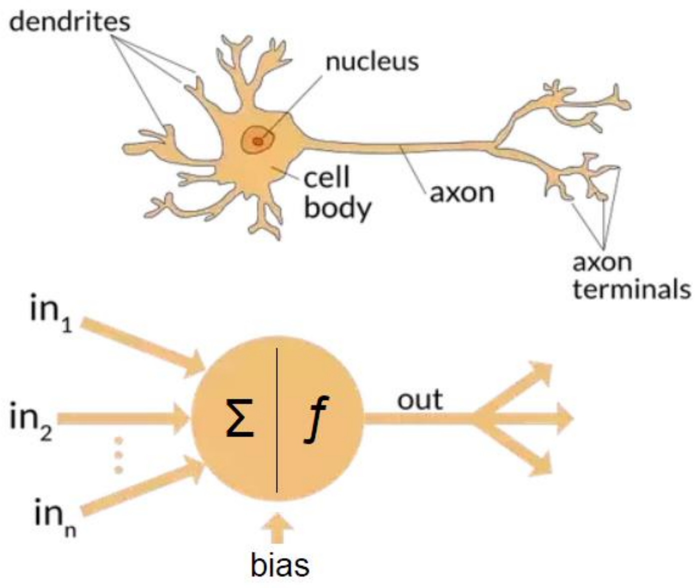
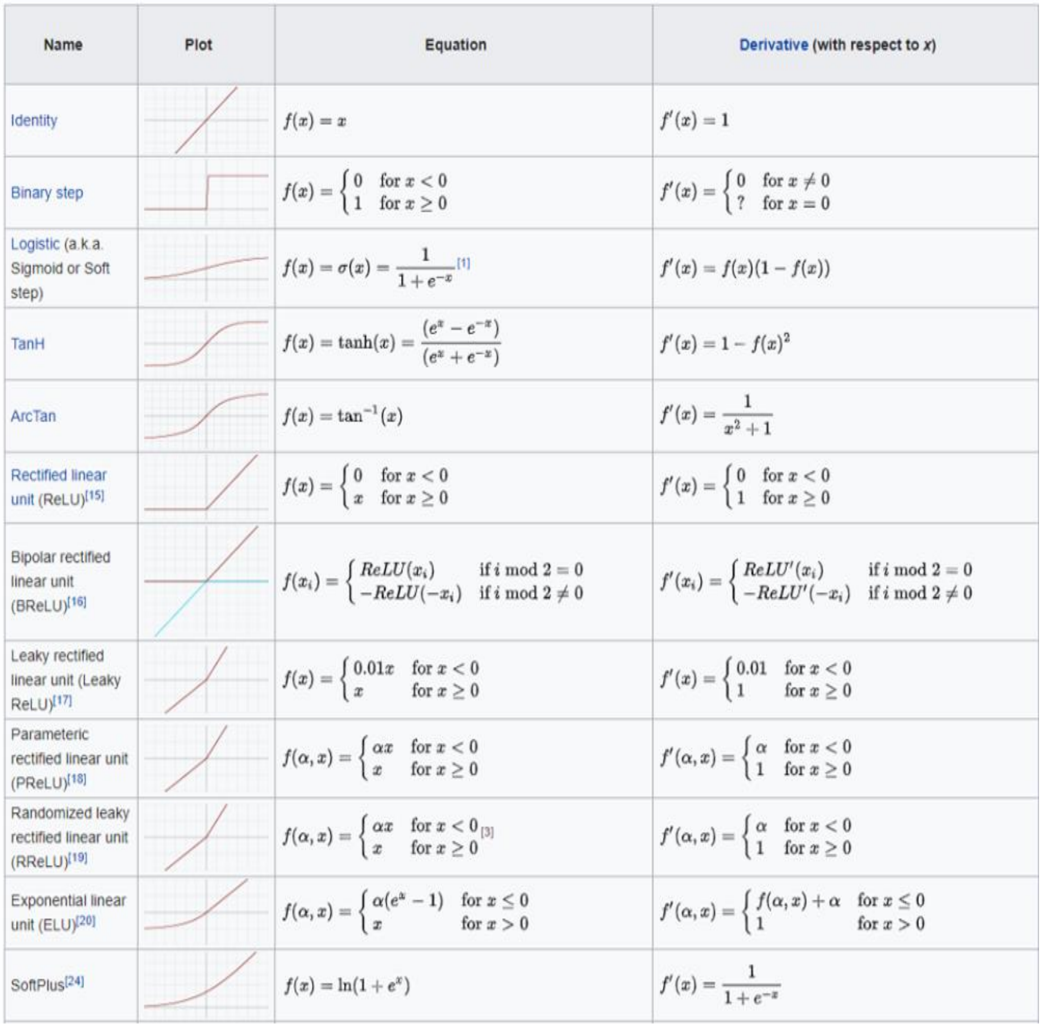


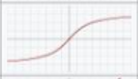
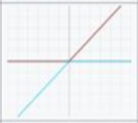
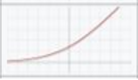
Figure 3. Biological and an artificial neuron [18]

Each neuron has a term called the bias. The bias help determines whether or not a neuron is going to fire by altering the threshold of the activation function in an indirect way. The mechanism helps increase the flexibility of the ANN model to fit the given data. Indeed, the bias vector is often critical for successful learning.

The next step is to pass the summation of input-weight products added to the bias term through the activation function that determines the output will generate based upon its input. The activation function controls whether the neuron would fire or not and to what extent that signal should progress further through the network and affect the classification outcome of the network. The commonly used activation functions are the step, sigmoid, tanh, Rectified Linear Unit (ReLU) and Exponential Linear Unit (ELU) functions. Table 1 shows some famous activation functions along with the plot and mathematical equation of each one.

Table 1. Plots of different activation functions



Name	Plot	Equation	Derivative (with respect to x)
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a. Sigmoid or Soft step)		$f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$[1]	$f'(x) = f(x)(1-f(x))$
TanH		$f(x) = \tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified linear unit (ReLU)[15]		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Bipolar rectified linear unit (BReLU)[16]		$f(x_i) = \begin{cases} ReLU(x_i) & \text{if } i \bmod 2 = 0 \\ -ReLU(-x_i) & \text{if } i \bmod 2 \neq 0 \end{cases}$	$f'(x_i) = \begin{cases} ReLU'(x_i) & \text{if } i \bmod 2 = 0 \\ -ReLU'(-x_i) & \text{if } i \bmod 2 \neq 0 \end{cases}$
Leaky rectified linear unit (Leaky ReLU)[17]		$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0.01 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric rectified linear unit (PReLU)[18]		$f(\alpha, x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Randomized leaky rectified linear unit (RReLU)[19]		$f(\alpha, x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$[3]	$f'(\alpha, x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential linear unit (ELU)[20]		$f(\alpha, x) = \begin{cases} \alpha(e^x - 1) & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} f(\alpha, x) + \alpha & \text{for } x \leq 0 \\ 1 & \text{for } x > 0 \end{cases}$
SoftPlus[24]		$f(x) = \ln(1 + e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

An ANN consists of an input layer, number of hidden layers, and the output layer. The input layer is where the input datapoint is presented to the network as arranged features and the nodes in the input layer do not contain an activation function; however, nodes in both the hidden layers and output layer do contain an activation function. The intermediate hidden layers are located between the input and output layer. The output layer is where the output class label is decided. Figure 4 shows a general representation of neural network architecture.

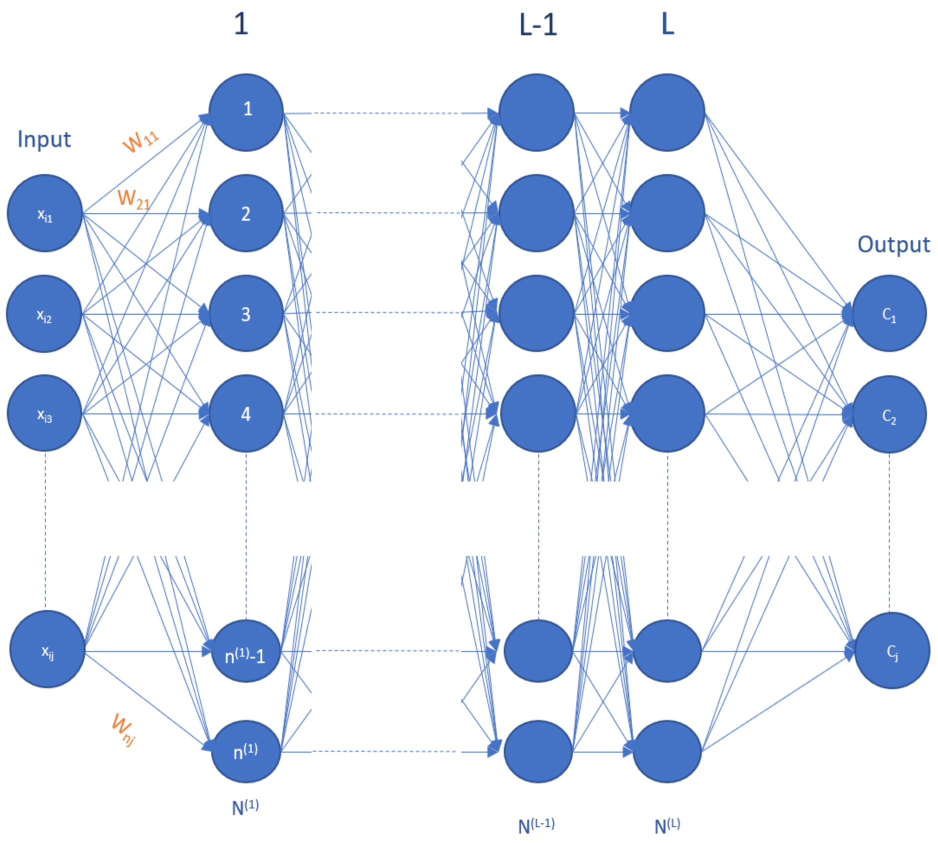


Figure 4. A neural network architecture

For the network above, assume that this neural network model designed to classify an input image into one of *k*-categories. The ith element of the dataset can be represented as a single column vector with shape $[j \times 1]$:

$$\begin{bmatrix} xi1 \\ xi2 \\ \vdots \\ xij \end{bmatrix}$$

The weights of the connection between the input layer and the first hidden layer that consist of *n* nodes can be arranged in a matrix *W* that would then have a shape of [$n{\times}j$] that indicates the number of class labels by the dimensionality of the input images:

$$\begin{bmatrix} W11 & W12 & \cdots & Wn1 \\ W21 & W22 & \ldots & Wn2 \\ \vdots & \vdots & \ddots & \vdots \\ Wn1 & Wn2 & \cdots & Wnj \end{bmatrix}$$

As discussed before, each neuron is associated with a bias-term, which determines how high the weighted sum should be, so the neuron gets meaningfully active. Therefore, the bias for the hidden layer in figure 4 would be a [$n{\times}1$] vector.

$$\begin{bmatrix} \mathrm{b1} \\ b2 \\ \vdots \\ bn \end{bmatrix}$$

Therefore, each neuron in a hidden layer carry out computations that can be summarized in function (1):

$$\mathrm{f}\,(\mathrm{x}i, \mathrm{W}, \mathrm{b}) = \mathrm{W}\,\mathrm{x}i + \mathrm{b} \qquad (1)$$

It is often monotonous to keep track of two separate variables. Hence, the “bias trick” technique is introduced to avoid the monitoring of the separate variables. The idea behind this technique is to embed the bias vector *b* in the weight’s matrix *W*, so they treated as single parameter.

To combine both the bias and weight matrix, an extra dimension (i.e., column) is added to the input data *X* that holds a constant 1. Typically, the new dimension is either appended to each individual vector representing a single datapoint from the dataset as the first dimension or the last dimension. Figure 5 demonstrates the bias trick technique.

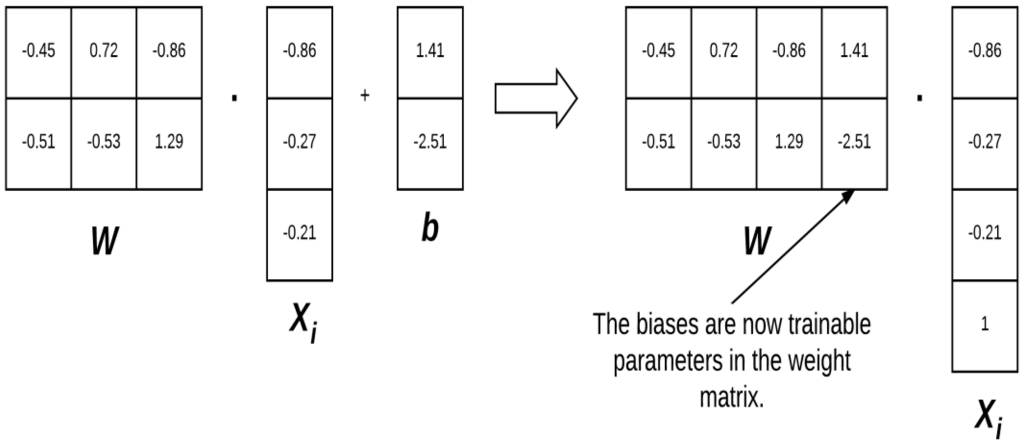


Figure 5. Embedding the bias vector to the weight matrix [20]

Therefore, the calculation carried out in the first hidden layer is nothing but the matrix multiplication between the input matrix and the weights matrix.

$$\mathrm{f}\,(\mathrm{x}_i, \mathrm{W}) = \mathrm{W}\, x_i \qquad (2)$$

ANN Learning and Optimization

ANN algorithms can learn patterns from input data designated for training the model during training phase called a training dataset to produce a parametric model that is capable of classifying an input data point to one of a number of classes. Therefore, the ANN model learns a function that maps the input to the output predictions by defining a set of parameters and optimizing over them. Neural learning refers to the method of updating the weights of the connections between nodes in a network, in a way that increase the strength of connections between nodes that have similar outputs when presented with the same input, which can be accomplished through deep learning algorithms.

❖ **Parametrized Learning**

The cornerstone of today’s machine learning and deep learning algorithms is the parameterized learning concept, which can be defined as “A learning model that summarizes data with a set of parameters of fixed size (independent of the number of training examples) is called a parametric model. No matter how much data you throw at the parametric model, it won’t change its mind about how many parameters it needs.” [21] In the task of machine learning, parameterization involves defining a problem in terms of four key components:

• Data: This component is the input data that the machine learning algorithm should learn from its features and classes. These data include both the data points and their associated class labels. In this work, the data points are raw pixel intensities from the IR images from the burning site, labeled as “people”, “pet” and “no victims”. Typically, the input dataset is denoted in terms of a two-dimensional design matrix, where each column in that matrix correspond to the flattened pixels of an image from the dataset.

• Weights and biases: The weight matrix, typically denoted as W and the bias vector b are called the weights or parameters that define the parametric deep learning model. Indeed, the level of optimization of these parameters control the performance of the classifier.

• Scoring function: This function is responsible for mapping the raw input data to class scores.

F(INPUT_IMAGES) => OUTPUT_CLASS_LABELS

• Loss function: It quantifies the agreement between the predicted scores and the ground truth labels, it is also an important feature of model optimization process, where optimization is the process of finding the set of parameters W that minimize the loss function. However, there are many versions of the loss function, such as hinge loss and cross entropy loss. While hinge loss is quite popular, but in the context of deep learning most often cross-entropy loss, and the Softmax classifiers are adapted. This is due to the Support Vector Machine (SVM) and other classifiers treats the outputs of the scoring function as uncalibrated scores for each class, while the Softmax classifier represent the output as normalized class probabilities, which is more intuitive and easier to interpret.

As mentioned before, the goal of a neural network training is to evaluate the weight and bias matrices such that the model can successfully classify an input datapoint. It is almost impossible to find a perfect set of weights for a neural network because there are too many unknowns. Instead, the problem of learning is cast as a search or optimization problem and an algorithm is used to navigate the space of possible sets of weights the model may use in order to make good or good enough predictions.

❖ **Gradient Descent Optimizer**

Optimization algorithms are responsible for improving the elements of the weight and bias matrices, so the neural network model is finally capable to map a set of inputs to a set of outputs from training data. Typically, the most common algorithm adapted for neural network training is the gradient descent optimization algorithm. The gradient descent optimizer iteratively modifies the values of the weights and biases, so it improves the performance on the training data. The idea behind this algorithm is to iteratively computes the gradient at the current position indicated by the current values of the weigh and bias matrices and performs a parameter update in loop trying to converge towards some local minima of the loss function. Gradient is the direction of the steepest increase in the loss, and the negative of that gradient is the direction of the step that will decrease the loss function most quickly. However, the size of this step or update is controlled via a hyperparameter called the learning rate.

