**Differentiate Friends from Foes:**

**How Deep Learning Techniques and Images from Computer Games Could Help Identify Combatants in Urban Irregular Combat**

Final Report



M.Sc. Data Science and AI

**Name: Denis Remo**

**Submission Date: 16th of August 2022**

**Supervisor: Aikaterini Chatsiou**

This report is submitted in partial fulfilment of the requirement for the degree of **M.Sc. Data Science and Artificial Intelligence** by **Denis Remo**.

Abstract

Identifying and differentiating friends from foes is one of the many problems a soldier on the ground encounters on the battlefield. The combatants who participate in urban irregular combat use different equipment and different tactics. To solve this problem, deep learning techniques were used. In some cases, the images for the image dataset which is used to train deep learning models could not be accessed due to different reasons. The dataset used in the project was composed of images from the computer game called Counter Strike Global Offensive. This image dataset was increased by its author by using data augmentation. The reason for this decision is due to the fact computer games have become more realistic over the years. In the literature reviewed for this project, two types of deep learning model were commonly used in human and military recognition problems: YOLO and its versions, and encoder-decoder model. In the methodology, a few deep learning models from the literature were replicated to see which one is the most effective when trained with this project’s image dataset. The results were mixed, as an encoder-decoder model had the fewest losses, while a YOLO variant was the most efficient in terms of memory. A GAN model was used to increase the dataset. However, the model did not generate images which could be used. Data augmentation techniques were used to prove the only hypothesis of this project. The hypothesis was that if data augmentation has influence of the effectiveness of a model when it is trained with images already augmented. It was discovered that data augmentation lowered the validation accuracy of a model but decreases the number of false positives and false negatives. This project could be used in the development of AI in computer games which are part of the first-person shooter genre. Also, it could be used to suggest the usage of images from computer games for solving deep learning problems which are not computer games related.

Table of Contents

[Abstract](file:///C:\Users\denis\OneDrive\Desktop\MSc%20Dissertation%20Template.docx#_Toc82699435) 1

[Table of Contents 2](file:///C:\Users\denis\OneDrive\Desktop\MSc%20Dissertation%20Template.docx#_Toc82699436)

[List of Figures 3](file:///C:\Users\denis\OneDrive\Desktop\MSc%20Dissertation%20Template.docx#_Toc82699436)

[List of Tables 4](file:///C:\Users\denis\OneDrive\Desktop\MSc%20Dissertation%20Template.docx#_Toc82699436)

[1. Introduction 5](file:///C:\Users\denis\OneDrive\Desktop\MSc%20Dissertation%20Template.docx#_Toc82699437)

[2. Acknowledgements 7](file:///C:\Users\denis\OneDrive\Desktop\MSc%20Dissertation%20Template.docx#_Toc82699438)

[3. Literature REVIEW 8](file:///C:\Users\denis\OneDrive\Desktop\MSc%20Dissertation%20Template.docx#_Toc82699439)

[4. Research Problems 18](file:///C:\Users\denis\OneDrive\Desktop\MSc%20Dissertation%20Template.docx#_Toc82699441)

[5. Methodologies 19](file:///C:\Users\denis\OneDrive\Desktop\MSc%20Dissertation%20Template.docx#_Toc82699445)

**5.1 Dataset…………………………………………………………………………………………………………19**

**5.2 GAN……………………………………………………………………………………………………………..21**

**5.3 Preprocessing……………………………………………………………………………………………….21**

**5.4 Replicated models from literature………………………………………………………………..22**

**5.5 Effectiveness metrics…………………………………………………………………………………….24**

**5.6 Data augmentation……………………………………………………………………………………….26**

[6. Resources 27](file:///C:\Users\denis\OneDrive\Desktop\MSc%20Dissertation%20Template.docx#_Toc82699451)

[7. Results 28](file:///C:\Users\denis\OneDrive\Desktop\MSc%20Dissertation%20Template.docx#_Toc82699456)

[8. Discussion AND CONCLUSION 41](file:///C:\Users\denis\OneDrive\Desktop\MSc%20Dissertation%20Template.docx#_Toc82699457)

[9. Limitations and Future Works 42](file:///C:\Users\denis\OneDrive\Desktop\MSc%20Dissertation%20Template.docx#_Toc82699458)

[References ……………………………………………………………………………………………………….45](file:///C:\Users\denis\OneDrive\Desktop\MSc%20Dissertation%20Template.docx#_Toc82699459)

**LIST OF FIGURES**

Figure1: Deep learning model identifying object in images from computer games ..……………8

Figure2: Ukrainian soldier with a yellow armband ……………………………………….………………………9

Figure3: Digitalized camouflage patterns in action …………………………………………………………….10

Figure4: Examples of urban camouflage ………………………………….…………………………………………11

Figure5: Polish resistance fighters during the Warsaw Uprising …..…………………………………….11

Figure6: Typical CNN model ………………………………………………………………………..…………………….13

Figure 7: A YOLO variant identifying objects ……………………………………………………………..………14

Figure8: How YOLO mapping an image to identify an object ……………………..……………………. 15

Figure9: An encoder-decoder model differentiate an object from the background ……….….16

Figure10: Classical data augmentation techniques …………..……………………………………………….17

Figure11: Characters representing the classes and examples of data augmentation in the image dataset………………………………………………………………………………………….…………………..…….20

Figure12: Examples of situations in which characters could be found in …………………………...20

Figure13: Design of the Qassim et al residual VGG16…………………………………….………………….…….23

Figure14: Design of the Zheng et al YOLOv3 variant ……………..…………………..……………………...23

Figure15: Design of the Badrinarayanan et al SegNet…………………………………………………………24

Figure16: Design of the Fang et al Tinier-YOLO ………………………………………………………………….24

Figure17: Examples of images generated by the GAN model …………………………….……………….28

Figure18: Accuracy and loss for training and validation for Qassim et al residual VGG16……31

Figure19: Accuracy and loss for training and validation for Zheng et al YOLOv3 variant………33

Figure20: Accuracy and loss for training and validation for Badrinarayanan et al SegNet……35

Figure21: Accuracy and loss for training and validation for Fang et al Tinier-YOLO………………37

Figure22: Accuracy and loss for training and validation for Tinier-YOLO with data augmentation.…38

**LIST OF TABLES**

Table1: The performance of Qassim et al residual VGG16..………………………………………………..31

Table 2: The performance of Zheng et al YOLOv3 variant…………….…………………………………….33

Table 3: The performance of Fang et al Tinier-YOLO ………………………………………………………….36

Table 4: The performance of Tinier-YOLO with data augmentation …………………………………..38

Table 5: Comparing the accuracies and losses of all the models in this project ………………….39

**1. Introduction**

With the rise of insurgencies and terrorist organizations, the combatants who participate in those instances of armed conflict rely on irregular warfare tactics for survival or to win against extreme odds. Those tactics could increase the chances of survival to fight another day or their distraction. On the battlefield, non-combatants could be caught in the crossfire, and many innocent lives could be lost. Those organizations fight against standing armies. Those armies use standard equipment and uniforms to equip their soldiers. While at the same, depending on the funding, the organizations participating in irregular warfare use outdated equipment, civilian clothing, and some armored vehicles. Because of this use of civilian clothing for those organizations, it is difficult in some situations for soldiers on the ground to distinguish between enemy combatants and civilians, especially in urban environments. Even in worse situations, the enemy uses capture equipment, making it difficult to distinguish between friend and foe in the heat of the battlefield.

The process of target acquisition takes a few seconds, which could be the difference between a person living and a life lost. This process was typically made by humans, who decide which person is a friend and which is a foe.