The algorithm for computing gradient efficiently is the backpropagation technique. In order to apply the backpropagation algorithm, the activation function must be differentiable so the partial derivative of the error is computed with respect to a given weight $w_{i,j}$ loss (E), node output o_j, and network output net_j, where partial derivatives of the loss function is calculated with respect to each parameter and store the results in a gradient as follows:

$$\frac{\partial E}{\partial w_{i,j}} = \frac{\partial E}{\partial o_j} \frac{\partial o_j}{\partial net_j} \frac{\partial net_j}{\partial w_{i,j}} \qquad (3)$$

Depending on the number of training examples considered in updating the model parameters in one iteration, there are three types of gradient descents:

1. Batch Gradient Descent: Parameters are updated after computing the gradient of error with respect to the entire training set.
2. Stochastic Gradient Descent: Parameters are updated after computing the gradient of error with respect to a single training example.
3. Mini-Batch Gradient Descent: Parameters are updated after computing the gradient of error with respect to a subset of the training set.

The mini-batch gradient descent makes a compromise between the speedy convergence, and the noise associated with gradient update that makes it a more flexible and robust algorithm. Even if the number of training examples is large, it is processed in batches of *b* training examples in one pass. Thus, it works for larger training examples and with lesser number of iterations or epochs.

Various extensions have been designed for gradient descent algorithm, such as Momentum, and the Nesterov's Acceleration. • Momentum: The goal of adding momentum term to the gradient descent optimizer is to increase the strength of updates for dimensions whose gradients point in the same direction and then decrease the strength of updates for dimensions whose gradients switch directions.

- Nesterov’s Acceleration: This technique ensures that the gradient descent algorithm along with the momentum will not overshoot the local minimum. This is accomplished by slowing down the momentum when approaching to the local minimum.

For this thesis, the Mini-batch Gradient Descent Optimizer is adapted for optimizing the convolutional neural networks implemented in this work through all the experiments. In the next chapter, the convolutional neural networks would be discussed in detail as they are the cornerstone of this research work.

V. CONVOLUTIONAL NEURAL NETWORKS

The Convolutional Neural Network (CNN) is a type of Neural Network (NN) architectures based solely on the convolution principle. The CNNs follow the learning model of the standard NNs as discussed in the previous chapter. NNs are made up of neurons with learnable weights and biases, every neuron receives several inputs, takes a weighted sum over them, pass it through an activation function and responds with an output. The whole network has a loss function and all the tips and techniques that were developed for neural networks still apply on CNNs. Most generally, CNN can be conceptualized as an ANN that has some type of specialization for being able to pick out or detect patterns and make sense of them.

CNNs are powerful image processing, artificial intelligence (AI) that use deep learning to perform both generative and descriptive tasks, often using machine vison that includes image and video recognition, along with recommender systems and natural language processing (NLP).

Image Recognition

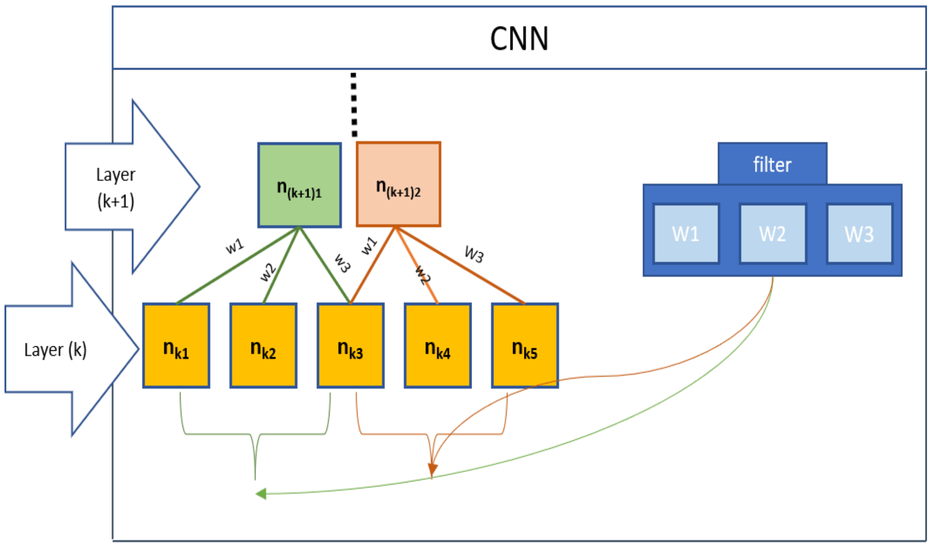
The goal of this work is to develop an approach to detect humans and pets in Infrared (IR) images received from a burning site by recognizing the type of the objects appearing in the input IR image, and classifying it into one of three categories: "*people*", "*pet*", and "*no victims*;" thereby, this task can be casted as object recognition problem. Image recognition according to the visual content is a very challenging task for computers due to large amount of intra-class variability that is most often associated with variations in viewpoint, illumination, occlusion, background clutter, scale variation, and deformation. Moreover, human detection is even more challenging task for computers due to the various poses of the human body.

Although hand-designed heuristics found to be useful to deal with the intra-class variability and high accuracies could be achieved for certain data and tasks. However, the manually designed low-level features for classification are usually limited to the certain task and data type it was developed for and designing effective features for new data and tasks requires new domain knowledge. Most hand-crafted features cannot simply be adapted to new conditions. Better human detection systems can be built by relying more on automatic learning, which is made possible by recent progress in machine learning in general, and deep learning in particular. The goal of applying machine and deep learning algorithms is to discover underlying patterns in the dataset enabling the trained model to correctly classify data points that it has not encountered yet.

In the context of machine learning, it is important to choose the correct approach for tackling the task appropriately. CNNs provide outstanding and state-of-the-art results of the feature-learning approach based on deep learning techniques, and this is the reason behind their popularity in computer vision and recognition system specifically. The success of deep CNN models over Fully-Connected NN models in computer vision tasks can be explained or referred to the properties that exist in the CNNs, but lacks in standard Fully-Connected NNs, such as weight sharing, and translation invariance.

❖ **Weight Sharing**

Standard Fully-Connected NNs are found to be not suitable for image recognition problems, because they have enormous number of parameters, where every neuron in an ANN model is connected to every single neuron in the previous layer. As an example, a monochrome image, with a resolution of 60x60, contains 3,600 pixels. If each pixel intensity of this image is input separately to a Fully-Connected network, then each neuron in the first layer of that network requires 3,600 weights. Thus, the overall number of free parameters in the neural network increase rapidly becomes extremely large as the image size increases. Therefore, these networks lead to slow performance and over-fitting. This situation can be avoided by using CNN models since they are capable of sharing the weights. The weight sharing greatly reduce the number of free parameters in the network leading the CNNs to have fewer connections and parameters making them easier to train. Weight sharing happens across the receptive field of the neurons, filters, in a particular layer. Figure 6 shows rough representation of a CNN, and demonstrate weight sharing principal. It also shows only three weights are required to connect between the two layers in the CNN model. Whereas, in a Fully-Connected NN approach, the number of weighted connections between two layers of the same size would be ten (2x5).



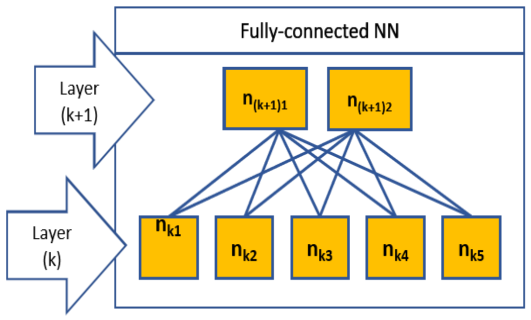


Figure 6. Weight sharing across filters in CNN and comparison between convolutional and Fully-Connected neural networks

❖ Translation Invariance

The weight sharing technique found in CNNs not only reduce the number of parameters in neural models, but also treats the input as a hierarchy of local regions, which adds advantage to the CNN by making it translation invariant. Translation invariance is the ability of the model to preserves the object's identity or category across changes in the specifics of the visual of its input value. Hence, the translation invariance helps the model to perform well in object recognition tasks to distinctive features of an object in the input image. Therefore, it is important to set feature detectors that can detect a particular

Instance of a feature anywhere on the input plane since the precise location of a feature is not relevant to the classification. [22].

In order to achieve translation invariance in a Fully-Connected NN, the model needs to be trained on different images showing the same pattern from different locations of an image, which is not an efficient approach. The detection of a particular feature at any location on the input can be easily done using CNNs due to weight sharing. CNNs leverage the hierarchical relations between the categories to tackle the data sparsity problem. For deep CNNs, each layer of nodes trains on a distinct set of features based on the previous layer’s output. The further you advance into the NN, the more complex the features the nodes can recognize, since they aggregate and recombine features from the previous layer. This is known as feature hierarchy, and it is a hierarchy of increasing complexity and abstraction. It makes deep learning networks capable of handling very large, high-dimensional data sets.

Due to the hierarchical learning fashion in CNNs, basic representation of the problem, such as edges in an image, are detected in the lower level layers of the network; whereas, higher level layers use these basic features to form more abstract concepts.

CNN for people and pet-detection in IR images

By deploying convolutional filters to NN architecture, along with the backpropagation algorithm, the deep learning model is able to learn filters that can detect edges and blob-like structures in lower-level layers of the network. The network then uses the edges and structures as building blocks to recognize line features that characterizes a person or pets in the context of this research.

Victims rescuing and evacuating in a fire situation is challenging because of variety of humans' and pets' poses that the IR camera may capture since the victims in such situation can be standing, sitting, or laying down. This challenge was handled by adapting the CNNs for classifying images transmitted from the burning site; where, as mentioned above, this type of neural networks is characterized with Translation Invariance property which is crucial in image recognition tasks in general and in the problem of this work particularly. Also, the relatively small size of the produced model due to weight sharing technique would ensure fast classification of the IR images, where fire situations are considered time critical.

The principle of Convolution

As mentioned earlier CNN is a type of Neural Network (NN) architectures based solely on the convolution principle. In this section, the convolution operation is discussed.

Convolution is a specialized type of mathematical linear operation combining two signals to form a third signal used for feature extraction, where a small array of numbers, called a kernel, where depending on the element values, a kernel can cause a wide range of effects. The kernel is then applied across the input, which is an array of numbers, called a tensor, then the output is calculated at each location of the tensor and summed to obtain the output value in the corresponding position of the output tensor known as a feature map. The general expression of a convolution is:

$$g(x,y) = \omega * f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} \omega(s,t) f(x-s, y-t) \qquad (4)$$

Where $g(x,y)$ is the filtered image, $f(x,y)$ is the original image, and ω is the filter kernel. Each element of the filter kernel is considered by $a \leq s \leq a$, and $-b \leq t \leq b$. It is one of the most critical and fundamental building-blocks in computer vision, image filtering, and processing to implement operators whose output pixel values are simple linear combinations of certain input pixel values, such as sharpening, edge detection, or gaussian blur.

Building blocks of CNN

A CNN is composed of stacking multiple types of layers in a specific manner. The most common layers of a CNN are convolutional layer, pooling layer, fully-connected layer, and batch normalization layer.

These layers are arranged in a 3D volume in width, height, and depth where the input layer depth refers to the number of channels. The hidden layers it refers to the number of filters in the previous convolutional layer. The convolutional, fully-connected, and batch normalization layers contain parameters that are learned during the training process.

• Convolution layer

A convolution layer is the core component of the CNN architecture. It consists of a combination of convolution operation and a nonlinear activation function. The main task of the convolutional layer is to detect local conjunctions of features from the previous layer and mapping their appearance to a feature map.

The convolutional layers are made up of number of independent neurons that have learnable weights and biases. These neurons are called kernels, and they are the parameters that would be learned by the network subsequently. Each kernel is independently convolved with the image producing a feature map. Figure 7 demonstrates the convolution operation, where a kernel is applied across the input tensor, and an element-wise product between each element of the kernel and the input tensor is calculated at each location and summed to obtain the output value in the corresponding position of the output tensor called a feature map.

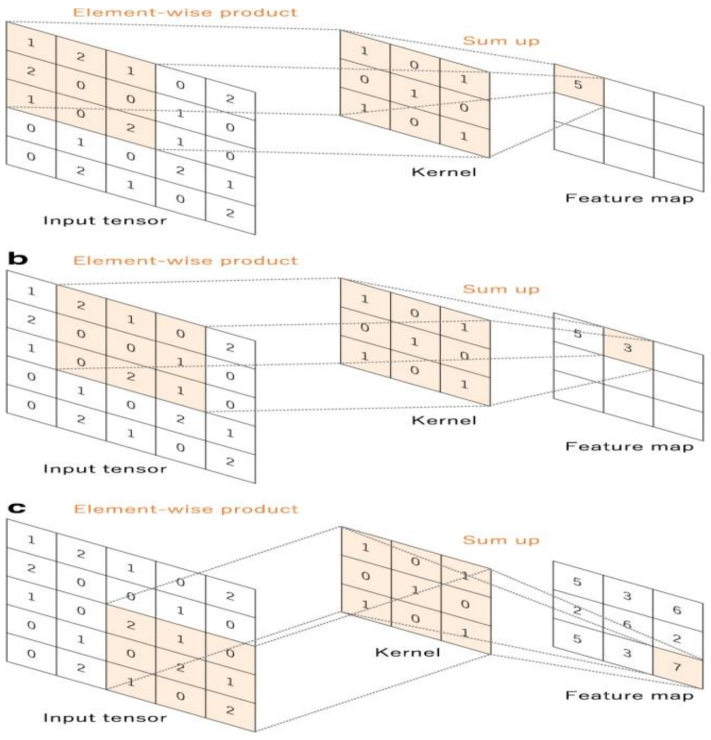


Figure 7. An example of convolution operation with a kernel size of 3 × 3 [23]

The convolution operation is performed by sliding the filter over the input. However, the parameter that controls how much the filter moves at each step is set with the stride size. The dimensions of the resulting feature map depend on the input image size, stride size, and whether or not the input is padded. Padding is commonly used in CNN to preserve the size of the feature maps; otherwise, they would shrink at each layer.

In the convolutional layer, multiple convolutions are performed on an input each using a different filter and resulting in a distinct feature map. These feature maps are stacked together and become the final output of the convolution layer. Figure 8 shows what happens in a convolutional layer where each filter convolves over the input volume producing stack of feature maps.

The convolution operation is performed by sliding the filter over the input. However, the parameter that controls how much the filter moves at each step is set with the stride size. The dimensions of the resulting feature map depend on the input image size, stride size, and whether or not the input is padded. Padding is commonly used in CNN to preserve the size of the feature maps; otherwise, they would shrink at each layer.

In the convolutional layer, multiple convolutions are performed on an input each using a different filter and resulting in a distinct feature map. These feature maps are stacked together and become the final output of the convolution layer. Figure 8 shows what happens in a convolutional layer where each filter convolves over the input volume producing stack of feature maps.

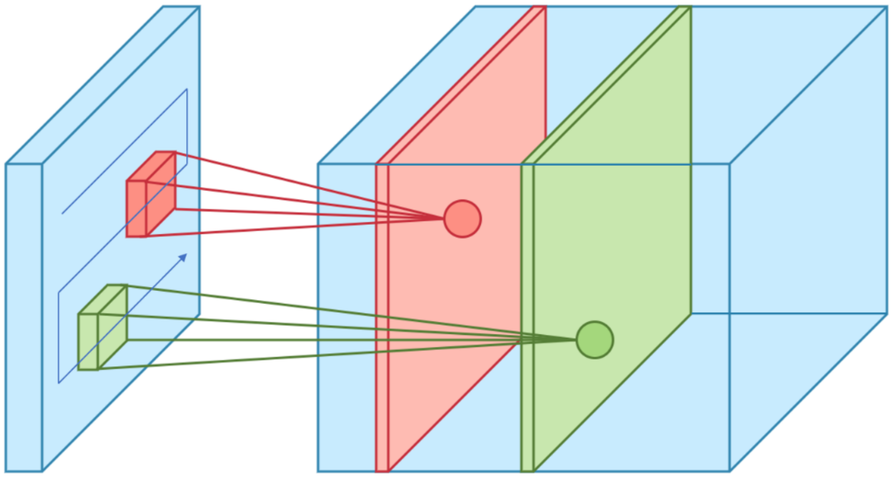


Figure 8. Convolution process and stacking feature maps

As mentioned in the previous chapter, every NN architecture utilizes a non-linear activation function to determine whether a neuron would fire or not and applied in the CNNs. Each convolutional layer is associated with an activation function. The activation function is applied to the resulting feature ap in an element-wise manner. The role of activation function in the context of CNNs is to activate the specific type of feature at a given spatial location in the input volume is found. In lower layers of the network, filters may activate when detecting edge-like or corner-like regions. For the deeper layers of the network, filters may activate in the presence of high-level features, such as parts of the face, the paw of a dog, or the hood of a car [20].

The ReLU and its variant are the most popular activation function used in CNNs. They are shown in figure 9 below.

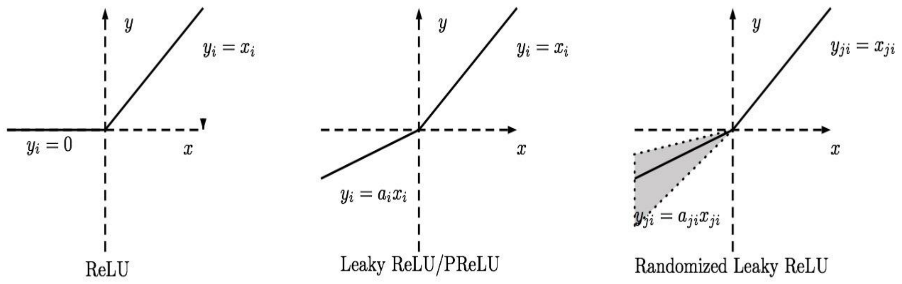


Figure 9. ReLU and its variant [24]

• Pooling Layer

Pooling layers are primarily used to reduce the spatial size of an input volume and are placed either in-between consecutive convolutional layers as in LeNet [25] architecture or between convolutional, and Fully-Connected layer as in a VGGNet [26] architecture. Pooling reduces the number of parameters, computation in the network, and helps to control overfitting. Pooling layers operate on each of the depth slices of an input independently using either the max or average function. Max Pooling is typically done in the middle of the CNN architecture to reduce spatial size, whereas average Pooling is normally used as the final layer of the network. A pooling operation is shown in figure 10. As seen from the figure, a decrease in input size depends on the size of pooling layer and stride.

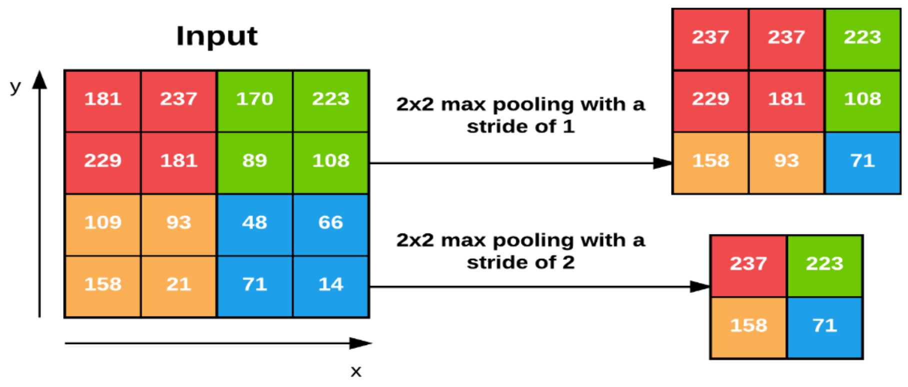


Figure 10. An example of a pooling operation with different strides [27]

• Fully-Connected layer

As the name suggests, in a Fully-Connected layer, neurons are Fully-Connected to all activations in the previous layer. Fully-Connected layers are always placed at the end of the network to perform classification based on extracted features from the preceding convolutional and pooling layers. This layer basically takes an input volume from either the conv, ReLU, or pool layer, and outputs an *N* dimensional vector; where *N* is the number of classes that the program must realize.

• Batch Normalization layer

Batch Normalization layers [28] are used to normalize the activations of a given input volume before passing it into the next layer in the network. If x is considered as the mini-batch of activations, then the normalized $\hat{\boldsymbol{x}}$ can be computed using equation (5).

$$\hat{\boldsymbol{x}} = \frac{x_i - \mu_\beta}{\sqrt{\sigma_\beta^2 + \varepsilon}} \tag{5}$$

Here, μ_β and σ_β^2 are respectively mean and variance over each mini-batch of the training images, β. The error ε is set equal to a small positive value in the range of 10^{-7} to avoid dividing by zero. Applying (5) implies that the activations leaving a Batch Normalization layer will have approximately zero mean and unit variance, i.e., zero-centered. At testing time, the mini-batch μ_β and σ_β^2 are replaced with running averages of μ_β and σ_β^2 computed during the training process. This ensures that images through the network can be passed and obtain an accurate prediction without being biased by the μ_β and σ_β^2 from the final mini-batch passed through the network during the training phase [20].

Batch Normalization has been shown to be useful for the following effects in deep NNs.

- ✓ It reduces the number of epochs a NN takes to train itself.
- ✓ It stabilizes training by allowing for a wide variety of learning rate and regularization techniques.
- ✓ It provides stable loss curve as it helps to minimize loss.

• Dropout layer

Dropout is actually a form of regularization that aims to help prevent overfitting by increasing the testing accuracy, and possibly at the expense of training accuracy. For each mini-batch in the training set with probability *p*, the Dropout layer randomly disconnect inputs from the preceding layer to the next layer in the network architecture. Figure 15 shows dropout operation with a value of 0.5, where 50% connections from previous layer are dropped. The Dropout method helps the model to generalize by reducing overfitting by dropping some connections from the previous layer ensures there are multiple nodes instead of one when presented with a given pattern, i.e. filter.

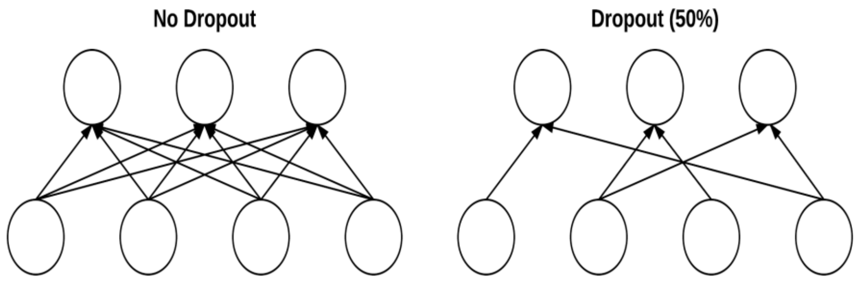


Figure 11. Effect of no Dropout (left) and Dropout with a value of 0.5 (right)

In this chapter, the role of a CNN in image recognition is discussed, and why it is the best path for image recognition tasks. Finally, the building blocks of a CNN are reviewed.

VI. DATASET

In this thesis, the problem of finding victims detection models with IR images is treated as a supervised learning problem in that the CNN model is trained with examples or data so it can learn to detect the target objects when unobserved data is introduced to the trained model. Most of the current work on object detection uses one of the handful of widespread image datasets [30] that are available today. Hence, the literature available is limited by the subjects of the existing dataset from open sources. Moreover, most of the available thermal image datasets would not be useful for this research work due to the angle where these images are taken from cameras hanged beside the traffic lights without being directed toward people detection in an emergency. These data sets include FLIR Thermal Dataset [34], and Pedestrian Detection dataset from Elektra [35]. This work is only concerned with detecting people and pets in IR images transmitted from a burning site. However, no dataset from open sources is meeting the requirements of this work; thus, the need to develop a dataset for the training and testing of the proposed deep learning models.

Dataset Specifications

The requirement of the proposed model, the dataset must consist of images that closely relate the angle and resolution that will be produced from the thermal camera mounted in the AESV embedded solution. Therefore, thermal images that contain shapes of people and pets in different poses along with presence with other objects are needed for the training phase of the model. The challenges associated with various poses of the human body and different sizes due to various distances from which the image is captured will be handled by using datasets with well-defined specifications and categories. For training and evaluating the deep learning model, a dataset consisting of images of people and pets would be captured using IR Thermal Imager.

The dataset has many defined categories in an attempt to cover all the possible situations and cases that the CNN may face while classifying the images sent from the burning site. The categories include images of people in various poses: lying down, sitting, and standing from different distances and different angles, where some images would show the whole human body, and others would show parts of the human body. Also, there would be images of pets from different distances and different angles. Finally, images containing other spaces and rooms with no human or pets would be gathered.

Dataset Acquisition

As mentioned before, this work is targeting the scenarios of burning structures, hence situations involving either high temperature, dense smoke, or both together are expected .The dataset was gathered using Perfect-Prime IR0002, IR Thermal Imager, with a resolution of 3,600 pixels and temperature range from (-4 ~ 572°F).Most of the infrared images were taken at room temperature. However, to mimic the expected high temperatures, a number of images were captured in front of, and beside heat sources, such as a hot stove or barbeque fire, furthermore, the images were processed to better represent the very high temperatures situations.

The total number of dataset images is 2,428 infrared images; these images were captured in different rooms layouts, indoor and outdoor, during day, and night.

Furthermore, the images were taken from different angles and distances from the targets, and some images were captured from near distances; less than 3 feet from the target object, others from far distances; up to 20 feet away from the object. The images captured covered the three classes which the CNN should distinguish "*people*", "*pets*", and "*no victims*."

• "People" Class

The people category includes images of babies, toddlers and adults, male and females, in different poses; lying down, sitting and standing. Furthermore, the person's appearances in the images vary; some images are showing the whole person's body, whereas others are showing parts of the body, such as head, legs, or arms. This category consists of 1,712 infrared images. Some images include only one person, whereas some include a group of people. Figure 12 shows image samples of the "*people*" class category.

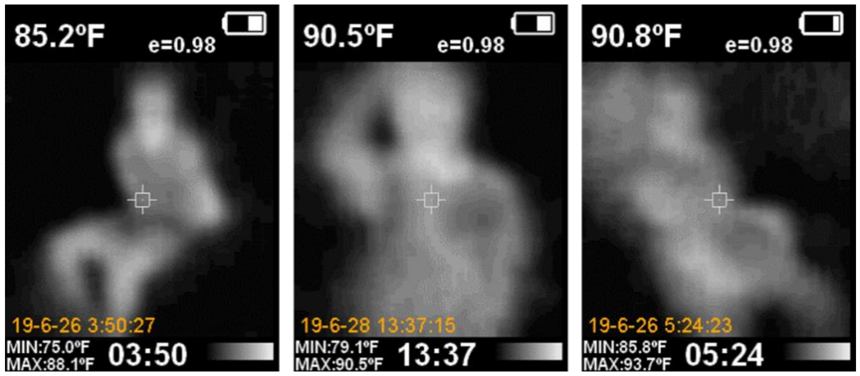


Figure 12. Sample of "people category" (a) sitting down person. (b) upper part of standing person. (c) laying down person

• “Pet” Class

This category consists of 275 thermal images of cats and dogs. The images show the pet sitting, lying down, and walking. Figure 13 shows a sample of the “*pet*” class category.

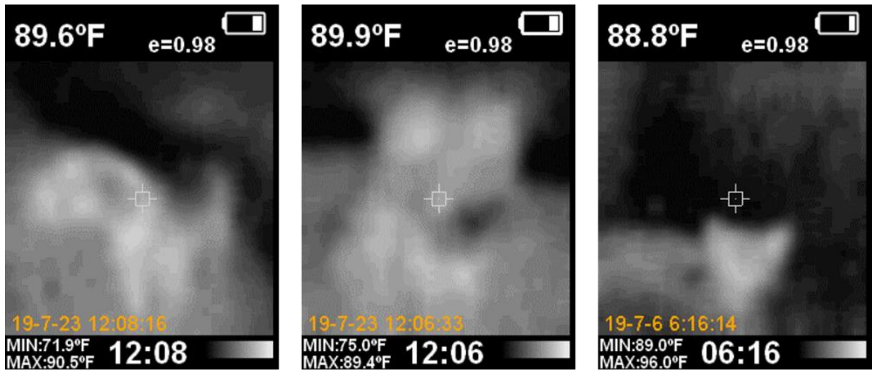


Figure 13. Sample of pet category. (a) standing dog. (b) upper part of a dog. (c) sitting cat

• “No Victims” Class

This category has 440 infrared images. It is intended to include everything other than people and pets to introduce a diversity of objects and space. In order to build this category, images of objects emitting heat such as barbeque fire, lamps, and hot water tap, were taken along with images including no heat sources, which is a possible situation also. Figure 14 shows IR image examples from the “*no victims*” class category.

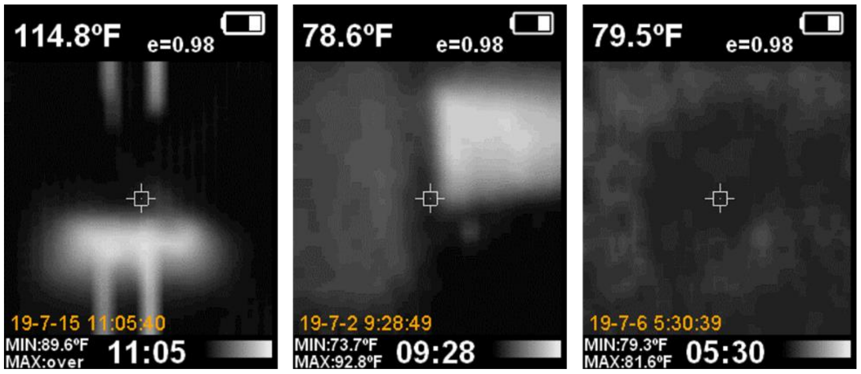


Figure 14. Sample of "no victims" category. (a) barbeque fire. (b) television. (c) space with no heat sources

Dataset Processing

➢ Color Inversion

The infrared images collected for this work were taken in temperatures ranging between 70°F and 110°F that are not representative of the temperatures in burning sites since capturing IR images from real-life fire incident was not possible. However, the gathered images still useful since there may exist locations in the burning building with temperatures in the same range to that the dataset images were captured in, but the smoke leaked into them.

In an attempt to qualify the dataset to represent all possible scenarios in burning site, including the extremely high temperature scenarios above 300°F, a method was researches to process the IR images, so they resemble IR images captured in high-temperature environment. The insight of how this type of images would look, is developed by capturing images of humans in front of hot stove, as seen in figure 15, the human body in such case appears in darker color than the hotter background that is appearing in a bright almost white color.

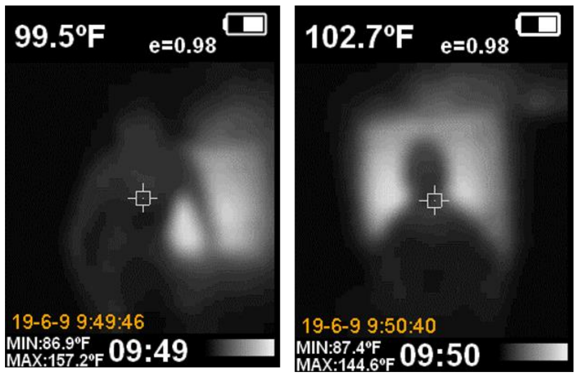


Figure 15. Sample of IR images captured in front of hot stove

The process that found suitable for this task is color inversion. Meaning, if the CNN model developed was successfully trained to detect victims in both the original IR images and the inverted version, then it would be able to classify images captured in burning structure, either if the temperatures are very high, or normal, with or without smoke existence. Figure 5 represent sample of the inverted IR images.

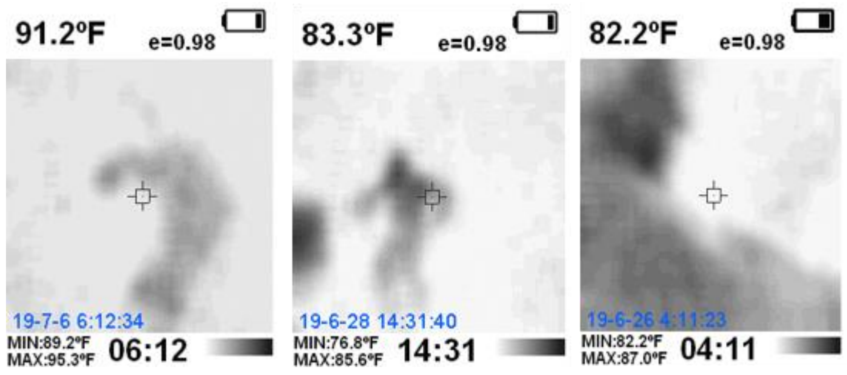


Figure 16. Sample of the inverted IR images (a) inverted IR image of a cat. (b) inverted IR image of a person. (c) inverted IR image of a person

- Oversampling

The total number of IR images in the dataset after using color inversion is shown in the next figure 17. It can be noticed in the figure that the dataset is imbalanced, which is when each of dataset classes have a different number of examples. In this dataset, the number of images in “*people*” category is larger than the number of images in the remaining two classes “*pet*” and “*no victims*”.