With the development of machine learning and in the last decade or so of deep learning, the decisions started to swift from persons to machine learning algorithms. This allows decreasing the chances of collateral casualties on the battlefield. Deep learning models and techniques could help prevent civilian casualties and friendly fire by training a deep learning model to recognize combatants who participate in urban irregular warfare. For this to work, an image dataset is required to train the deep learning model. The difficulty of procuring such a dataset is that the images of such situations are limited and far in-between, not enough to train a deep learning model.

However, there is a solution to this problem. Furthermore, that solution is computer games. Since computer games have become more and more realistic, real-life situations can be replicated when playing computer games. Also, by recording in-game footage, thousands of images are available to compose an image dataset for training deep learning models. The average computer game runs at 60 frames per second, depending on the player's hardware. Sometimes computer games run at a higher rate.

However, for this study, a smaller image dataset was chosen. Furthermore, the dataset's author used data augmentation to increase the dataset. The reason for choosing this type of image dataset was because the study aims to look for deep learning techniques and models which would solve the problem of identifying the combatants who participate in urban irregular combat no matter the image dataset.

**2. Acknowledgment**

This project could not be done without the help and support of the people who I interacted with throughout the years of study in the university. Firstly, and most importantly, I must thank my family who helped and support me throughout the years. Their presence was felt even when there was an ocean between us. Though my best and worst days, they were still there to push me towards being the best version of myself. Secondly, I want to thank the lecturers who guided me on the right path academically and professionally. Thirdly, I want to the thank my friend for making my free time as fun as possible. Fourthly, I want to thank my course mates who made the lectures more interesting. And finally, I want to thank the airsoft community for creating an environment in which I could relax and organize my thoughts, while at the same time improving my physical health.

**3. Literature review**

This chapter covers a few topics relevant to the issues addressed in this thesis, such as military camouflage, irregular warfare in urban environments, computer vision in computer games, and deep learning techniques.

This study's primary image source comes from computer games because video games nowadays are inspired and can simulate reality. However, it is essential to ensure that in the literature, there were studies using in-game footage as data to train deep learning architectures. This collection method does not refer to using in-game footage in studying computer games related to a computer vision problem. It refers to using this method to study problems that are entirely unrelated to computer games. In the literature, the authors use game footage to test the performance of deep learning algorithms in simple tasks such as object identification. In methods very hungry for data like deep learning (Mahendran, et al., 2016), computer games are a good source of easy-to-obtain images to be used in image databases. However, there could be problems in using those images to solve real-life problems because synthetically generated data are not like authentic world images. (Shafaei, et al., 2016) Those concerns could be valid when using computer games in which the graphics are not detailed, or the hardware used to run those computer games are not powerful enough. However, in the right conditions, newer computer games developed by a video game company with a high budget, run on hardware with enough power, could produce high-quality in-game footage which looks like real life. (Richter, et al., 2016)

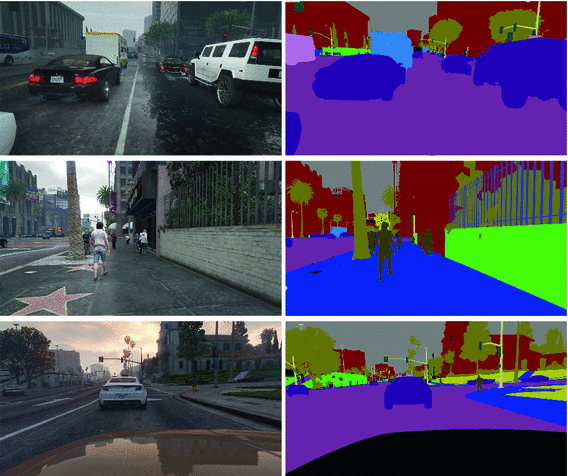


Fig1: Deep learning model identifying object in images from computer games

With the possible main source of images for study reviewed, the next step in this chapter is to review the literature with covers the equipment and uniforms of the combatants which participate in urban irregular fighting and which are the source of inspiration for the computer games characters. Military camouflage is considered a fashionable arms race between the militaries of different nations. The scope of the camouflage is to blend with the environment, thus saving lives in the process. Each country has its unique camouflage patterns and designs. The soldiers must recognize each other, thus limiting friendly fire. The patterns and designs are based on a country's military's culture and the terrain that the country's military would likely fight on. (Talas, et al., 2017) Smaller countries with small military budgets keep using decades-old uniforms. The worst-case scenario when it comes to old uniforms is if the states enter a war with each other, and they were part of the same country (e.g., URSS or Yugoslavia), there is an increased chance of instances of friendly fire. One example of this scenario happening in the current war in Ukraine. Since both countries were part of the URSS, both countries' militaries still use in limited amount Soviet-era or Russian-made weapons and equipment, including uniforms with the same camouflage patterns. (Barret, n.d.)

A picture containing ground, person, outdoor, military vehicle

Description automatically generated

Fig2: Ukrainian soldier with a yellow armband

Larger militaries, with soldiers worldwide, have different uniforms for different theaters of war. Although the uniforms used today by the larger militaries used classical patterns and designs, some countries and branches started adopting digitalized camouflage patterns. The term digitalized refers to the way the patterns were designed. Before, the patterns were created ether by hand with a paint and brush or by mechanical machines for mass production. Those patterns are created with use of line and dots to recreate natural colors and patterns. In the case of digitalized camouflage pattern, those were created, or in some cases generated, with computer software, thus looking more pixelated from a short distance, but blending better with the environment from a distance. Those patterns were designed based on deep learning. (Xiao, et al., 2020) In the context on this study, the digitalized modern camouflages like classical patterns, must be taken into consideration, because in computer games, the camouflage patterns look pixelated when the resolution is low, or the computer game characters are located far from the player’s camera. In the literature on this topic, the recent papers primarily focuses on the smaller countries or the history of military camouflage because larger militaries already updated their uniforms about a decade ago.

Graphical user interface

Description automatically generated

Graphical user interface

Description automatically generated

Fig3: Digitalized camouflage patterns in action

The camouflage patterns were designed to conceal the soldier in an environment. Most battles were fought in the natural environment, hence the tendency to use natural colors such as shades of greens, grays, or browns. However, developing a camouflage pattern and design for the artificial environment (i.e., urban environment) comes with different problems. Each city is unique in its colors, so making a camouflage pattern that works everywhere is complicated.  Toet and Hogervorst (Toet & Hogervorst, 2013) tried experiments with a possible urban camouflage. The colors of choice are like the traditional patterns, but the designs are entirely different. The camouflage they designed only work for a small town or suburban areas, where the construction materials are produced locally. The camouflage design is useless when going to the downtown area of a large metropolis, especially in Western countries. These areas are made of steel, concrete, and glass, with no natural colors around. The camouflage uniforms help the soldiers defensively; however, they could be inconvenient when used offensively, especially in irregular warfare. The uniform is easily identifiable in an urban environment, while the insurgents they encounter in those situations are not. As the world gets more urbanized, the tactics to counter irregular warfare will focus more on the urban environment. (Bojor, 2019)

Graphical user interface

Description automatically generated

Fig4: Examples of urban camouflage

Although irregular warfare can be encountered in any environment, military thinkers have suggested that the urban environment is the most difficult due to the inconvenience of using heavy equipment such as artillery and tanks. The use of heavy equipment could increase the chances of civilian casualties, and insurgents use that to their advantage by hiding in the populace. (Vacca & Davidson, 2011) With increased intelligence from recognizance and the use of special forces trained to mimic irregular warfare tactics (Kitzen, 2020) civilian casualties decreased. One example of a resistance organization used captured enemy equipment to equip its own combatants, was the Warsaw Uprising. In this case, this Polish resistance capture German equipment including uniforms, helmets, and weapons. To recognize each other during the intense urban fighting, the Poles used red and white ( the colors of their national flag) armbands. (Anon., 2020)