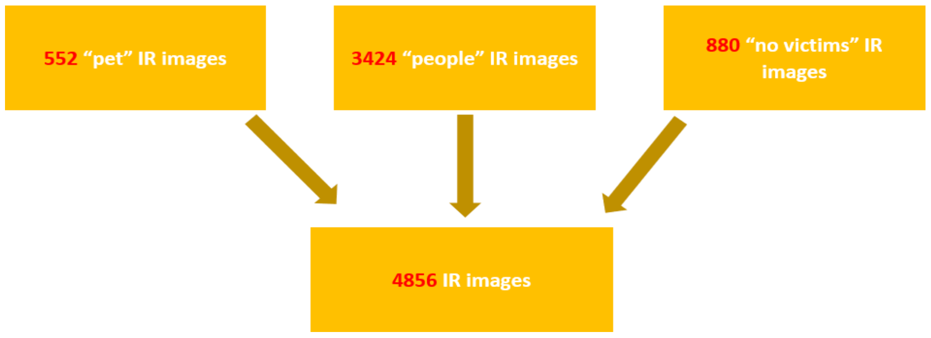


Figure 17. IR Dataset categories counting after color inversion

Imbalanced data poses added difficulty, as most learning algorithms will exhibit bias towards the majority class, and in extreme cases, may lead to total ignorance of the minority classes altogether. Indeed, class imbalance has been studied thoroughly over the last two decades, since most of the real-world classification problems display some level of class imbalance. Traditional methods to handle class imbalance in machine learning are over and under sampling. Under-sampling voluntarily discards data, reducing the total amount of information the model has to learn from, hence may defect the model’s performance [31]. Whereas over-sampling technique aims to increase the size of the minority classes in the dataset through instantiating new versions of the original available information in that class in order to have the same number of examples as the majority class. Figure 18 illustrates the idea of under and over sampling.

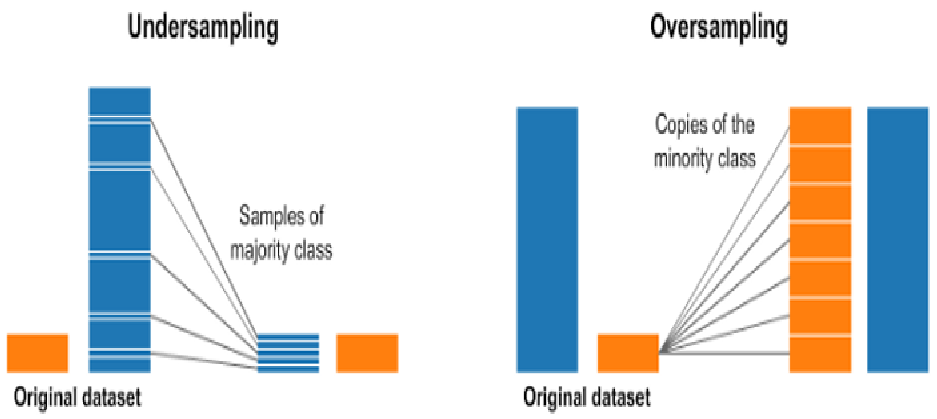


Figure 18. Over and under sampling [32]

Using data augmentation to increase the dataset size was adapted by Krizhevsky et al in [33] in their work. In this work, over sampling of the "*pet*" and "*no victims*" categories were adapted. The number of images in these classes was increased through flipping and rotating the original IR images in different angles. Therefore, applying data augmentation to the IR dataset has produced a balanced dataset consisting of about 10,200 IR images in total.

Dataset Partitioning

When working on a deep learning model, the first step is to train it with the train split data from the overall dataset. Afterwards, the model must be tested on a different split data from the overall dataset that the model did not use during the training phase. The training and testing dataset were developed by splitting the dataset that was developed in a random manner into two sets. The random separation is needed to be done before start training the model. The test set is created by randomly picking images from the processed dataset such that the test set contains 90% of the total number of the final dataset. The final partitioning is presented in Table 2.

Table 2. Balanced dataset partitioning

	People	Pet	No victims
train	3,082	3,080	3,081
test	342	342	342

VII. METHODOLOGY

This chapter is addressing the different phases to develop deep learning models for detecting victims in IR images. In the course of describing the CNN model design number of consecutive steps that need to be demonstrated, starting from deciding on the CNN architecture and layers configuration, then discussing the optimization algorithm adapted to optimize the deep learning models, along with the hyperparameters tuning process associated with each step. Next, the problem of overfitting and the potential approaches explored in this work to handle this phenomenon were introduced. This chapter closes with a mathematical explanation of how each layer of the model works in an attempt to develop better understand the methodology of each layer of CNN.

Dataset Preprocessing

The aim of preprocessing dataset images as a preparing step before they being fed to the deep learning model either for training or testing, is to improve the image data by suppressing unwanted distortions or enhancing some image features in a way that can improve the performance of the proposed classification model.

• **Image Cropping**

The IR images gathered for this work to build the dataset need to be cropped to remove the parts of the image where data such as the time and temperature are displayed on the upper and lower parts of the image as shown in figure 19 (a). Therefore, to ensure that the deep learning model would not get confused with these data by considering them as deterministic features of the objects of interest, the dataset images were cropped using slicing function from the NumPy Python library. Figure 19 (b) shows the IR image after cropping the unwanted part of it.

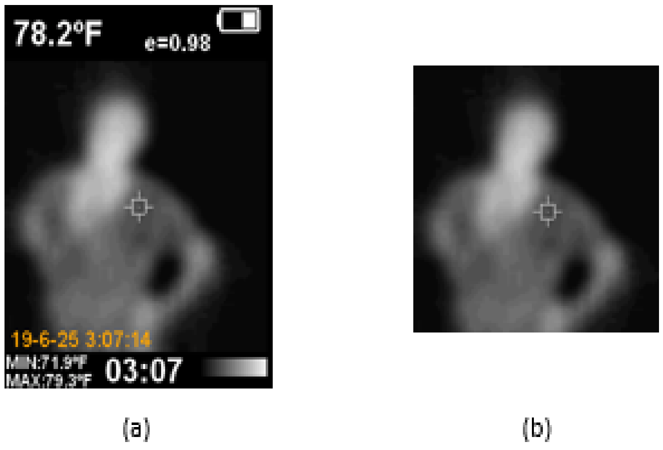


Figure 19. Image cropping

• **Histogram Equalization (HE) and Contrast-Limited Adaptive Histogram Equalization (CLAHE)**

In this work the dataset images were defected with the low contrast between the humans and the background especially in the cases of people in motion or when the background and humans' temperatures are similar. Indeed, low contrast is a common drawback of IR images, thus, contrast enhancement is one of the essential preprocessing methods in infrared imaging systems to improve the visual quality of infrared images. Histogram Equalization (HE) found to be a potential solution for the low contrast images in this work. The HE technique used to enhance dynamic range in an image by having pixels' intensities occupy a wider range of the grayscale intensity levels and have the appearance of high contrast image with a large variety of gray tones. Figure 20 on the top shows a low-contrast image with its histogram, whereas on the bottom the resultant image and its histogram after applying histogram equalization is shown.

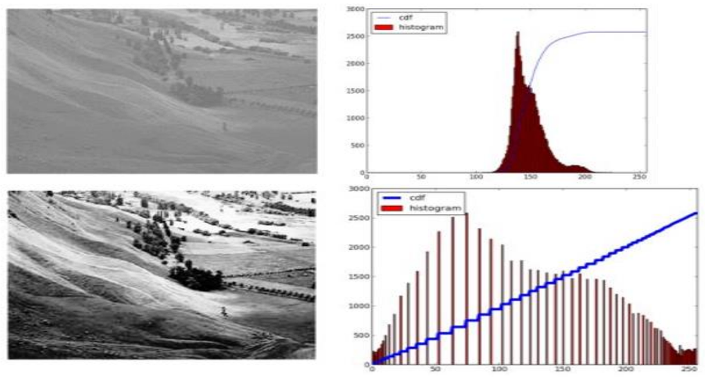


Figure 20. Comparison between an image before and after applying HE [45]

However, after applying Histogram Equalization (HE) function available in OpenCV Python library, the results was not satisfying, where the classification accuracies were deteriorated with about 10% due to the high brightness of the processed images, which in result led to highlighting unimportant features in the IR images as shown in figure21. This problem has evolved since the brightness of homogeneous areas in images are prone to saturation when using Histogram Equalization (HE) technique because its histogram is not confined to a particular region.

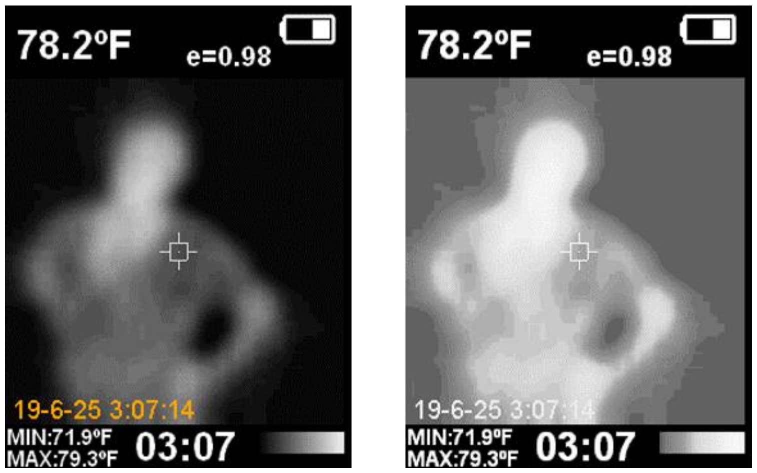


Figure 21. On the left the original IR image, and on the right the image after applying HE

In order to avoid these effects another approach had to be researched for enhancing the contrast and maintaining low brightness level of the resulting images. After investigation Contrast limited Adaptive Histogram Equalization (CLAHE) found to be the proper choice.

The main difference between ordinary HE and CLAHE is the way histogram is computed for the input image, while the ordinary HE technique proposes computing the histogram for the entire image at once. Performing CLAHE, the image is diced into number of regions or sections and then several histograms are computed, each corresponding to a distinct region of the image, and uses them to redistribute the lightness values of the image. This approach is suitable for improving the local contrast and enhancing the definitions of edges in each region of an image. In the CLAHE function available in OpenCV library, the image is divided into small blocks or windows called "tiles", then the histogram is built and equalized for each of these blocks rather than the entire image as in HE. Figure 22 Shows an example of a sample of the gathered IR dataset before and after applying CLAHE.

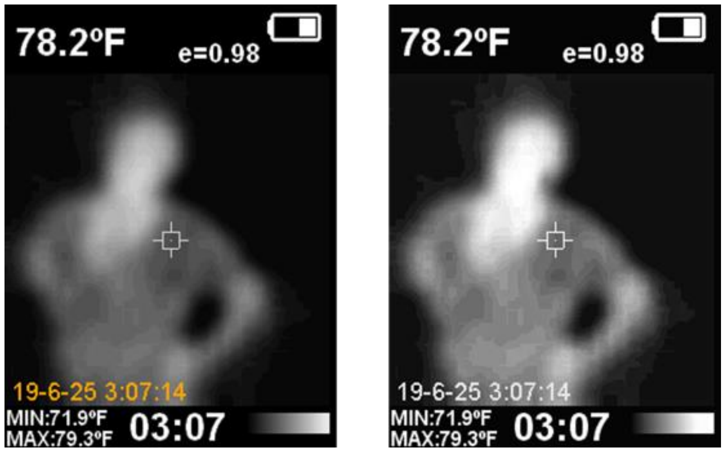


Figure 22. On the left the original IR image, and on the right the image after applying CLAHE

However, this process is considered computationally expensive, thus, the decision to adapting CLAHE is a tradeoff between performance and cost. The performance of multi CNN models that were tested with and without using CLAHE. The results show that no added value attained from using this preprocessing technique, where the accuracies either were not affected or in some cases the performance was getting impaired with a drop-in accuracy between 2% and 3%. Therefore, there is no benefit in using any of the contrast enhancement mechanisms, where these results proved that convolutional neural networks are powerful enough to learn from and classify poor-contrast IR images. However, more detailed results along with figures would be reviewed in the next chapter.

Defining CNN Model Architecture

Defining the architecture of the CNN model is about defining the numbers, types and arrangement of the different types of layers in the model along with the values of any variables associated with these layers, such as the size of filters in the convolutional and max pooling layers, or the number of nodes in the fully-connected layer.

The process of developing the proper architecture to accomplish a certain job requires conducting many experiments since the architecture is dependent on the characteristics and size of the dataset used to train the model. Therefore, there are no formulas or certain rules to follow while designing a CNN model.

The procedure to determine the architecture of the proposed model involved trying out a number of the famous models, such as LeNet [25] and VGGNet [26]. The models that behave in a promising manner was adapted as a source of inspiration for the final CNN architecture. The next step is defining the number of layers in the model. The trade-off of adding more hidden layers is that it is computationally expensive to train the network. Thus, the experiments were initiated with shallow models then extending to deeper ones as the performance keeps getting enhanced. However, after a certain number of layers, either no significant boost in accuracies is gained or in some cases, the performance of the model was deteriorating and no more layers were added. Nevertheless, the model's performance can still improve via tuning more hyperparameters of the CNN model, such as the learning rate, batch size, and kernels' sizes, optimization and regularization techniques.

As previously mentioned, a deep learning model was implemented and benchmarked. This model takes an input image of 60×60 pixels and processes it through several convolution, max-pooling, and fully-connected layers before providing the final output of either three classes: “*people*”, “*pet*” or “*no victims*.” Finally, a Softmax Classifier is used to classify the image into distinct categories.

Many architectures inspired by different popular CNN models were tested before deciding on the final CNN model architecture in this work. The model is trained and validated using the balanced three-classes IR dataset consisting 10,200 IR images. The high-level architecture of the one-step model is shown in figure 23.

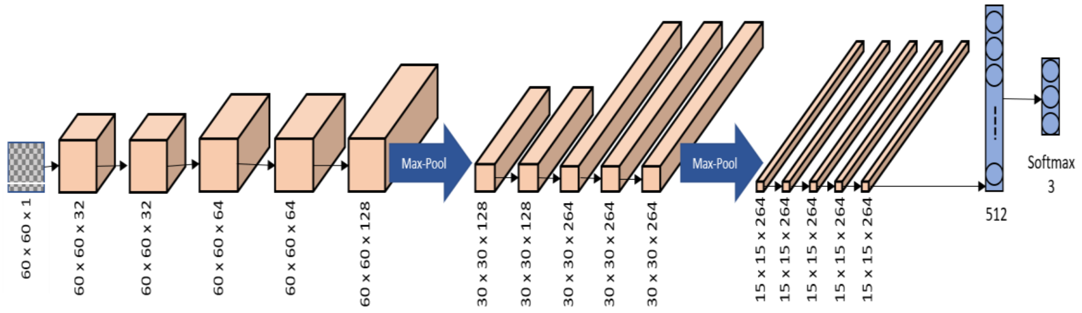


Figure 23. Architecture of the implemented one-step

Hyperparameters are parameters the neural network can’t learn itself via the optimization algorithm such as gradient descent. These include learning rate, number of layers, number of neurons in a given layer, and many other parameters. However, this section is aimed to cover the hyperparameters that are related to the CNN model layers. Each layer type has its hyperparameters to be tuned. The next section is dedicated to reviewing the hyperparameters choices associated with the different types of layers in the implemented deep learning model.

a) Convolutional Layer Hyperparameters

- Filter size: Depending on the application the filters sizes usually used are 3x3, 5x5 or 7x7 filters. In this work, all the filters sizes are 3x3, filters are 3D and have a depth dimension as well, but since the depth of a filter at a given layer is equal to the depth of its input, it is not explicitly expressed.

- Filter count: This is the most variable parameter. Using more filters results in a more powerful model, but the risk overfitting is presented due to the increased parameter count. Usually, the initial step is to start with a small number of filters at the initial layers and progressively increase the count as images go deeper into the network.
- Stride: Stride is the number of pixels shifts over the input IR image pixels. It was kept at the default value 1.
- Padding: Sometimes kernel does not perfectly fit the input of the convolutional layer, to handle this situation there are two options; either zero-padding that pads the input tensor with zeros so that it fits, or valid-padding by dropping the part of the image where the filter does not fit. In this work, zero-padding was adapted.
- Activation Function: Each convolutional layer is associated with an activation function, several choices for the activation function were tested; however, the ReLU function was found to produce the best performance.

b) Max-Pooling Layer Hyperparameters

- Filter size: The size of the pooling operation or filter is smaller than the size of the feature map; specifically, it is almost always 2×2 pixels. This configuration was adapted for this work.

- Stride: Stride is set to 2 in all the Max-Pooling layers in this work.

c) Dropout Layer Hyperparameters

- Dropout probability: Dropout probability is the probability at which outputs of the layer are dropped out. The dropout probability throughout the convolutional neural network developed in this work is 25%.