A picture containing building, outdoor, stone

Description automatically generated

Fig5: Polish resistance fighters during the Warsaw Uprising

The next logical step in this chapter is to understand the deep learning techniques which could be used to solve the research problems of this study. A good way to do that is to investigate deep learning literature and decide which techniques are useful. In the literature, deep learning is defined in different ways depending on the description of the structure. Zhang et al (Zhang, et al., 2018) looked over and discussed different definition of deep learning. The unified definition which resulted from the study says that deep learning is a “process not only to learn the relation among two or more variables but also the knowledge that governs the relation as well as the knowledge that makes sense of the relation.” (Zhang, et al., 2018) In the context of this study it refers to the concept creating a multilayer neural network learning algorithm which could be used in image recognition problems. (Wu & Chen, 2015) This concept of using deep learning in solving problems with the use of images as training materials describes the basis of a term called ‘computer vision’. Computer vision refers to the ability of a machine or computer to see the world around it. Like a person who uses vision to learn about the world, a computer could use vision to train and learn to have better decisions later. (Learned-Miller, 2011) The image recognition problems are based on a classic type of machine learning problems called classification problems. In this type of problems, the data which is used as the input of a machine learning algorithm are classified based on labels. (Manna, 2022) When the data used in training machine learning algorithms is labeled like in classification problem, it is considered supervised. The name comes to the fact that the data was labeled before a machine learning algorithm was train on it. I has been shown in the literature that deep learning model had great success with large-scale supervised training data (Lu, et al., 2021)

One type of deep learning model commonly used in image identification problems is Convolutional Neural Network, or CNN. CNN are effective “at the task of image classifying”. (Islam, et al., 2018) As the name suggests, the layers used in this deep learning model are convolutional layers. In a typical CNN model, the convolutional layers are used in combination with pooling layers and fully connected layers. The convolutional layer is based on the mathematical operation called convolution. Which means that CNN is a neural network “ that use convolution in place of general matrix multiplication in at least one of their layers ” (Goodfellow, et al., 2016, p. 326) Due to the fact that CNN is used primarily in image classification problems, this type of deep learning algorithm are used as solution for many research problem. One such research problem used was face recognition. Wang et al (Wang, et al., 2014) used CNN implementation in face recognition. The model resulted in 0.92 accuracy. However, the image dataset used for the training of the model was detailed, with the main features of the faces easily recognizable in the images and no instances of overfitting. Another situation in which CNN was used in its basic form is food recognition. Islam et al (Islam, et al., 2018) proposed a deep learning model which used a dataset of 16643 images in eleven food categories. The model had a validation accuracy of 0.92. However, the difference between the training and validation accuracy was significant. Which that the deep learning model encountered problems with overfitting.

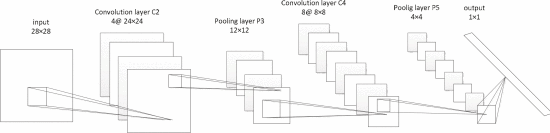


Fig6: Typical CNN model

The next and final in this literature review is to look over the deep learning model which were used in human and battlefield recognition. The two most common types of used in these deep learning problems were YOLO, or You Only Look Ones, and its versions, and encoder-decoder models. When used in battlefield scenarios, the most common purpose of the deep learning models is to identify between civilians and foes. (Miller, 2019) However, in some studies, the deep learning models were used to identify military equipment in some situation. For example, deep learning techniques were used to identify soldier and military equipment in low-resolution or infrared images. (Zheng, et al., 2019) Out of the two types of deep learning models, the most common used one is the YOLO with its versions. In the literature, YOLO and its versions was used primarily in mapping the areas of an image in which the object is located. YOLO is based on the VGG-16 model (Zheng et al., 2019) and has had a few versions over the years. In some studies, the authors took one of the versions and modified it to increase performance. Although the most common version of YOLO found in the literature is YOLOv3, other versions of it were used. For example, in one article, the author proposed a variant of the YOLOv2 and used for human detection from UAV. (Boudjit & Ramzan, 2022)

Diagram

Description automatically generated

Fig7: A YOLO variant identifying objects

One such model was suggested by Fang et al. called Tinier-YOLO. (Fang, et al., 2020) This model is an improved version of YOLOv3 with improved accuracy and real-time performance. This model works in constrained environments, where the hardware does not have the power to process complex deep learning architecture. Unlike other variants of YOLOv3 which look like VGG16 in composition, (Zheng, et al., 2019) the tinier-YOLO uses a different type of layer called ‘fire’ layer. This layer is a simple convolutional layer, but it is on its own, without being grouped with other convolutional layers, or pooling layer. The difference between the ‘fire’ layer and a common convolutional layer used in other CNN type model is that the ‘fire’ layer uses a 1x1 filter, while the common convolutional layer uses a 3x3 layer. This means that the ‘fire’ layer increases the performance of the deep learning model by reducing the number of parameters, which also decreases the memory used. Besides this type of layer, tinier-YOLO also has in composition an up-sampling layer Compared to well-known deep learning model, YOLO performs well.It was proven in the literature that YOLO has a lower error rate than GoogleNet, Faster RNN, and mska RNN. (Peng, et al., 2020) In the context of this study, YOLO has a deep learning model was used in other military related uses besides identifying humans and equipment. The YOLO model was used in the development of digitalized patterns of camouflage uniform which many countries use today. (Xiao, et al., 2020)

Figure 1. - 
Schematic of the YOLO model.


Fig8: How YOLO mapping an image to identify an object

Another deep learning model used in battlefield scenarios is the encoder-decoder type model.

This type of deep learning model was used in the literature primarily to distinguish the objects and the background. One study used this type of model to differentiate between soldiers and the background. (Morocho-Cayamcela & Lim, 2021) A common occurrence notices in the literature when it comes to the encoder-decoder deep learning models is that the authors used the background as a label in the binary classification problems. Like the YOLO and its versions, the encoder-decoder type model are based on the VGG16 model. These models are composed of two CNN models. One CNN has the role of the encoder, and the other has the decoder. The encoder downscales an image to a feature vector, while the decoder expands the compressed feature vector back to a definite matrix. (Morocho-Cayamcela & Lim, 2021) However, they also use up-sampling layers. These layers are placed instead of the pooling layer in the decoder part of the model. Unlike the YOLO model, the up-sampling layers are not used in moderation, which means it could increase the memory require to train such deep learning models. This type of model are not as common in the literature as the YOLO versions, mainly because of the memory requirement. However, they do not need a lot of images to train. An example of an encoder-decoder model is SegNet, proposed by Badrinarayanan et al. (Badrinarayanan, et al., 2017)

A picture containing text, fabric

Description automatically generated

Fig9: An encoder-decoder model differentiate an object from the background

Besides the models, some deep learning techniques could be used simultaneously to increase performance. One such technique is data augmentation. It modifies the images from the image dataset, thus increasing the number of examples. It also helps if a deep learning model encounters overfitting, decreases validation loss, (Abirami, et al., 2021) and increases accuracy in some situations. (Perez & Wang, 2017) The literature suggests many data augmentation techniques to choose from. However, it can come with the cost of decreased performance. Another deep learning technique that could be used is GAN or generative Adversarial Networks. It is used to generate fake images to balance the image dataset between the classes. (Abirami, et al., 2021) The GAN is made of two parts: the generator and the discriminator. The generator creates fake images but is convincing enough to look natural. The discriminator learns to distinguish between real and fake images. If the fake images pass the discriminator’s checks, the fake images become part of the image dataset.  A more advanced version of GAN is called Deep GAN. The difference between the basic model of the GAN and the Deep Gan is that the Deep GAN has more layers in its design, thus making it more complex. The problem with the GAN model is that the performance and the images generated by the model are dependable on the quality of the images. If the images do not show the objects which represent the labels property, the GAN would not generate proper images which could be used in increasing the image dataset.