So far, the first phase of developing a CNN model skeleton was discussed. In the next section, the optimization process of the deep learning model would be demonstrated along with the associated hyperparameters

Optimizing the CNN Model

Optimization methods aim to obtain the optimal Weight Matrix W and Bias b in the deep learning model. For optimizing the neural network in this research work, mini-batch gradient descent is adapted. Three hyperparameters noticed to have a strong influence on the performance of the learning algorithm are the batch size, epochs number, and the learning rate (lr).

a) Batch Size: The batch size is a hyperparameter that defines the number of samples to work through before updating the internal model parameters. Popular batch sizes include 32, 64, and 128 samples. The ideal batch size for each model developed in the course of this work was defined through trial and error.

b) Number of Epochs: The number of epochs is the number of times the entire training set pass through the convolutional neural network. It is common to watch both the training and validation accuracies during the model's training and refer to the learning curves. These plots can help to diagnose whether the model has over-learned, under-learned, or is suitably fit to the training dataset. The deep learning model should be trained for the adequate number of epochs allowing the learning algorithm to run until the error from the model has been sufficiently minimized. Throughout the experiments carried out the number of epochs set to100, 200, 300, 400, and 500. However, the final model required 200 epochs.

c) Learning Rate: The learning rate hyperparameter controls the rate or speed at which the model learns, through controlling the amount of change applied to the weights matrix. Generally, it is not possible to calculate the best learning rate for particular model and dataset. Instead, a good learning rate must be discovered via trial and error. However, the value of 0.001 was found to be a good starting point to initiate the experiments after trying out other rates such as 0.1, 0.01, and 0.0001. Defining one fixed value for the learning rate is a sensitive task, if the learning rate too small the neural network may not learn at all, and if it is too large then low loss area on loss curve maybe overshoot. Thus, using a learning rate schedule proved to be a good practice when training neural networks, where instead of choosing a fixed learning rate hyperparameter, the configuration challenge is now involving choosing the initial learning rate and a learning rate schedule. In this work, the standard learning rate schedulers available in the Keras library are used. This scheduler decreases the learning rate gradually after every batch update, and it is controlled by applying the following formula for adjusting the learning rate after every batch update:

$$lr = initia\_lr \times (1/(1 + decay \ \times iteration)) \tag{6}$$

Here, the term *lr* refers to the learning rate and initial learning rate which needs to be tuned. Iterations can be calculated as the total number of training images in the dataset divided by the batch size. The decay is usually set to the value of the initial learning rate divided by the number of epochs, and this is the case in the implemented CNN model for this work.

The training loss curves are very helpful and informative for tuning the learning rate, where the shape of the training loss curve provides signs about how good the learning rate is and if it should be increased or decreased to enhance the model's performance. Figure 24 demonstrates the influence of the learning rate on the CNN model's loss.

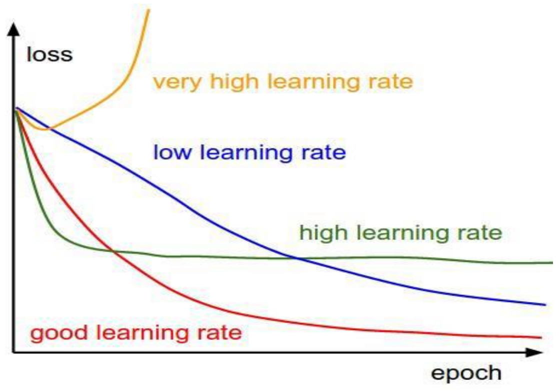


Figure 24. Learning rate influence on training loss curves [40]

For the model to learn faster and reduce the training time, Momentum and Nesterov Acceleration were used, which are extensions to the gradient descent algorithm that were discussed earlier in the Deep Learning chapter. Momentum hyperparameter controls the acceleration of gradient descent in the relevant direction, and it is set to a value greater than zero and less than one, where common values such as 0.9 and 0.99 are used in practice [41]. In this work, the SGD optimizer was initialized with a momentum value of 0.9.

Handling Overfitting

One of the major problems that a CNN model may suffer from is overfitting, which happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data, so it is unable to generalize to other data. This phenomenon causes the deep learning model to perform better on the training dataset than the test set. This can be diagnosed via the loss curves of testing and validation, wherein such case the training loss curve would be descending and reaching very low values that the training accuracies attained are high and satisfying. However, the validation loss curve would show that the trained model fails to classify the inputs of the validation dataset. After some number of epochs neither the validation accuracy getting improved nor the validation loss getting minimized to acceptable values, since it lacks the ability to generalize to other data than the training dataset. The loss curve in case of overfitting would look like the curves in figure 25.

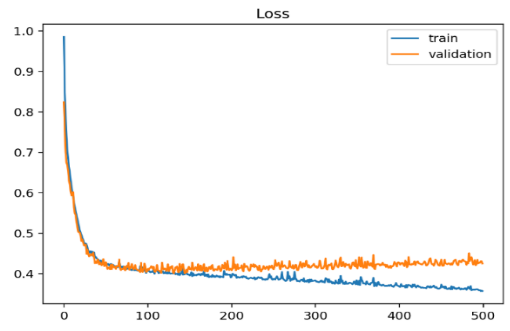


Figure 25. Example of Train and Validation Learning Curves Showing an Overfit Model [39]

This often occurs if the model has more capacity than is required for the problem, and, in turn, too much flexibility. It can also occur if the model is trained for too long.

The practice adopted in developing the CNN models helped mitigating overfitting to some extent, where the processes of color inversion, flipping and rotating the gathered dataset IR images play a major role in enhancing the ability of the models to generalize and preventing overfitting. The initial CNN design with shallow nets and gradually increase the number of layers until satisfying performance is attained. However, the networks developed suffered from overfitting, and regularization techniques were investigated in an attempt to solve this issue. During experimentation and optimization of the model, the L2 regularization is added to the model to see if the model reduces overfitting and be able to better generalize. However, no improvement of generalizability for the final model was observed; therefore, no regularization was used except the inclusion of the Dropout layer. Additionally, more data augmentation techniques were used during the training phase of the CNN model, including zooming the images, moving all pixels of the image in one direction, either horizontally or vertically, while keeping the image dimensions the same, that essentially provides the model more flexibility to unobserved data, reduce overfitting, and enhance the ability to generalize. The exact final one-step model architecture, which is consisting of 15 convolutional layers in total, is shown in the table below.

Table 3. One-step CNN architecture

Type of Layer	***Output Size***	***Filter / Pool Size***	***Dropout probability***
Input Layer	60 × 60 × 1		
2 x (Convolution => ReLU =>BN)	60 × 60 × 32	3 × 3	
2 x (Convolution => ReLU =>BN)	60 × 60 × 64	3 × 3	
Convolution => ReLU =>BN	60 x 60 x 128	3 x 3	
Max-Pooling => Dropout	30 × 30 × 128	2 × 2	0.25
2 x (Convolution => ReLU =>BN)	30 × 30 × 128	3 × 3	
3 x (Convolution => ReLU =>BN)	30 × 30 × 256	3 × 3	
Max-Pooling => Dropout	15 × 15 × 256	2 × 2	0.25
5 x (Convolution => ReLU =>BN)	15 × 15 × 256	3 × 3	
Fully Connected => ReLU => BN => Dropout	512		0.25
Fully Connected	3		
Softmax	3		

Mathematical Representation of the one-step CNN model

Referring to the model architecture demonstrated in table 3, the architecture of the model can be conceptualized as three sets of convolutional layers, each one consisting of five consecutive instances of (CONV => ReLU => BN). The three convolutional layers sets are separated with two sets of POOL. Finally, the last convolutional layers set is followed with (Fully-Connected => ReLU => BN) layers set. The CONV denotes a Convolution layer, ReLU is the Activation function, BN refers to Batch Normalization and POOL refers to a Pooling layer. Next, the math will be broken down for each set of convolutional and pooling layers' sets.

It should be noted out that all the convolutional layers configurated throughout this work have the same zero-padding at $p = 1$, and stride $s = 1$. There are three types of zero padding – full, same, and valid. For the CNN model, the same padding configuration is used due to its preservation of the height and width of the input images, or tensors, which makes the design of the network architecture more efficient. For this model design, the spatial size was preserved in the convolution layer using the same padding and spatial size and decreased via the Pooling layers. The padding of the input image is shown in figure 26.

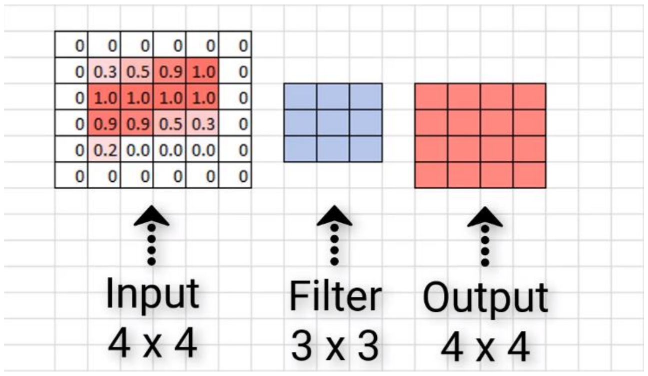


Figure 26. An example of same padding [43]

The size of convolutional layer output is computed using (7). This formula also applies when computing the output of pooling layers:

$$o = \left\lfloor \frac{n + 2p - m}{s} \right\rfloor + 1 \tag{7}$$

Here, the n term refers to the input pixel size of the image, p is the padding used, which is -as mentioned above- zero padding at $p = 1$, m is the filter size which equals 3, and s is the stride value which is set to 1 throughout all convolutional layers in this work. Therefore, applying (7) on the input image of size n, would result in having an output size equal to n, this size operation is shown in (8).

$$\left\lfloor \frac{n + 2 \times 1 - 3}{1} \right\rfloor + 1 = n \tag{8}$$

1. First Set of Convolutional layers

This set consists of five successive instances of (CONV => ReLU => BN). Let X[i , j] be the input IR image of size 60×60 pixels. The image is then fed into the first convolutional layer of the network, resulting an output, this output has depth dimension which is equal to the number of filters or kernels used in the convolutional layer as stated in table 3 Therefore, the first convolutional layer which has 32 filters that would convolve over the input image producing 32 of two-dimensional output A_f[i , j], that can be written as:

$$A_f[i,j] = \sum_{k_1=0}^{2} \sum_{k_2=0}^{2} \omega_f[k_1,k_2] X[i + k_1, j + k_2] \tag{9}$$

Here, f is the total number of filters in the first convolutional layer, and ω_f is denoting the kernel or the filter of size (3×3). For the ReLU activation function associated with every convolutional layer, this function works element wise such that:

$$B_f[i,j] = f_{ReLU}\left(A_f[i,j]\right) = \max\,(0, A_f[i,j]) \tag{10}$$

Next, the output of the activation function would be fed to batch normalization layer. The output of the batch normalization layer is $C[i,j]$. For the transformation, the mean and variance are calculated first as shown below in (11) and (12). If the mini-batch mean and variance are denoted by μ_B, σ_B^2 respectively, then,

$$\mu_B = \frac{1}{60 \times 60} \sum_{i=1}^{60} \sum_{j=1}^{60} B[i,j] \tag{11}$$

And the variance σ_B^2 is obtained as:

$$\sigma_B^2 = \frac{1}{60 \times 60} \sum_{i=1}^{60} \sum_{j=1}^{60} (B[i,j] - \mu_B)^2 \tag{12}$$

The normalized image of $B[i,j]$can be written as the output $C[i,j]$ as,

$$C[i,j] = \hat{B}\ [i,j] = \frac{B[i,j] - \mu_B}{\sqrt{\sigma_B^2}} \tag{13}$$

During the training of the model, $C[i,j]$ can be written in terms of the learnable parameters that are learned and updated by the CNN itself. This equation is shown below:

$$C[i,j] = \gamma B[i,j] + \beta \tag{14}$$

Here, γ and β are learnable parameters, which are functions of μ_B and σ_B^2.

The three layers of CONV, ReLU and BN repeats for four more times forming the first set of convolutional layers. Using of consecutive Convolution layers is necessary with a large number of pictures, supports the learning convergence, and performance of the CNN model. The mathematical representation is same for the rest of the five groups of CONV, ReLU and BN layers as shown in (9), (10), (11), (12), and (13). Therefore, after performing computation in these three layers for five times, an output image of $D[i,j]$ will be available as the input to the first pooling layer in the network.

1. First Max-Pool Layer

The goal of using pooling layer is to reduce the spatial size of $D[i,j]$ to 30×30. Where the size of pooling layer output is calculated using (7) through applying a 2×2 filter, a padding at $p = 0$, and stride $s = 2$. The effect of applying such parameters can be seen through the example in figure 27.

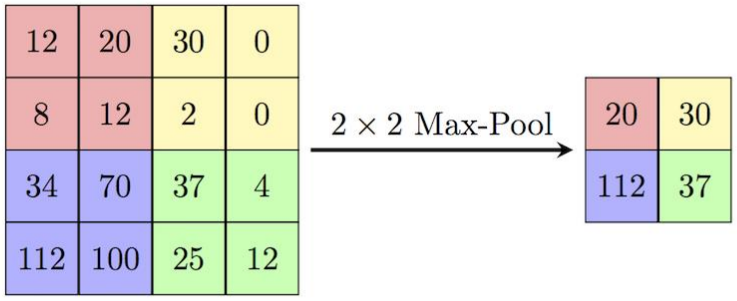


Figure 27. Effect of using a filter of size 2×2 with zero padding with a stride, s=2 on an input image of size 4×4.The resultant image is of size 2×2 [44].

The mathematical equation representing the POOL operation is given below:

$$E[i,j] = max(D[i+m, j+n]) \tag{15}$$

Here, $i = j = 29$. The output size is 30, and $m = n = 0{:}1$ since a 2×2 filter is used. Verification of the output size is shown by using (7) as:

$$o = \left\lfloor \frac{60 + 2 \times 0 - 2}{2} \right\rfloor + 1 = 30 \quad (16)$$

2. Second Set of Convolutional Layers

This set of layers works exactly the same way as shown for the previous set of layers with the exception of input size and number of filters used, where the input size throughout the layers in this layer is (30×30).

However, the filter size, padding, and stride value remained same for the corresponding layers. For the first CONV layer, the input image will be the output from the previous POOL layer denoted as $E[i,j]$ and the output of this layer is $F[i,j]$ as shown in (17). This output image is the input of the ReLU layer and output of ReLU layer is G$[i,j]$, as shown in (18). Batch Normalization is then applied to $G[i,j]$and output of this layer is $H[i,j]$as shown in (19).

$$F[i,j] = \sum_{k_1=0}^{2} \sum_{k_2=0}^{2} \omega_f[k_1,k_2]E[i+k_1,j+k_2]\,; i = j = 0{:}\,29 \quad (17)$$

$$G[i,j] = max(0, F[i,j]) \quad (18)$$

$$H[i,j] = \hat{G}\,[i,j] = \frac{G[i,j] - \frac{1}{30 \times 30}\sum_{i=1}^{30}\sum_{j=1}^{30} G[i,j]}{\sqrt{\frac{1}{30 \times 30}\sum_{i=1}^{30}\sum_{j=1}^{30}(G[i,j] - \frac{1}{30 \times 30}\sum_{i=1}^{30}\sum_{j=1}^{30} G[i,j])^2}} \quad (19)$$

Keeping $i = j = 0{:}\,29$, this set of convolutional layer consists of five patterns of (CONV => ReLU => BN). The output of performing computation in these three layers for four more times is denoted by $I[i, j]$, which is the input to the second pooling layer in the network.

3. Second Max-Pool Layer

The second max-pooling layer is identical to the first one, where the filter size is 2×2, padding at $p = 0$, and stride $s = 2$. The only difference here is the size of the input, where this pooling layer receives the output of the previous set of convolutional layers, $I[i, j]$ which is 30×30. Then, the size of the output denoted by $J[i, j]$ of the second max-pooling layer is:

$$o = \left\lfloor \frac{30 + 2 \times 0 - 2}{2} \right\rfloor + 1 = 15 \tag{20}$$

4. Third Set of Convolutional Layers

This set is also composed of five convolutional layers each has 256 filter and associated with an activation function followed by a batch normalization layer. For this set the input is $J[i, j]$ with size 15×15.

Keeping the filters' sizes, padding, and strides' sizes the same as in the previous two sets of convolutional layers would preserve the size of the input. Convolving the 256 filters of the first convolutional layer in this set over the elements of $J[i, j]$ would produce $K[i, j]$ as demonstrated in (21). The a ReLU function would be applied to $K[i, j]$, resulting in $L[i, j]$, as shown in (22). Next, $L[i, j]$ would be fed to the batch normalization layer, to get $M[i, j]$, as shown in (23).

$$K[i,j] = \sum_{k_1=0}^{2}\sum_{k_2=0}^{2} \omega_f[k_1,k_2]J[i+k_1, j+k_2]\,; i = j = 0{:}14 \tag{21}$$

$$L[i,j] = max(0, K[i,j]) \tag{22}$$

$$M[i,j] = \hat{L}\,[i,j] = \frac{L[i,j] - \frac{1}{15\times 15}\sum_{i=1}^{15}\sum_{j=1}^{15} L[i,j]}{\sqrt{\frac{1}{15\times 15}\sum_{i=1}^{15}\sum_{j=1}^{15}(L[i,j] - \frac{1}{15\times 15}\sum_{i=1}^{15}\sum_{j=1}^{15} L[i,j])^2}} \tag{23}$$

$M[i,j]$ would go through the same steps of (CONV => ReLU => BN) for four more times producing finally $N[i,j]$, which is the input to the next set of layers.