Fig. 3.
 - 

(a) Original Image (b) Flipping (c) Cropping (d) Rotation (e) Noise Injection



Fig10: Classical data augmentation techniques

**4. Research problems**

The purpose of this study is to use deep learning techniques and models to identify combatants which participate in urban irregular warfare. To do that, different deep learning techniques have been used would be analyzed and tested to find the most effective way to solve this study’s problem. These models need to recognize the combatants in every scenario, no matter what, from them being fully visible and easy to recognize with the naked eye to hiding behind cover and when only a part of their body is visible, from being very close to the camera to cover the whole image, to them being visible from a distance, but not recognizing their small details.

Another possible hypothesis that we are also investigating, which was discovered during the review of the literature relates to the role data augmentation plays in image classification. Since the author of the image dataset used data augmentation in increasing the dataset, a hypothesis arose which puts into question the possibility of data augmentation negatively affecting the effectiveness of a deep learning model when trained with this study’s image dataset.

**5 Methodology**

As mentioned in the previous chapter, the main aim of this research is to solve the problem of identifying combatants in urban irregular warfare by using deep learning techniques i.e., classifying game actors as ‘friends’ or ‘foes’ relevant to the player’s team. Based on the literature review in Chapter 2, the deep learning techniques commonly used in identifying combatants and military equipment were the YOLO model, mostly the version 3, (Zheng, et al., 2019) and encoder-decoder models (transformers) (Morocho-Cayamcela & Lim, 2021). A couple of models presented in the literature would be replicated in this study to see which couple be more effective in solving this study’s problem with the image database used in this study.

**5.1 Dataset**

Before presenting and testing the models replicated from the literature, it is important to understand the dataset used in this study. The image dataset chosen for this dataset was collected form a website called roboflow.com. The dataset consists of 6540 images divided into two classes, with 3270 images per class, thus making it a balanced dataset. The classes are called ‘Standard’ and ‘Irregular’. The ‘Standard’ comes from the standard uniform used by soldiers in organized militaries. The ‘Irregular’ comes from the combatants who use irregular warfare tactics, and they usually wear unstandardized equipment and civilian clothing. These images are collected from gameplay footage of the computer game called Counter Strike Global Offensive and they are augmented by the author to increase the size of the dataset. Initially, the dataset was not labeled due to the fact the images contain elements of both classes. In case an image has both types of combatants, that image is would be part of the class of the combatant with the most predominant role in the image, are more combatants of type over the other, or the class of the combatant closest to the camera. There are two challenges a model could encounter when training with this image dataset. The first is that not two images are similar, except those augmented before by the author. This happens due to the fact when playing computer games, every single second on gameplay is unique and not every aspect of gameplay is influenced by the player. Hence why there are instances of two or more combatants in the same image. The second is the fact that the combatants may not show entirely on the image. They could be found in different situations. For example, they could be found hiding behind cover, running away, jumping, fighting each other etc. The images would be split between training and validation datasets. For this project, the split would be 80/20, with the batch size of 32. Throughout the program, the images would have the size 180x180.



Fig11: Characters representing the classes and examples of data augmentation in the image dataset



Fig12: Examples of situations in which characters could be found in

**5.2 GAN**

A good way to increase the number of images in an image dataset is GAN or generative Adversarial Networks. It has the role of generating images based on the images found in the image database. As presented in the literature review, GAN is composed of two parts: generator and discriminator. (Zhou, et al., 2020) The architecture behind the GAN is like a CNN model. Although the images from the dataset could be created by playing a computer game, using GAN would reduce the time of creating the images. In the case of images from gameplay, a researcher must play computer games for a couple of hours to increase the possibility of replicating specific in-game events for the dataset. After that the gameplay footage must be edited to extract the images. And then finally, it is important to label the images. In the case of GAN, the program would run the deep learning model and it would create the images. A model like this would be used in this study to increase the size of the image dataset. The GAN model used would the basic one to see if it is possible to create new images with this study’s image dataset. As explained earlier in this chapter, the image dataset is a difficult dataset to work with due to the representation of the classes in the images. Not only that, but the images would also be compressed to 180x180 size images, which in some situations, the representation of the classes in the images would have difficulties to be recognized by deep learning models. After testing to see if using basic GAN is feasible, Deep GAN (Abirami, et al., 2021) could be used to generated more realistic images to increase the dataset.

**5.3 Preprocessing**

Before using the training dataset in training of a deep learning model, it is important to do some preprocessing to configurate the dataset, thus increasing the performance of the models. The TensorFlow library, the library from which keras library comes from, offers some function which deals with these steps in the methodology process. Two methods are used when loading data to the models. The tf.data.Dataset.prefetch() (tf is short for TensroFlow) is a function which overlaps data preprocessing and model execution during the training of the deep learning model. It is done by decoupling “the time when data is produced from the time when data is consumed. [...] The number of elements to prefetch should be equal to (or possibly greater than) the number of batches consumed by a single training step. “ (Anon., 2022) The value can be manually tuned, or it could be set by the tf.data.AUTOTUNE. The second method used in configurate the dataset is tf.data.Dataset.cache(). This function helps in keeping the images in memory after they were loaded off disk during the first epoch in the training of a deep learning model. Besides keeping the images in the memory, this function also help in preventing the dataset to become a bottleneck during the training of a deep learning model. The next step in the preprocessing process is to standardize the data. The images are RGB images because they are colored images. And RGB images have a RGB channel when saved in a tensor and has the values between 0 and 255. For a neural network like the deep learning models used in this study, these values are not ideal because it requires memory to process, which means the input to be reduced to increase performance. To standardize the values, they should be reduced to values between 0 and 1. This can be done by using a rescaling layer in the models.

**5.4 Replicated models from literature**

The methodology for this project would involve choosing deep learning models from the literature and then train and test them with the image dataset presented earlier. The models chosen for the testing should represent every type of deep learning architectures discovered in the literature. The types of deep learning models found the literature are YOLOv3 (Boudjit & Ramzan, 2022; Fang, et al., 2020; Sumit, et al., 2020; Xiao, et al., 2020; Zheng, et al., 2019; Boudjit & Ramzan, 2022; Zhang, et al., 2020) and variants of it and encoder-decoder (Morocho-Cayamcela & Lim, 2021) such as SegNet. (Badrinarayanan, et al., 2015; Badrinarayanan, et al., 2017)

Four model would be tested in total. A VGG16 variant would be tested first due to the fact YOLO and all its versions and variants (Zheng, et al., 2019) and all the encoder-decoder models are based on VGG16. (Morocho-Cayamcela & Lim, 2021)The VGG16 model chosen from the literature for testing is Quassim et al residual VGG16. (Qassim, et al., 2018) The second model which would be tested is a YOLOv3 variant proposed by Zheng et al. (Zheng, et al., 2019) This model was chosen due to the fact that this model was used in identifying soldiers and military vehicles on the battlefield during daytime and nighttime. During nighttime, the researchers used infrared low-resolution images to test the limits of the deep learning model. The third model is a encoder-decoder model named SegNet proposed by Badrinarayanan et al. (Badrinarayanan, et al., 2017) This model was used in segmenting images to identyfy the small representation of the classes on the images. The fourth and final model which would be tested in this study is different varient of YOLO. This model is called Tinier-YOLO and it was proposed by Fang et al. (Fang, et al., 2020) This model was chosen due to the fact it was designed to use less paramaters than typical YOLOv3 variants to increase effectiveness of the model and reduce the memory required to train the model.

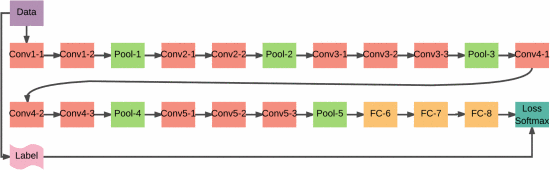


Fig13: Design of the Qassim et al residual VGG16

Table 2- 
Proposed Network Structure


Fig14: Design of the Zheng et al YOLOv3 variant

Fig. 2. - 
An illustration of the SegNet architecture. There are no fully connected layers and hence it is only convolutional.
 A decoder upsamples its input using the transferred pool indices from its encoder to produce a sparse feature map(s).
 It then performs convolution with a trainable filter bank to densify the feature map. The final decoder output feature
 maps are fed to a soft-max classifier for pixel-wise classification.


Fig15: Design of the Badrinarayanan et al SegNet

Diagram

Description automatically generated

Fig16: Design of the Fang et al Tinier-YOLO

**5.5 Effectiveness metrics**

The testing of the model would show how effective those deep learning architecture discovered in the literature are with this study’s image dataset an d which one works best with it. The training session for each model would be 50 epochs. The effectiveness of the models would be represented primarly by training and validation accuracy of the models. However, other metrics could be used such as precision, recall and f1 could be used to show the effectiveness of the models. In a binary classification problem like this study, the metrics which measure the efectiveness of a deep learning model are based on four values: true posivitive (TP), true negative (TN), false positive (FP) and false negative (FN). True positive refers to the number of predictions where the classifier correctly predicts an image as being part of a specific class. True negative refers to the number of predictions where the classifier correctly predicts an image as not being part of a specific class. False positive refers to the number of predictions where the classifier incorrectly predicts an image as being part of a specific class. False negative refers to the number of predictions where the classifier incorrectly predicts an image as not being part of a specific class. The accuracy, which is the main measurement of a deep learning model’s effectiveness, is the result dividing the total amount of samples correctly classified by the classifier to the total amount of samples, or (TP+TN)/(TP+TN+FP+FN). The precision is a metric which measures how much the positive pretictions is actually true. This is calculated by dividing the true positives to the total amount of positive predictions, or TP/(TP+FP). Recall is a metric which measures how much of the correctly predicted by the classifier are predicted as positive. It is calculated by dividing the true postives to the total amount of samples correctly classified by the classifier, or TP/(TP+FN). The F1 score is a metric which combines precision and recall into a single measurement. It is mathematically defined as the harmonic mean of precition and recall and it is calculated as follows:

During the calcutaltion of the mesurements of a deep learning model, three types of results could be calculated which combines the metrics of all the classes in the binary classification proble. Those are micro metrics, macro metrics and weighted metrics. The micro metrics is calculated by considering the total TP, TN, FP and FN of the model. By using micro metrics, the end values do not consider each class invididually due to the fact that the metrics are calculated globally. The macro metrics calculate the metrics for each class individually and then take the unweighted mean of the measurements. Lastly, the weighted metrics are opposite to the macro metrics due to the fact that the y use the weighted mean of the measurements. The weights for each class are defined as the total number of samples of a specific class.

The optimizer of choice for complilling the models is adam, with sparse categorical crossentropy as the loss function for the models. Adam is a optimizer which is used when the computing enviroment does not have a lot of memory and it is usually suited for deep learning problems that have large quantities of data and parameters. (Kingma & Ba, 2014) Sparse categorical crossentropy loss function is a probabilistic loss function which process the crosentropy loss between the labels of the images and the predictions of the deep learning models.

**5.6 Data augmentation**

After testing the deep learning models, the results would show which model is the most effective with this study’s image dataset. The most effective model would be modifyed to have a deep learning technique implemented into it. That technique is called data augmentation. As mention in the literarture review, literarture review is used to increase the image dataset inside the deep learning model, while at the same time limit the chances of overfitting. Data augmentation would be used by implementing it in the most effective model to see if it would incresase the validation accuracy of the model. Although the author of the image dataset used data augmentation to increase the dataset, in this project, the role of data augmentation is to increase the validation accuracy and redus the chances of overfitting. The data augmentation techniques which woulb be used in the program of this study are the basic ones such as image flipping, image rotation and random zoom on the image. Besides a data augmentation sequential model, a dropout layer would be implemented in the model. Dropout layers and data augmentation work together and gives data augmentation the ability to reduce the chances of overfitting. (Taylor & Nitschke, 2018)

**6. Resources**

The project was developed on the personal laptop. This laptop has Intel(R) Core (TM) i7-9750H CPU @ 2.60GHz with an Nvidia GeForce RTX 2080 GPU. For software, the code for the project was programmed on Google Collab, with Python as the programming language of choice.

**7.Results**

The purpose of this chapter is to look over the program of this study and analyze the results of different deep learning techniques and models used in recognizing combatants who participate in urban irregular warfare. As presented in previous chapters, the image dataset used for this study is comprised of 6540 images divided in two classes, with 3270 images per class.

Firstly, a GAN model was used to see if images could be created to increase the size of the image dataset. The model is divided into two parts, a generator, and a discriminator. As the literature review presented, the generator has the role of generating images by using images from the image dataset as templates. Furthermore, the discriminator has the role of looking over the generated images and decide if the image looks real and not. The images used in the training of the GAN model was set at 180x180 size. The dataset is split into training and testing dataset for the model by using 80/20 split. With the training dataset, the GAN model was train and generated images with technically could be used in increasing the dataset. For this model, the batch size was set to 32, with 100 epochs. The GAN model chosen was the basic GAN model to see if it is possible to generate with this study’s image dataset. Both generator and discriminator are sequential CNN models, which means they can be compressed into layers for a bigger model. After the GAN model is trained, images are generated based on the images in this study’s image dataset.

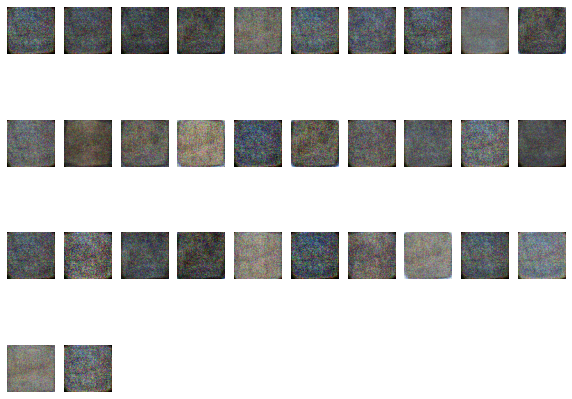


Fig17: Examples of images generated by the GAN model

The images shown above are the generated images after the basic GAN model has been trained. These images are pixelated, colorful or very clouded. A possible explanation for those results is the type of images the GAN must work with. The images in the dataset have a unique appearance. Unlike other image datasets, this dataset has images created ‘naturally’ though gameplay. The images are unique to the fact when playing computer games every second is unique. However, some images are similar since the author of the dataset used data augmentation techniques to increase the dataset. As shown above, the images are quite diverse. The characters which represent the classes are shown in different situations. The characters can be seen being far away from the player’s in-game camera, thus making them small and hard to notice. Sometimes, they can be seen close to the in-game camera, thus making them big. However, there are situations where only a part of the characters are shown. In some situations, two or more characters from the same class could be noticed in an image, in others two or character from both classes could be seen in the same image. Those characteristics of the images could be the main reason why the discriminator believes that the generated images by the generator part of the GAN model are real images, and they could be used to increase this study’s image dataset. Some colors used by the GAN model do show some a tentative to recreate the background of the images. However, the result do not look like the original images. Thus, it is concluded that GAN did not bring positive result and the image dataset would not be increased. An advance version of the GAN model named Deep GAN. However, with the poor results of the basic GAN model in mind, Deep GAN was not used in this study.