5. First Fully-Connected (Dense) Set of Layers

This set includes a Fully-Connected => ReLU => BN layers. The input to this set is $N[i,j]$ from the previous layers set, which size is 15×15. However, before applying the dense layer, $N[i,j]$is flattened to convert it from 2D to 1D. This is needed to predict the probability of each class in the Softmax layer. Also, the Fully-Connected layer takes inputs as a one-dimensional vector; it can't process 2D images as a Convolutional layer operates. Therefore, after performing a Flatten operation, the total data points remaining is $15 \times 15 \times 256 = 57{,}600$. This Flatten operation is shown in (24).

$$O[p] \times 256 = N[i,j] \times 256 \tag{24}$$

Here, $p = i \times j = 15 \times 15 = 225$. The Flatten operation can be visualized as shown in Figure 28.

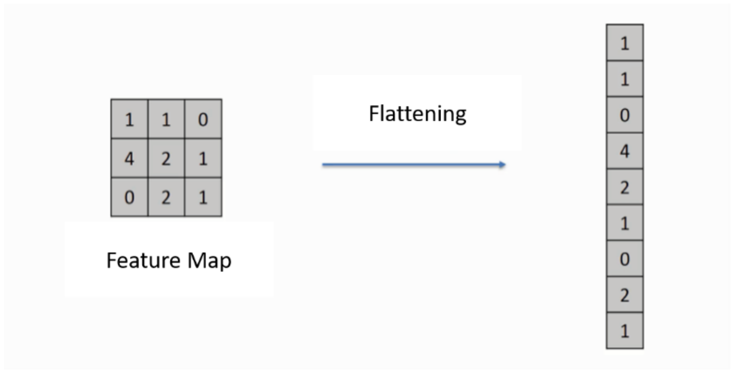


Figure 28. An example of Flatten operation

Now, the first set of Dense, ReLU, and BN layers can be described mathematically as in (25), (26), and (27) respectively.

$$a[q] = O[p] \cdot \boldsymbol{W} \tag{25}$$

Here, $q = 0{:}\,255$ and $\boldsymbol{W}$ is a learnable kernel/filter during training which is knows as weight matrix. The size of this matrix is learnt by CNN itself as the output size of dense layer is given by the user. Now, this output is the input to the ReLU layer.

$$b[q] = f_{ReLU}(a[q]) = max(0, a[q]) \tag{26}$$

$$c[q] = \frac{b[q] - \frac{1}{256}\sum_{q=1}^{256} b[q]}{\sqrt{\frac{1}{256}\sum_{q=1}^{256}(b[q] - \frac{1}{256}\sum_{q=1}^{256} b[q])^2}} \tag{27}$$

6. Second Fully-Connected (Dense) Layer

The final dense layer is used to provide the scoring function for each class, in this case three, and the scoring function is determined as,

$$g[r] = c[q]\,.\,\boldsymbol{K} \tag{28}$$

Here, $r = 0{:}\,2$ (total three classes) and **K** is learnable filter which is learnt and updated by the CNN itself during the training process.

7. Softmax Classifier

Finally, the probability of each class and training loss is calculated by the Softmax layer. The probability of each class P_c can be found by the scoring function $c[q]$,

$$P_c = \frac{e^{g[r]}}{\sum_{i=1}^{3} e^{g[r]}} \tag{29}$$

Loss of each class, L_c can be computed as,

$$L_c = -\log\left(P_c\right) \tag{30}$$

Total gross-entropy loss, L_{Total} is the average of total three classes,

$$L_{Total} = \frac{1}{3}\sum_{i=1}^{3} L_i \tag{31}$$

VIII. RESULTS AND EVALUATION

Programming Environment and Computational Resources

The CNN models in this work were developed using Python 3.5.7 programming language. Python contains special libraries for deep learning such as Keras, which is one of the most popular libraries that support the building of CNNs using backend libraries such as TensorFlow and Theano. For this work, Keras with TensorFlow backend was used. Training and evaluating a CNN model can be quite slow due to the number of computations required for each iteration. Furthermore, Keras with TensorFlow backend by default provides higher priority to the GPU's when placing operations if both CPU and GPU are available for the given operation. Thus, a GPU would be adapted for training the CNN in the proposed work, to speed up the computation. Access to the High-Performance Engineering (HiPE) servers is available, and this would be leveraged to train the CNN using GPUs.

The HiPE1 server is a Dell PowerEdge C4130 rack server is configured with dual 2.4 GHz Intel Xeon E5-2460 v4. The server has 128 GB of memory and two 800 GBs of SSD storage. Using the HiPE1 server for training the CNN has sped up the process since the HiPE1 server is equipped with two NVIDIA Tesla V100 accelerators. The next table 4 shows the specifications of the accelerators.

Table 4. NVIDIA Tesla V100 specifications

Architecture	Volta
Tensor Cores	640
GPU Cores	5,120
Memory	16GB HBM2
Memory Bandwidth	900GB/sec

Training and Evaluation Results

Many experiments have been carried out in the course of developing a reliable classification model to detect victims in infrared images. However, this section is dedicated to highlight and discuss the essential experiments that led to the final CNN model. First, experiments to train the CNN model using different datasets are discussed in an attempt to figure out the advantage of using the balanced dataset, which was developed in the course of this work. Then, the experiments and results that helped in defining the deep learning model architecture and hyperparameters values would be reviewed. Next, the benefits of the different preprocessing steps would be investigated.

However, it is noteworthy that the process of tuning the model's architecture and hyperparameters is a nested process, as the decision on one of the parameters is affecting the other parameters in the model. Therefore, for every experiment, many combinations of hyperparameters values were investigated, in an attempt to get the best performance of the tested CNN architecture.

- Dataset

As mentioned earlier, a balanced dataset was developed for the thesis work. The gathered dataset suffered from classes imbalance, which in turn impaired the performance of the deep learning model. This was reflecting in classifying the images of the smaller classes' sizes: "pet" and "no victims", where the best average evaluation accuracy gained when training a 10 convolutional layers CNN model on the imbalanced dataset was 89% as follows; 98% accuracy on "people", 81% and 88% on "no victims" and "pet" respectively. It shows obvious signs of biasing toward the largest class size towards the "people" category. The learning and validation loss and accuracy curves are shown in figure 29.

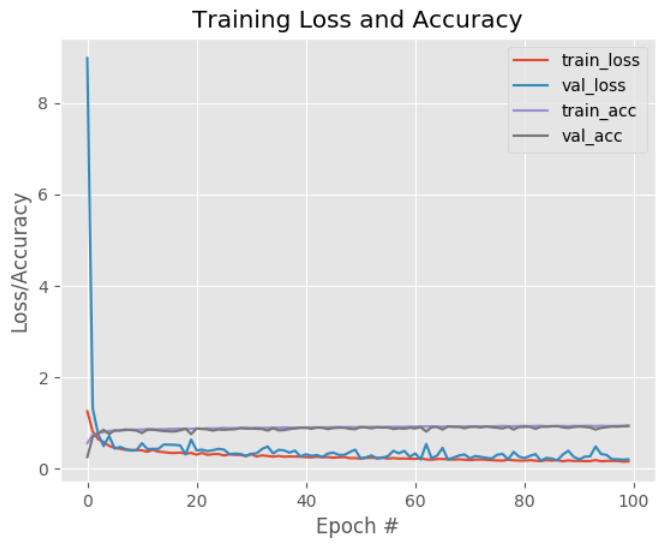


Figure 29. Loss/Accuracy plot for the imbalanced dataset

However, using the balanced dataset when training the CNN model, accuracies as much as 94% as an average was obtained; 99% on "people", 95%

accuracy on “pet” and 89% on “no victims.” Figure 30 shows the loss/Accuracy curves of training and testing a CNN model trained on the balanced dataset. Therefore, developing the balanced dataset added advantage to the work through enhancing the performance of the CNNs and mitigate the bias in the model.

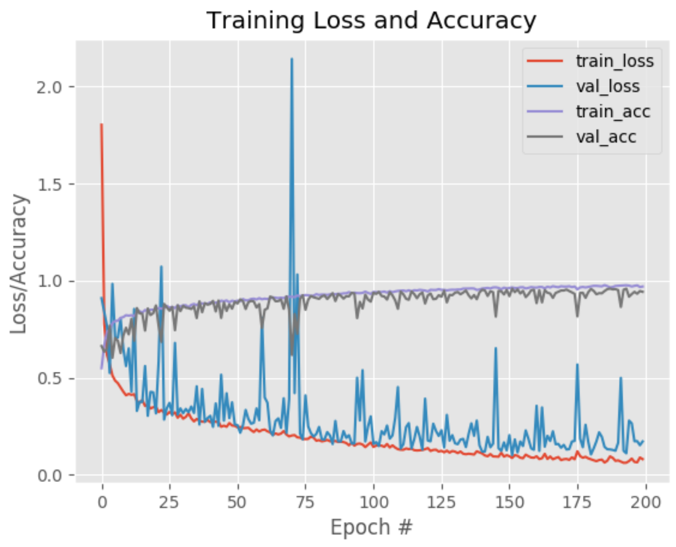


Figure 30. Loss/Accuracy plot for the balanced dataset

- Model Architecture and Hyperparameters Tuning

The process of deciding the number of layers in the implemented CNN architecture has passed through tens of experiments to determine the optimum number and arrangement of the different types of layers in the implemented CNN model. Since the model’s capacity is a tradeoff between performance and cost, shallow models were investigated first, then the number of layers was increased until adding more layers found to add no advantage to the model’s performance. Starting with one and two convolutional layers architecture, average accuracy attained was in the range of 83% to 86%. However, shallow models are very biased toward the “people” category. Indeed, these classification accuracies achieved with the shallow models are considered good, this can be referred to the low resolution of the IR images used to build the dataset, where the images don't include many features to be learned and the edges of the target objects in the image are well defined.

The next experiment was conducted with 12 convolutional layers CNN model, after trying a variety of combinations and hyperparameters values, the best classification accuracy was achieved using this model was 89.67%. In an attempt to get higher accuracies, 15 convolutional layers architecture was tested with variety of combinations of hyperparameters values starting from 0.0001 value for the initial learning rate combined with 32, 64, and 128 batch size, the loss curves resulted after training is terminated indicates that the initial learning rate is low as seen in figure 31.

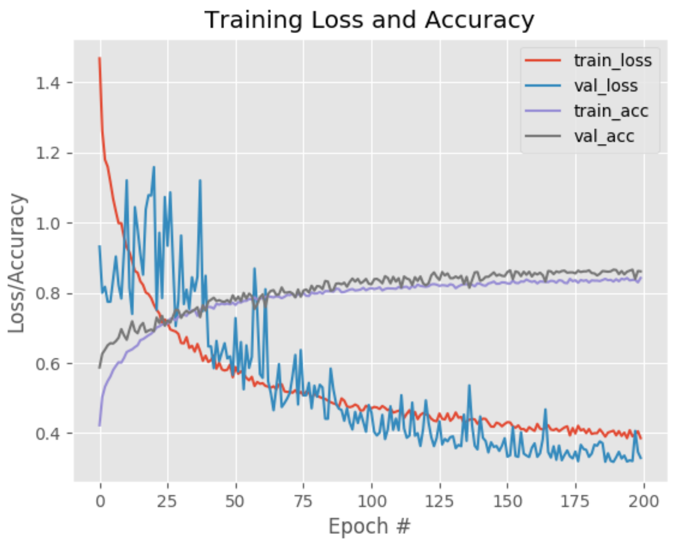


Figure 31. Loss/Accuracy plot of 15 convolutional layers model trained with 0.0001 lr and 128 batch size

The learning rate schedule mechanism is used in the course of this work due to the flexibility it adds to the tuning process. Therefore, the learning rate tuning task includes tuning the initial learning rate and the decay parameter. In this work, the decay parameter is set to the initial learning rate value divided by the number of epochs, which is a common practice. After running a number of experiments, 200 epochs found to be suitable for this work. This number of epochs was found to be sufficient for the learning accuracies reach sufficient levels without overfitting.

As the initial learning rate is increased to 0.00075 the loss training loss curve indicated a good learning rate. Also, good performance is attained with an average classification accuracy of 92.67%, where "*people*" classifying accuracy was 97%, 93% classification accuracy on "*no victims*" and 88% accuracy on the "*pet*" category. This model is trained for 200 epochs using and batch size of 128. The loss/accuracy plot for the 15 convolutional layers model is represented in figure 32.

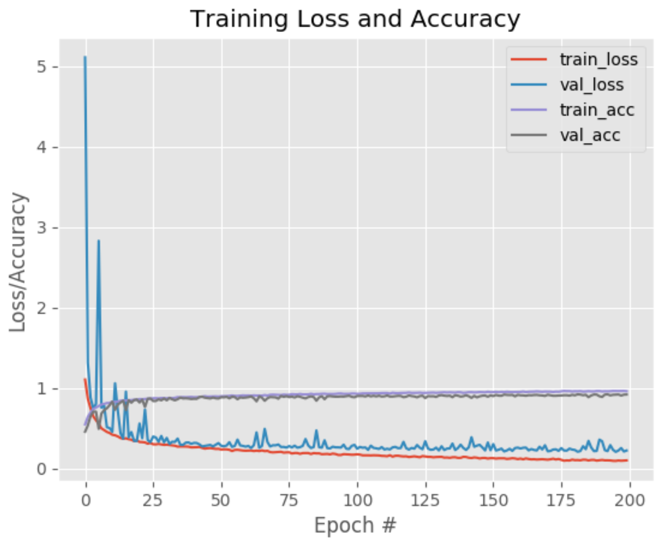


Figure 32. Loss/Accuracy plot for 15 convolutional layers model

However, experiments with a bigger model were carried out to investigate if more layers would enhance the performance of the deep learning model. For this experiment, a model consisting of 20 convolutional layers was under test, this model has achieved almost the same accuracies that the 15 layers model did, where the accuracy attained this time was 91%.

Through observation, adding the layers did not increase the performance of the accuracy. Hence, the 15 convolutional layers model was adopted for this work. Inspired by the results and performance of a variety of models explored in the course of this work during the training phase, an insight about good fixed learning rates were generated. Therefore, a batch of experiments was conducted using these fixed learning rates to enhance the performance.

The next experiment was conducted using a fixed learning rate of 0.075. This enabled the deep learning model to achieve 96% overall accuracy, 98% on "*people*" and 95% on the other two classes. Moreover, the loss/accuracy plot in figure 33 shows that good fit is attained also, and the model is not suffering from overfitting, where regularization mechanisms including incorporating drop out layers and applying data augmentation through rotating the dataset images, shifting, and zooming. Figures 6 and 7 show the confusion matrix, and its normalized version respectively for this CNN model.