After trying using the GAN model, the step in the program was to test the effectiveness of the models from the literature. When developing the program for this study, the models coded were like the models from the literature, but not identical due to some limitations which will be explained in a future chapter. The first model from the literature imitated in the program is a residual VGG16 model proposed by Qassim et al. (Qassim, et al., 2018) The main reason for choosing this model is due to the fact the rest of the model which will be covered in this chapter were based on the VGG16 model. This means it is resonable to start from the basic model and continue to more complicated models. As mentioned in a previous chapter, the model will be tested for effectivenes when trained with this study’s image dataset. The effectiveness will be measured based primarily on the accuracy of the model. Besides the accuracy, other measurements included in the testing process are precision, recall and F1 score. Those three metrics will be shown into three different types. Those type include micro, macro and weighted. Like the basic GAN model, the models reprlicated from the literature were trained and tested with images set at the sizze of 180x180. The batcg size was set at 32 and the training takes 100 epoch. Also, the training and validation split was set at 80/20.

After 100 epochs, the residual VGG16 model had a training accuracy of 0.49 with a training loss of 0.69. While for validation, it had an accuracy of 0.5 with a loss of 0.69. The very small difference between the accuracies of training and validation shows no sign of overfitting. The model supposedly have 16 layers as the name suggest. However, this model is a residual model which means some layers may have been included or modified in the original VGG166 to make it more effective. The model consists of 13 convolutional layers with ReLu activation function, 5 max pooling layers, one flatten layer, 3 fully connect layers and one SoftMax layer. The number of neurons in a convolution layer is increased with each group of layers. A group of layers in this model is composed of a few convolutional layers with a max pooling layer. In a group of layers, the number of neurons in the convolution layers is double the number of neurons set in the convolutional layers which are part the previous group of layers until it reaches 1024. In the first group of layers which is composed of two convolutional layers and a max pooling layer, each convolutional layer has 64 neurons. In the second group of layers which is composed of two convolutional layers and a max pooling layer, each convolutional layer has 128 neurons. In the third group of layers which is composed of three convolutional layers and a max pooling layer, each convolutional layer has 256 neurons. . In the fourth group of layers which is composed of three convolutional layers and a max pooling layer, each convolutional layer has 512 neurons. . In the fifth and final group of layers which is composed of three convolutional layers and a max pooling layer, each convolutional layer has 1024 neurons. The flatten layer has the role of transforming the input data from a multi-dimensional matrix of features into a single column vector without effecting the batch size. This the data being a vector, it can be fed to the fully connected neural network classifier. Each fully connected layer is composed of two dense layers. The first dense layer has 1024 layer, while the second layer has the same number of neurons as the number of classes in which the images from the image dataset are divided into. In the case of this study, the number of classes is two.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Precision | Recall | F1 score |
| Micro | 0.50 | 0.50 | 0.50 |
| Macro | 0.25 | 0.50 | 0.33 |
| Weighted | 0.25 | 0.50 | 0.34 |

Table 1: The performance of Qassim et al residual VGG16

The recall is identical throughout the table, while also being equal to the validation accuracy. At a micro level, both precision and F1 score are equal to validation accuracy. However, at the individual class level, the precision and F1 score are very different. A possible reason for such results are the effectiveness of the model in recognizing the classes. The camouflage uniform of the characters which represent the ‘Standard’ class could have had a huge effect in the high numbers of false negatives and false positives.

Graphical user interface, text, application

Description automatically generated

Fig18: Accuracy and loss for training and validation for Qassim et al residual VGG16

As shown above, the training and validation accuracy remains in the same range at around 0.5 from almost the beginning throughout the model's training. However, the training accuracy had a sudden increase during the first few epochs, while at the same time the validation accuracy was nonexistence. In the case of the loss, the training and validation loss remains the same. After 80 epochs validation accuracy was equal or higher than training accuracy.

The second model replicated is a variant of the YOLOv3 proposed by Zheng et al. (Zheng, et al., 2019) After 100 epochs, the residual model had a training accuracy of 0.49 with a training loss of 0.69. While for validation, it had an accuracy of 0.5 with a loss of 0.69. Those result are identical with the residual VGG16 presented previously. Like the residual VGG16, the accuracies remained the same throughout the training of the model. The very small difference between the accuracies of training and validation shows no sign of overfitting. The model is variant of the YOLOv3 model. Since the YOLO model is based on the VGG16, that means that this model as well is based on the VGG16. The composition of this model is very different than the previous model. It is a CNN model composed of the same type of layers as the previous one, however the depth of this model is double. The number of neurons in a convolution layer is increased with each pair of layers. In each pair of convolutional layers, the number of neurons are different, with one layer having the double the number of neurons than the other. With each pair, the number of neurons is increased until it reaches 512 neurons. After that, the number is not increased. A few max pooling layers are used in the construction of the deep learning model. These are used once after every few pair of convolutional layers. The flatten layer has the role of transforming the input data from a multi-dimensional matrix of features into a single column vector without effecting the batch size. This the data being a vector, it can be fed to the fully connected neural network classifier. The fully connected layer is composed of two dense layers. The first dense layer has 1024 layer, while the second layer has the same number of neurons as the number of classes in which the images from the image dataset are divided into. In the case of this study, the number of classes is two.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Precision | Recall | F1 score |
| Micro | 0.50 | 0.50 | 0.50 |
| Macro | 0.25 | 0.50 | 0.33 |
| Weighted | 0.25 | 0.50 | 0.34 |

Table 2: The performance of Zheng et al YOLOv3 variant

The results shown in the table above are like the previous model.

Graphical user interface, text, application

Description automatically generated

Fig19: Accuracy and loss for training and validation for Zheng et al YOLOv3 variant

As shown above, the training and validation accuracy remains in the same range at around 0.5 from almost the beginning throughout the model's training. However, the training accuracy had a sudden increase during the first few epochs, while at the same time the validation accuracy was nonexistence. In the case of the loss, the training and validation loss remains the same. After 80 epochs validation accuracy was equal or higher than training accuracy.

The third model replicated is encoder-decoder model proposed by Badrinarayan et al. (Badrinarayanan, et al., 2017) called SegNet. After 100 epochs, the SegNet model had a training accuracy of 0.99 with a training loss of 0.0029. While for validation, it had an accuracy of 0.97 with a loss of 0.12. The very small difference between the accuracies of training and validation shows little to no sign of overfitting. This model is an encoder-decoder type deep learning model, which, as the name suggest, is composed of two components: an encoder and a decoder. In the encoder’s composition, the convolution layers are combined with batch normalization, max pooling and they use ReLu as their activation function. The decoder is like encoder, but instead of max pooling layers there are up sampling layers. In the previous models, the max pooling layers are located at the end of a group of layers. In the encoder, the up-sampling layers are the beginning of the groups of layers. The encoder’s composition is like Qassim et al residual VGG16 (Qassim, et al., 2018), however the authors used convolutional layer in combination with ReLu activation function and batch normalization. The encoder part of this deep learning model is composed of five groups of layers. The first two groups of layers are composed of two convolutional layers and one max pooling layer. The convolutional layers in the first group has 64 neurons, while the convolutional layers in the second group has 128 neurons. The last three groups of layers are composed of three convolutional layers and one max pooling layer. The convolutional layers in the first group have 512 neurons. In the second and third groups, the convolutional layers have 1024 neurons. The decoder part of this deep learning model is like the encoder. It mirrored the encoder and then flip the model. Unlike the encoder which has max pooling in its composition, the decoder has up sampling layers. The flatten layer has the role of transforming the input data from a multi-dimensional matrix of features into a single column vector without effecting the batch size.