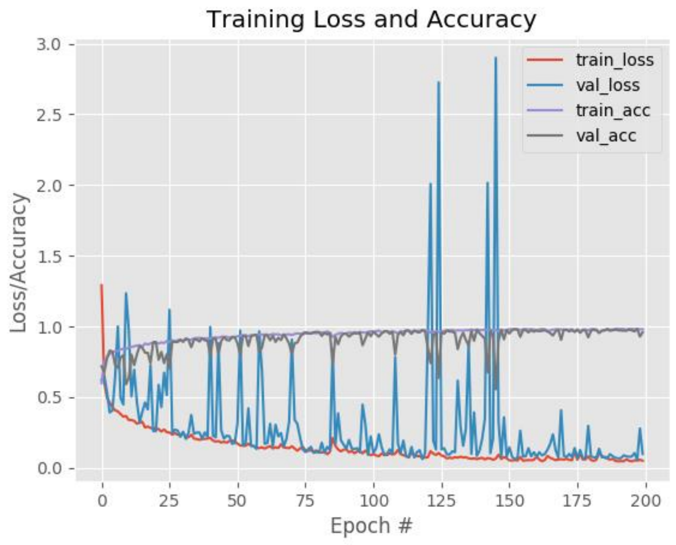


Figure 33. Loss/ accuracy plot of 15 layers model, trained for 200 epochs with 0.075 lr and 128 batch size

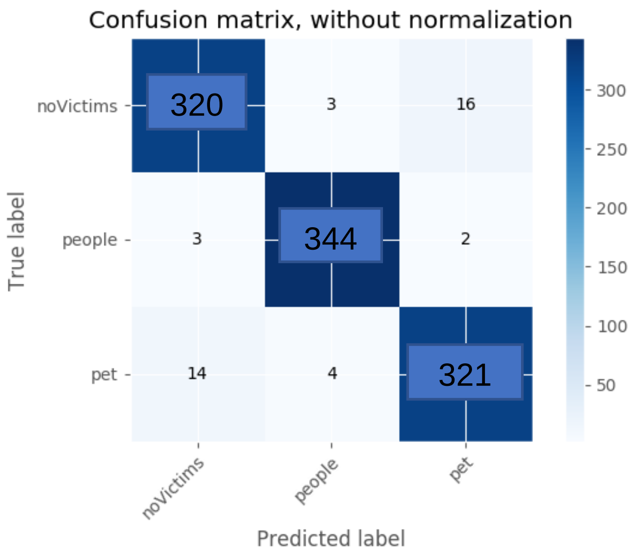


Figure 34. Confusion matrix

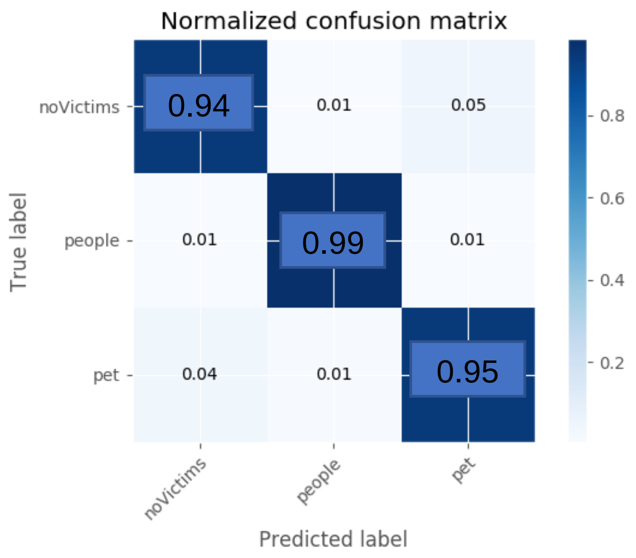


Figure 35. Normalized confusion matrix

However, the previous model is trained using regularization techniques to mitigate any potential overfitting and obtain better performance. In the next section, the model's performance would be assessed without using proposed preprocessing mechanisms in an attempt to evaluate the advantages added when using these mechanisms.

- Preprocessing

  The aforesaid preprocessing mechanisms investigated in this work include using Histogram Equalization (HE) and Contrast-Limited Adaptive Histogram Equalization (CLAHE). This experiment is dedicated to investigating the effect of equalizing the histogram of the IR images before being fed to the CNN model, either for training or testing. Here, HE is applied to the IR images used to train and evaluate the CNN model, the results attained indicate deterioration in the model's performance, where the average accuracy of the model is 86%. This impairment can be referred to as the over brightness associated with equalizing the IR image's histogram that leads to showing up minor features in the images.

Next, CLAHE was applied to the IR images for training and evaluating the 15 layers CNN. This time the model's performance remained the same and no improvement noticed when adapting CLAHE. Therefore, there is no point of considering either histogram equalization or contrast-limited histogram equalization in this work.

IX. CASCADED TWO-STEP CNN MODEL

In this chapter, another arrangement of CNN models, which is a Cascaded-CNN (C-CNN) model is introduced and benchmarked as an extended work in an attempt to reveal any potential advantages of breaking down the classification problem into two different deep learning models.

While CNN may be powerful enough to learn one step reconstruction, such networks are more prone to overfitting, unless there is a vast amount of training data. Besides, training such networks may require a long time as well as careful fine training steps. Therefore, in some cases, it may be better to use CNNs for iterative reconstruction approaches as in cascaded CNN architectures. [38]

Cascading has been widely used in the field of detection, where it was popularized by the seminal Viola-Jones framework before its popularity has extended to other pipelines such as DPM and CNN [36]. Also, this mechanism proved its efficiency, since state-of-the-art object detection algorithms adopt cascaded models. Such as the work proposed in [37], where using of cascaded convolutional neural network is proposed for face detection in an attempt to find a powerful alternative mechanism for the prohibitively expensive models that were used to address the face detection problem through two-stage model, the first stage is a network for region proposal generation, whereas, the following stages are dedicated for face detection.

In this work, a cascaded discriminative model that is capable of classifying an input IR image into one of three categories: “people”, “pet” or “no victims” is implemented. However, this task would be accomplished in two stages using two different CNN models, the first one is designated to classify the input into one of two classes, either “people” or “no people”. Then, the images that are classified as “no people” would be fed to the second CNN model which is responsible for classifying the input IR images to either “pet” or “no victims.” Figure 36 shows a high-level representation of the cascaded model allowing to train all the neural networks simultaneously.

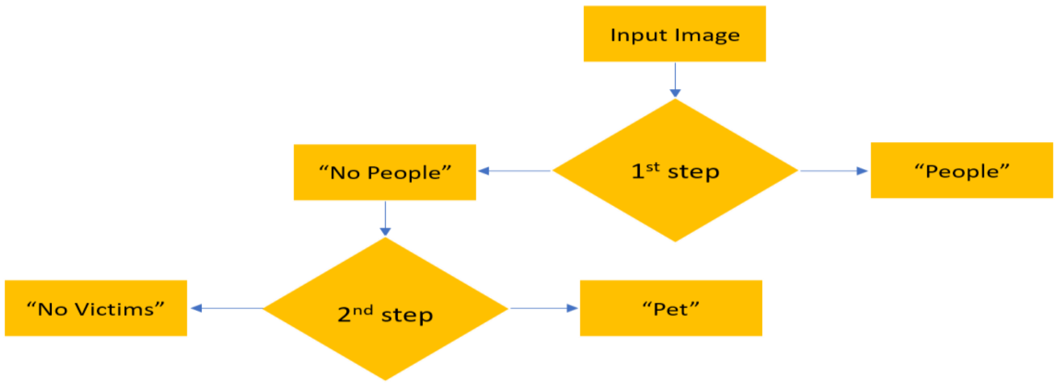


Figure 36.Two-step cascaded CNN model for victims' detection

For training the cascaded CNN model, the same balanced dataset that is used to train the one-step CNN model had to be rearranged so dataset balance is maintained. Concurrently, the largest amount of data available is invested in training the model, and higher performance accuracies are attained. The figure 37 shows the rearranged datasets used to train and validate each CNN in the cascaded model.

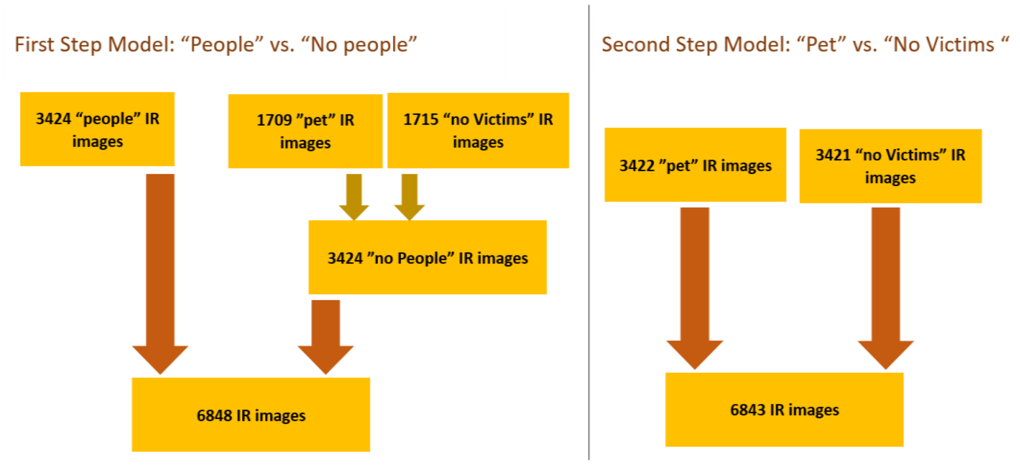

First Step Model: "People" vs. "No people"
3424 "people" IR images
1709 "pet" IR images
1715 "no Victims" IR images
3424 "no People" IR images
6848 IR images
Second Step Model: "Pet" vs. "No Victims "
3422 "pet" IR images
3421 "no Victims" IR images
6843 IR images

Figure 37. Datasets associated with each CNN model in the cascaded model

Following the same methodology adopted for defining the architecture and parameters of the one-step classification model, both CNNs in the cascaded model architecture design process has passed through the same steps. Even though both networks are trained using almost the same amount of data; however, each model has distinct architecture. The "*people*" detection model's architecture is inspired by the family of VGGNet [26], whereas the architecture of the second network in the model is based on LeNet [25] architecture.

Both cascaded models investigated take an input image of 60 × 60 pixels and processes it through several convolution, max-pooling, and fully- connected layers before providing the final output of either three classes: "*people*", "*pet*" or "*no victims*". Finally, a Softmax Classifier is used to classify the image into distinct categories. Tables 5 and 6 demonstrate the exact architectures of first and second step models respectively.

Table 5. "People" vs. "No people" model architecture

Type of Layer	*Output Size*	*Filter / Pool Size*	*Dropout probability*
Input Layer	60 × 60 × 1		
2 x (Convolution => ReLU =>BN)	60 × 60 × 32	3 × 3	
Max-Pooling => Dropout	60 × 60 × 32	2 x 2	0.1
2 x(Convolution => ReLU =>BN)	30 x 30 x 64	3 x 3	
Max-Pooling => Dropout	30 × 30 × 64	2 × 2	0.2
2 x (Convolution => ReLU =>BN)	30 × 30 × 128	3 × 3	
(Convolution => ReLU =>BN)	30 × 30 × 256	3 × 3	
Max-Pooling => Dropout	15 × 15 × 256	2 × 2	0.3
Fully Connected => ReLU => BN => Dropout	500		0.4
Fully Connected	2		
Softmax	2		

Table 6. "Pet" vs. "No victims" model architecture

Type of Layer	*Output Size*	*Filter / Pool Size*	*Dropout probability*
Input Layer	60 × 60 × 1		
Convolution => ReLU =>BN	60 × 60 × 20	3 × 3	
Max-Pooling => Dropout	60 × 60 × 20	2 x 2	0.1
Convolution => ReLU =>BN	30 x 30 x 50	3 x 3	
Max-Pooling => Dropout	30 × 30 × 50	2 × 2	0.2
Convolution => ReLU =>BN	15 × 15 × 60	3 × 3	
Max-Pooling => Dropout	15 × 15 × 60	3 x 3	0.25
Convolution => ReLU =>BN	7 × 7 × 60	3 × 3	
Max-Pooling => Dropout	3 × 3 × 60	3 x 3	0.25
Fully Connected => ReLU => BN => Dropout	10		0.3
Fully Conneceted	2		
Softmax	2		

As mentioned above, the cascaded model consists of two different CNN architectures that are implemented and learned separately. The first step model is dedicated in deciding human existence has been trained using a learning rate scheduler with an initial learning rate value of 0.002, and 64 batch size for 200 epochs. The performance of this model has reached very satisfying levels using five convolutional layers architecture as shown in the table 6. The model's accuracy on "*people*" classification is 99%, and 99% on the "*no people*" category. Figures 38, 39, and 40 show the loss/accuracy plot for training and validation, the confusion matrix of the classification, and the normalized confusion matrix respectively.

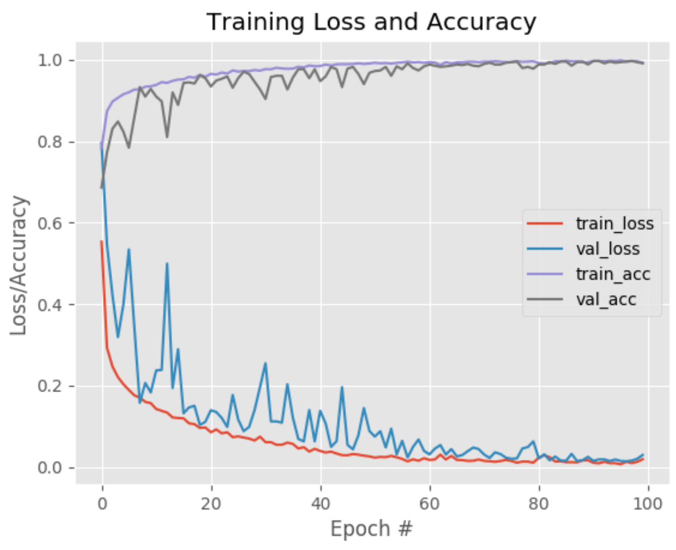


Figure 38. Loss/accuracy plot for first-step CNN model

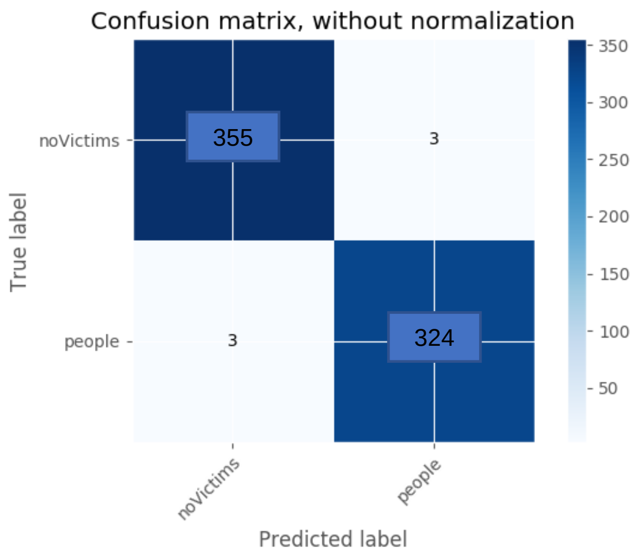


Figure 39. Confusion matrix of first-step CNN model

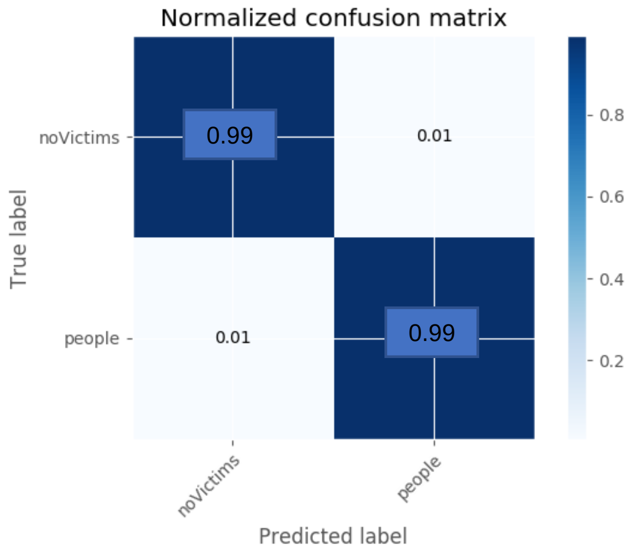


Figure 40. Normalized confusion matrix of first-step CNN model

Whereas the second step model which is devoted to classifying the images into either "*pet*" or "*no victims*" category, is trained with an initial learning rate of 0.003 along with (0.003/300) decay, and a 32 batch size for 300 epochs. The accuracy of this model is 97% on the "*pet*" class, and 90% on the "*no victims*" class. The loss/accuracy plot for this model is shown below in figure 41.

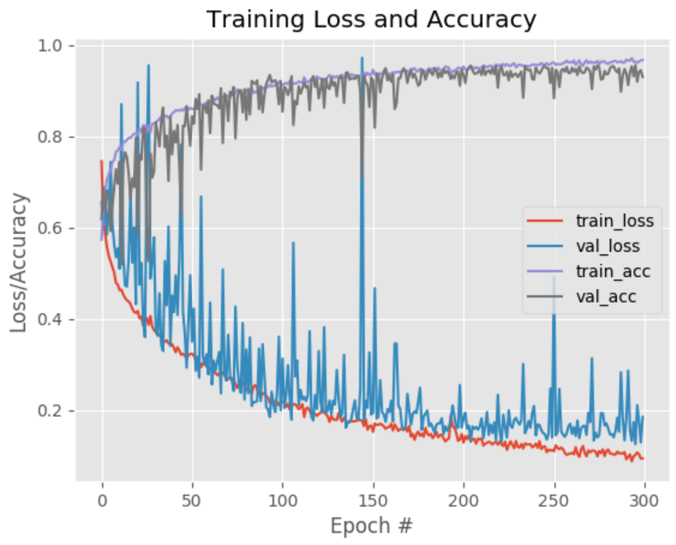


Figure 41. Loss/accuracy plot for the second-step CNN model

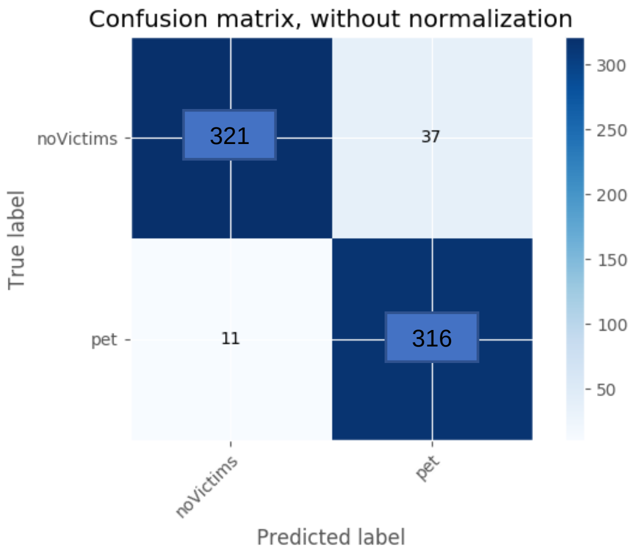


Figure 42. Confusion matrix of second-step CNN model

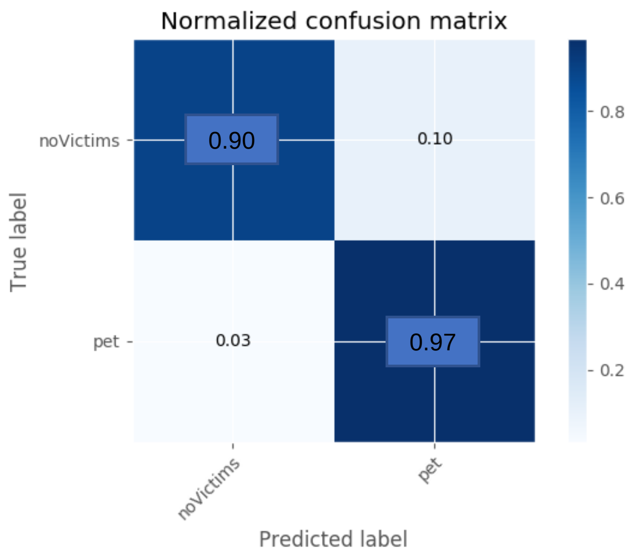


Figure 43. Normalized confusion matrix of second-step CNN model

However, the outcome accuracies of the second step cascaded model can be cast as a statistical problem, since Softmax classifiers assign probabilities for each class label. Therefore, the attained accuracies from the second step model on "*pet*" and "*no victims*" classes are the conditional probabilities given that the IR images are classified as "*no people*." Therefore, to compute the overall accuracy of the cascaded model on the second stage categories; "*pet*" and "*no victims*," the Bayes' theorem and the low of total probability are used with (32) and (33):

$$P\left(B_{"pet"}\right) = P\left(B_{"pet"}\middle|A_{\text{no people"}}\right) P\left(A_{"no\ people"}\right) \tag{32}$$

$$P(B_{"no\ victims"}) = P(B_{"no\ victims"}|A_{\text{no people"}})\ P(A_{"no\ people"}) \tag{33}$$

Where $P(A_{"no\ people"})$ indicates the classification accuracy on "*no people*" class from the first step model, which detects human existence in the input image. Whereas $P(B_{"pet"})$ and $P(B_{"no\ victims")}$ denote the classification accuracies of the second step model on “*pet*” and “*no victims*” respectively. The final outcomes of the implemented cascaded model in this work are 99% classification accuracy on “*people*”, 96% on the “*pet*” class and 89% on the “*no victims*” category.

Referring to the results achieved from the CNN models proposed in this thesis work, it can be noticed that even though the accuracies achieved using the one-step model are slightly higher than the accuracies gained from the two-steps cascaded CNN model. But still, both models’ performances are comparable and yet satisfying. However, it is noteworthy that the cascaded model architecture consists of a smaller number of convolutional layers than the one-step model. Hence, there is a tradeoff between cost and performance.

X. CONCLUSION

This work has been an investigation into how deep learning models for human and pet detection can be applied to help firefighters in situations of foreign structures layouts. The proposed solution suggests detecting victims in thermal images of the burning site that are captured and transmitted by the AESV, detection would be performed through a CNN model that is trained to detect humans and pets, so firefighters would be able to prioritize the locations that they should target. This can enhance their safety and ensure better exploit of the time to accomplish the rescue mission.

The approach of detecting victims in IR images was through design and development of a deep convolutional neural network capable of predicting human and pet's existence in IR images. For this work, two different styles of convolutional neural networks were investigated; one-step CNN model and two-step cascaded CNN model.

A dataset including three categories: "people,", "pet," and "no victims" was developed for training the deep learning models. When developing this dataset, it was considered that the data set represents the different types of situations that arise due to the fact that both the target objects and the AESV are not static. In the case of detecting humans, the data set needs to contain enough images to represent the various poses and views of the human body captured from different angles and distances. The most direct way is to make sure that these types of situations are covered during data acquisition and designed to cover these diverse situations.

However, the gathered dataset suffered from class imbalance, where the data points count in "people" class surpasses the other two classes, which caused the classification results to be biased toward the “people” category. Therefore, the oversampling technique was adapted to solve this issue.

The other issue in the dataset, that it doesn’t include IR images captured in high-temperature environments representing the temperature range in real fire situations. In an attempt to overcome this limitation, the dataset images were inverted in color to mimic the IR images style when the background temperature is higher than the targets’ temperature.

The balanced inverted dataset was used to train and evaluate the CNN models developed. Two CNN model styles were implemented and benchmarked in this work. The first style is a one-step CNN model that accepts an input IR image and classify it into one of the three aforesaid classes. Whereas the second one is a cascaded CNN model which is designed to accomplish the task of IR images classification in two distinct CNN models. The first model classifies the input IR image into one of two categories either “people” or “no people.” The images that are labeled as “no people” would be fed to the next stage in the cascaded CNN model to be classified into “pet” or “no victims” categories.

The process of developing and defining the deep learning models architectures and designs has passed through many steps of tuning and tweaking of the CNN parameters and testing of many potential preprocessing. Regularization techniques before making a decision on the optimum combination of parameters and methods were adapted to obtain high accuracy values.

The best test accuracy of 96.3% without overfitting was achieved using the one-step CNN model consists of 15 convolutional layers. However, comparable results were achieved using the cascaded version with a total of 11 convolutional layers. The cascaded model was able to reach an accuracy of 94.6%.

XI. FUTURE WORK

The next step is to assess the danger level associated with each detected person and define the best path to reach those persons in danger. By defining a suitable scoring criterion for danger level associated with each detected person, the firefighters can prioritize their tasks during fire circumstances. The AESV is equipped with gas and temperature sensors, and their output data would be parameters in the danger assessment criterion. The information from the GPS unit, available in the AESV, would be essential in defining the location of detected people when utilizing the thermal images used in classification. A path can then be determined in order to get to the detected victims while considering the firefighter’s safety. Future prediction techniques such as the one used in [42], could be exploited to predict the temperatures and gas levels on every location and recognize the immediate danger of people experiencing high environmental levels.

REFERENCES

[1] H. Zulkifli, "Understanding Learning Rates and How It Improves Performance in Deep Learning," 21 Jan 2018. [Online]. Available: https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10. [Accessed July 2019].

[2] H. Zulkifli, "Understanding Learning Rates and How It Improves Performance in Deep Learning," 21 january 2018. [Online].

[3] R. Yamashita, M. Nishio and Richard K, "Convolutional neural networks: an overview and apllication in radiology," 2018.

[4] Wikipedia, "Deep learning," [Online]. Available: https://en.wikipedia.org/wiki/Deep_learning. [Accessed 22 August 2019].

[5] E. Valldor*, Person Detection in Thermal Images Using Deeplearning (MS Thesis),* 2018.

[6] M. Ulrich, T. Hess, S. Abdulatif and B. Yang, "Person Recognition based on Micro-Doppler and Thermal Infrared Camera Fusion for Firefighting," in *21st International Conference on Information Fusion (FUSION)*, Cambridge, UK, 2018.

[7] Tsung-Han Chan, Kui Jia, Shenghua Gao, Jiwen Lu, Zinan Zeng and Yi Ma, "PCANet: A Simple Deep Learning Baseline for Image Classification?," *IEEE TRANSACTIONS ON IMAGE PROCESSING,* 2015.

[8] A. A. Trofimova, A. Masciadri and F. Verones, "Indoor Human Detection Based on Thermal Array Sensor Data and Adaptive Background Estimation," *Journal of Computer and Communications, Vol.5 No.4,* pp. 16-28, 2017.

[9] S. Sudhakar, "Histogram Equalization," 10 July 2017. [Online]. Available: https://towardsdatascience.com/histogram-equalization-5d1013626e64. [Accessed July 2019].

[10] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *ICLR*, 2015.

[11] A. Sharma, "Why Rectified Linear Units (ReLUs) actually work?," 12 February 2019. [Online]. Available: https://adityashrm21.github.io/Why-Rectified-Linear-Units-Work/.

[12] G. Seif, "Handling Imbalanced Datasets in Deep Learning," 19 November 2018. [Online]. Available: https://towardsdatascience.com/handling-imbalanced-datasets-in-deep-learning-f48407a0e758. [Accessed September 2019].

[13] J. Schlemper, J. Caballero, J. V. Hajnal, A. N. Price and D. Rueckert, "A Deep Cascade of Convolutional Neural Networks for Dynamic MR Image Reconstruction," *IEEE Transactions on Medical Imaging,* vol. 37, no. 2, pp. 491 - 503, 2018.

[14] I. S and S. C, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *ArXiv150203167*, 2015.

[15] A. Rosebrock, Deep Learning for Computer Vision with Python, PyImageSearch, 2017.

[16] D. A. Roebrock, Deep learning for computer vision with python, PYIMAGESEARCH, 2017.

[17] X. Ren , S. Du and Y. Zheng, "Parallel RCNN: A Deep Learning Method for People Detection Using RGB-D Images," in *10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI),* Shanghai, China, 2017.

[18] S. Ren , K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence (Volume: 39 , Issue: 6 , June 1 2017),* pp. 1137 - 1149, 2017.

[19] H. Qin, J. Yan, X. Li and X. Hu, "Joint Training of Cascaded CNN for Face Detection".

[20] A. Pinales and D. Valles, "Autonomous Embedded System Vehicle Design on Environmental, Mapping and Human Detection Data Acquisition for Firefighting Situations," in *IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, Vancouver, BC, Canada, 2018.

[21] S. R. a. P. Norvig, Artificial Intelligence: A Modern Approach, Upper Saddle, NJ, USA: Prentice Hall Press, 2009.

[22] S. Mashiyama , J. Hong and T. Ohtsuki, "Activity recognition using low resolution infrared array sensor," in *2015 IEEE International Conference on Communications (ICC)*, London, UK, 2015.

[23] D. lizard, "Zero Padding in Convolutional Neural Networks explained," 14 February 2018. [Online]. Available: https://deeplizard.com/learn/video/qSTv_m-KFk0. [Accessed 13 October 2019].

[24] H. Li, Z. Lin, X. Shen, J. Brandt and G. Hua, "A Convolutional Neural Network Cascade for Face Detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[25] Y. LeCun, Y. Bengio and G. Hinton, "Deep learning," *Nature, Volume 521*, pp. 436-444, 28 May 2015.

[26] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard and L. Jackel, "Backpropagation Applied to Handwritten Zip Code Recognition".

[27] L. Bottou, C. Cortes, J.S. Denker, H. Drucker, I. Guyon, L.D. Jackel, Y. LeCun, U.A. Muller, E. Sackinger, P. Simard and V. Vapnik, "Comparison of classifier methods: a case study in handwritten digit recognition," in *Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 3 - Conference C: Signal Processing (Cat. No.94CH3440-5)*, Jerusalem, Palestine, 1994.

[28] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet Classification with Deep Convolutional".

[29] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 25 (NIPS)*, 2012.

[30] M. Klenner, C. Zech, A. Hülsmann, . J. Kühn, M. Schlechtweg, K. Hahma, B. Kleiner, M. Ulrich and O. Ambacher, "A portable W-band radar system for enhancement of infrared vision in fire fighting operations," in *SPIE Security + Defence*, Edinburgh, United Kingdom, 2016.

[31] K. Kim and S. Y. Chun, "SREdgeNet: Edge Enhanced Single Image Super Resolution using Dense Edge Detection Network and Feature Merge Network," in *Computer Vision and Pattern Recognition*, 2018.

[32] V. John , S. Mita , Z. Liu and B. Qi, "Pedestrian Detection in Thermal Images Using Adaptive Fuzzy C-Means Clustering and Convolutional Neural Networks," in *2015 14th IAPR International Conference on Machine Vision Applications (MVA)*, Tokyo, Japan, 2015.

[33] E. S. Jeon, J. H. Kim, H. G. Hong, G. Batchuluun and K. R. Park, "Human Detection Based on the Generation of a Background Image and Fuzzy System by Using a Thermal Camera," *Sensors,* 30 March 2016.

[34] F. Jaradat and D. Valles, "Human Detection Approach for Burning Building Sites Using Deep Learning," in *CSCI"18*, Las Vegas, 2018.

[35] F. Jaradat and D. Valles, "Early Warning Embedded System of Dangerous Temperature Using Single Exponential Smoothing for Firefighters Safety," in *Int'l Conf. Embedded Systems, Cyber-physical Systems, & Applications (ESCS'18)*, Las Vegas, 2018.

[36] J. Gua, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, X. Wang, T. Liu, L. Wang, G. Wang, J. Cai and T. Chen, "Recent Advances in Convolutional Neural Networks," 2017.

[37] B. Grossfeld, "A simple way to understand machine learning vs deep learning," 18 July 2018. [Online]. Available: https://www.zendesk.com/blog/machine-learning-and-deep-learning/. [Accessed 22 August 2019].

[38] R. Girshick, "Fast R-CNN," in *IEEE International Conference on Computer Vision(ICCV),* Santiago, Chile, 2015 .

[39] Georgios Zoumpourlis , Alexandros Doumanoglou , Nicholas Vretos and Nicholas Vretos Petros Daras, "Non-linear Convolution Filters for CNN-based Learning".

[40] P. Fritsche, B. Zeise, P. Hemme and B. Wagner, "Fusion of Radar, LiDAR and Thermal Information for Hazard Detection in Low-Visibility Environment," in *2017 IEEE International Symposium Safety, Security and Rescue Robotics (SSRR),* Shanghai, China, 2017.

[41] R. C. Fong, Walter J. Scheirer and David D. Cox, "Using human brain activity to guide machine learning," 2018.

[42] FLIR, "FREE FLIR Thermal Dataset for Algorithm Training," [Online]. Available: https://www.flir.com/oem/adas/adas-dataset-form/.

[43] ELEKTRA, "Pedestrain detection," [Online]. Available: [http://adas.cvc.uab.es/elektra/datasets/pedestrian-detection/.

[44] I. Ćirić , Ž. Ćojbašić and V. Nikolić , "Computationally intelligent system for thermal vision people detection and tracking in robotic applications," in *11th International Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Services (TELSIKS),* Nis, Serbia, 2013.

[45] T. Chavdarova and F. Fleuret, "Deep Multi-Camera People Detection," in *16th IEEE International Conference on Machine Learning and Applications (ICMLA),* Cancun, Mexico, 2018.

[46] J. Brownlee, "Parametric and Nonparametric Machine Learning Algorithms," 14 March 2016. [Online]. Available: https://machinelearningmastery.com/parametric-and-nonparametric-machine-learning-algorithms/. [Accessed 22 August 2019].

[47] J. Brownlee, "How to use Learning Curves to Diagnose Machine Learning Model Performance," 27 February 2019. [Online]. Available: https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/. [Accessed September 2019].

[48] J. Brownlee, "How to Configure the Learning Rate When Training Deep Learning Neural Networks," 23 January 2019. [Online]. Available: https://machinelearningmastery.com/learning-rate-for-deep-learning-neural-networks/. [Accessed August 2019].

[49] V. Bianco , M. Paturzo , A. Finizio, . K. A. Stetson and . P. Ferraro, "Portable IR Laser System for Real-Time Display of Alive People in Fire Scenes," *Journal of Display Technology,* pp. 834 - 838, 2014.

[50] K. Ahirwar, "Everything you need to know about Neural Networks," 1st November 2017. [Online]. Available: https://hackernoon.com/everything-you-need-to-know-about-neural-networks-8988c3ee4491. [Accessed August 2019].

[51] J. M. Johnson and T. M. Khoshgoftaar, "Survey on deep learning with class," *Journal of Big Data,* 2019.

[52] "What is the differences between artificial neural network (computer science) and biological neural network," [Online]. Available: https://www.quora.com/What-is-the-differences-between-artificial-neural-network-computer-science-and-biological-neural-network. [Accessed August 2019].

[53] "Max-pooling / Pooling," [Online]. Available: https://computersciencewiki.org/index.php/Max-pooling_/_Pooling.

[54] "Histogram Equalization," 18 December 2015. [Online]. Available: https://docs.opencv.org/3.1.0/d5/daf/tutorial_py_histogram_equalization.html. [Accessed June 2019].

[55] "CS23 iIn Convolutional Neural Networks for Visual Recognition," [Online]. Available: http://cs231n.github.io/convolutional-networks/.