A picture containing text

Description automatically generated

Fig20: Accuracy and loss for training and validation for Badrinarayanan et al SegNet

As shown in above, both the training and validation accuracies increase overtime. At 100 epochs, both accuracies are very close to 1. Throughout the training of the model, the validation accuracy has random downward spikes to around 0.5, while the training accuracy grows steadily. At 25 epochs, the training accuracy reaches approximately around its highest value. Those two behaviors are also reflected in the training and validation loss and in reverse. Instead of downward spikes, the validation loss has upward spikes. Besides the instances of spikes, the difference between the accuracies of the training and validation is very small, which shows little to no of instances for overfitting to occur.

The fourth and final model replicated is another variant of the YOLOv3 called Tinier-YOLO. This model proposed Fang et al. (Fang, et al., 2020) After 100 epochs, tinier-YOLO model had a training accuracy of 0.90 with a training loss of 0.1.39. While for validation, it had an accuracy of 0.97 with a loss of 0.1.61. The very small difference between the accuracies of training and validation shows no sign of overfitting. The model has similar type as the previous variant of YOLOv3. However, in this case, Fang et al used a combination of batch normalization with max pooling and convolution layers which has ReLu as their activation functions. There is no use of SoftMax layers nor connection layers. The first ten layers of this deep learning model is composed of five pairs of layers. Each pair is composed of one convolutional layer with ReLu activation function and batch normalization and a max pooling layer. The number of neurons in each convolutional layers are double with each layer used starting from 64 neurons and ending at 1024 neurons. After the five pairs of layers, three so called ‘fire’ layers (Fang, et al., 2020) are used. These fire layers are composed of two convolutional layers with ReLu activation function and 64 neurons. The next part of the deep learning model is composed of a combination of three convolutional layers with ReLu activation function and batch normalization , a fire layer and a up sampling layer. The first convolutional layer has 256 neurons, the second has 512 neurons and the third has 1024 neurons. The flatten layer has the role of transforming the input data from a multi-dimensional matrix of features into a single column vector without effecting the batch size.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Precision | Recall | F1 score |
| Micro | 0.97 | 0.97 | 0.97 |
| Macro | 0.65 | 0.65 | 0.65 |
| Weighted | 0.97 | 0.97 | 0.97 |

Table 3: The performance of Fang et al Tinier-YOLO

Most of the results in the table above are equal to the validation accuracy. The macro version of the measures calculates metrics for each class individually and then takes unweighted mean of the measures. This means that the false positive and false negatives for the individual classes are high.

Graphical user interface, text

Description automatically generated

Fig21: Accuracy and loss for training and validation for Fang et al Tinier-YOLO

As shown above, both the training and validation accuracies increase overtime. At 100 epochs, both accuracies are very close to 1. Throughout the training of the model, the validation accuracy has random downward spikes to around 0.70, while the training accuracy grows steadily. At 25 epochs, the training accuracy reaches approximately around its highest value. Those two behaviors are also reflected in the training and validation loss and in reverse. Instead of downward spikes, the validation loss has upward spikes. Besides the instances of spikes, the difference between the accuracies of the training and validation is very small, which shows little to no of instances for overfitting to occur

The next model replicated is a variant of the YOLOv3 called Tinier-YOLO which was presented previously. This model is tested with a deep learning technique included in its composition called data augmentation. The model is used to challenge the hypothesis of using data augmentation when the author of the image dataset already used this deep learning learning could modify the effectiveness of a deep learning model. After 100 epochs, the model had a training accuracy of 0.74 with a training loss of 3.49. While for validation, it had an accuracy of 0.90 with a loss of 1.66. The very small difference between the accuracies of training and validation shows no sign of overfitting. The model is identical with tinier-YOLO. However, a dropout layer was included in deep learning due to the fact it is recommended to have it when using a data augmentation as a deep learning technique. (Khosla & Saini, 2020) The dropout layer has the role to set the frequency of rate at each step during training to prevent overfitting.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Precision | Recall | F1 score |
| Micro | 0.90 | 0.90 | 0.90 |
| Macro | 0.90 | 0.90 | 0.90 |
| Weighted | 0.90 | 0.90 | 0.90 |

Table 4: The performance of Tinier-YOLO with data augmentation

All the measurement in the table above are equal to the validation accuracy. With means that data augmentation help in decreasing the chances of encountering false positives and false positives.

A picture containing chart

Description automatically generated

Fig22: Accuracy and loss for training and validation for Tinier-YOLO with data augmentation

As shown in above, the training and validation accuracy increases over time. In this case, validation accuracy is higher than training accuracy. This in turn eliminates the possibility of encountering instances of overfitting in the model.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Training accuracy | Training loss | Validation accuracy | Validation loss |
| Qassim et al VGG16 | 0.69 | 0.49 | 0.50 | 0.69 |
| Zheng et al YOLOv3 variant | 0.69 | 0.49 | 0.50 | 0.69 |
| Badrinarayanan et al SegNet | 0.99 | 0.029 | 0.97 | 0.12 |
| Fang et al Tinier-YOLO | 0.99 | 1.61 | 0.97 | 1.61 |
| Tinier-YOLO with data augmentation | 0.74 | 3.49 | 0.90 | 1.66 |

Table 5: Comparing the accuracies and losses of all the models in this project

As shown in the table above, the VGG16 and the first YOLOv3 variant have the same results. Even though composition of the model are different with the YOLOv3 variant being bigger in depth in composition. Those two deep learning models are the only models which have SoftMax and fully connected layers. This means that the SoftMax layer may be the reason why the models have low validation accuracies. The only encoder-decoder deep learning model has the lowest training and validation losses. However, it is the only with does not have other measurements besides training accuracy because it uses multiple up sampling layer, thus increasing the memory needed to process the measurements. This model, alongside tinier-YOLO have the best accuracies. This may be because they use a combination of convolution layers with ReLu activation function with max pooling and batch normalization. The main reason for implementing data augmentation in the tinier-YOLO model was because it was one the most accurate model and the only one of those models which allows the possibility of calculating the rest of the measurement such as precision, recall and F1 score. This gives the opportunity to see if a deep learning technique like data augmentation influences those types of measurements. Although the validation accuracy is lower than the same model without data augmentation, all the measurements are equal to the validation accuracy. Using data augmentation on an already augmented image dataset decreased the chances of false positives and false negatives.

**8.Discussion and conclusion**

The encoder-decoder model was promising. They show potential in bringing high accuracy in detecting combatants. As discussed previously, it used a lot of memory due to the number of layers and parameters those layers produce. It would be useful in solving deep learning problems like this study’s problem if memory were not an issue and time was not of essence. This comes from the fact that the encoder-decoder model replicated from the literature took the longest to train, at least double the amount of time for the rest of the models to train. For example, tinier-YOLO took three times less time to train. The YOLO variants did show potential. However, the effectiveness of the models was influenced by how far they were in terms of composition from VGG16, the model which inspired YOLO in the first place. Tinier-YOLO shows that batch normalization in combination convolution layers could be used with great effect in designing deep learning model used in human and battlefield detection. GAN models could see potential in increasing image dataset. However, the image dataset for this study did not allow for the possibility of generating images. The solution to this problem could be editing the images in such a way that only the characters which represent the classes are shown clearly and only one character shown per image. However, although these modifications could help the GAN model in generating images, in the long run it would be difficult for the deep learning to recognize combatants shown in situation depicted in the images. This study shows which types of deep learning could be used in identifying combatants of urban irregular combat. Also, the images which forms the image dataset come from a computer games in-game footage. This means that this study could help in choosing the right deep learning model for developing and improving computer games AI. A computer games genre which could use such deep learning model to great effect is the first-person shooter genre, the type of genre in which Counter Strike Global Offensive is part of. Besides, it would incentivize future research in choosing in-game footage from computer games are the closest to reality as a source of images for image datasets. With computer games being more realistic, a deep learning model which works in solving real world could help computer games AI to be more realistic in the identification process.

**9.Limitations and future work**

Some limitations were encountered throughout the process of this study. The first limitation encountered was the type of images the Keras python library can work with. When the original image dataset was access, the images were jpg type images. The python works when using bmp type images. The images from the original dataset were converted from jpg to bmp.

The next limitation of this study encountered was the basic GAN model results. As mentioned in a previous chapter, the result of the GAN model could not be used in increasing the image dataset of this study. The images resulted from the GAN model were very pixelated, colorful, or cloudy. As mentioned in a previous chapter, an explanation for why the images generated by the GAN model looked like that is the way the characters which represents the classes are portraited in the images. The images in the dataset have a unique appearance. This dataset has images created ‘naturally’ though gameplay. The images are unique to the fact when playing computer games every second is unique. However, some images are similar since the author of the dataset used data augmentation techniques to increase the dataset. The characters which represent the classes are shown in different situations. The characters can be seen being far away from the player’s in-game camera, thus making them small and hard to notice. Sometimes, they can be seen close to the in-game camera, thus making them big. However, there are situations where only a part of the characters are shown. In some situations, two or more characters from the same class could be noticed in an image, in others two or character from both classes could be seen in the same image. Those characteristics of the images could be the main reason why the discriminator believes that the generated images by the generator part of the GAN model are real images, and they could be used to increase this study’s image dataset. Some colors used by the GAN model do show some a tentative to recreate the background of the images. However, the result do not look like the original images. Thus, it is concluded that GAN did not bring positive result and the image dataset would not be increased. An advance version of the GAN model named Deep GAN. However, with the poor results of the basic GAN model in mind, using Deep GAN or other more complexed versions of the GAN model were not taken into consideration when developing the program for this study.

Another limitation encounter during the development of the code is the programming of the models. The scope of the code was to imitate and test the deep learning algorithms from the literature to analyze and evaluate their effectiveness. The problem was that sometimes the models cannot be replicated, they do not work properly, or they do not have the chance to be analyzed in code without modifying their structure. Two types of layers caused obstacles in the development of the code. Those two are the up-sampling and the SoftMax layer. The up-sampling layers could be used to increase the effectiveness of a deep learning model when combined with convolution layers and batch normalization. However, when used to much, it could cause problems. For example, this type of layer is used extensively in the SegNet model proposed by Badrinarayanan et al (Badrinarayanan, et al., 2017). It increased the total number of parameters created by the number which in turn increased the duration of training for the model by doubling it compared to similar deep learning models. The second type of layer which caused limitations to the code was the SoftMax layer. Depending on the model and the number of parameters generated by it, the SoftMax layer could cause errors to the code. In the Keras, there are two ways to implement a SoftMax layer: a specialized SoftMax layer and a dense layer which uses SoftMax as an activation function. Most of the time, the errors occurred when using the specialized SoftMax layer. Those errors were mostly avoided when the dense layer which uses SoftMax as its activation function.

The last limitation encountered during the development of the program was the limitation of Google Collab. Its cloud servers has limited memory for each user depending on membership type. The models used in the program generates a lot of parameters which requires memory. Due to the limited memory on the Google cloud servers, the servers could not process the whole program in one run. That meant that for each run on the cloud servers, only one or two models could be trained. After the models are trained and the measurements were calculated, the run on the Google cloud servers was restarted to reset the memory allocation. This caused a little bit of delays in acquiring the measurements of the models, because it took multiple runs of the program to generate all the measurements However, Google Collab does allow users to save the results of last runs.

Some promising future work could be done based the lessons learned from this study. A proposed deep learning model could be developed specially designed to be effective on image datasets like the one used in this study. Although the encoder-decoder model had a lower training and validation losses than the tinier-YOLO model, the tinier-YOLO require less memory to train. The combination of convolutional layers with batch normalization gave promising results. A possible proposed model could be a variant of the YOLOv3. However, it had problems with the measurements on the individual class level. By using data augmentation, the measurements helped at the individual class level at the cost of validation accuracy. This may be due to the use of data augmentation techniques by the author of the dataset in the creation of the image dataset. In possible future work, this problem could be avoided by avoiding image databases which have not used data augmentation in increasing the dataset or do the extra work and create a specialized image dataset from in-game footage of the same computer game as the one used in the creation of this study’s dataset, or a different computer game which could be more realistic. After developing and testing a proposed model a good practical way to test it to its fullest effect is to implement it in a computer game. The computer game in which the model could be implemented should be the same computer game from which images for dataset was taken from. A way to implement the model is by developing a game mod for the computer game. That means using mod tools and have access to the computer game’s game engine for the implementation to work.

# References

Abirami, N. et al., 2021. Deep CNN and Deep GAN in Computational Visual Perception-Driven Image Analysis. *Complexity,* Volume 2021.

Anon., 2020. *30 Incredible Vintage Photos of Warsaw Uprising Have Been Brought to Life After Colorized.* [Online]   
Available at: https://www.vintag.es/2020/01/colorized-warsaw-uprising.html  
[Accessed 1 August 2022].

Anon., 2022. *Better performance with the tf.data API.* [Online]   
Available at: https://www.tensorflow.org/guide/data\_performance#prefetching  
[Accessed 7 August 2022].

Badrinarayanan, V., Kendall, A. & Cipolla, R., 2017. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 39(12), pp. 2481-2495.

Barret, M., n.d. *What happens if two sides of the enemy in a war was wearing the same camouflage uniform?.* [Online]   
Available at: https://www.quora.com/What-happens-if-two-sides-of-the-enemy-in-a-war-was-wearing-the-same-camouflage-uniform  
[Accessed 3 August 2022].

Bojor, L., 2019. Training Military Leaders for Urban Irregular Warfare. *International conference KNOWLEDGE-BASED ORGANIZATION,* 25(1), pp. 32-39.

Boudjit, K. & Ramzan, N., 2022. Human detection based on deep learning YOLO-v2 for real-time UAV applications. *Journal of Experimental \& Theoretical Artificial Intelligence,* 34(3), pp. 527-544.

Fang, W., Wang, L. & Ren, P., 2020. Tinier-YOLO: A Real-Time Object Detection Method for Constrained Environments. *IEEE Access,* Volume 8, pp. 1935-1944.

Goodfellow, I., Bengio, Y. & Courville, A., 2016. *Deep Learning.* s.l.:MIT Press.

Islam, M. T., Karim Siddique, B. N., Rahman, S. & Jabid, T., 2018. Image Recognition with Deep Learning. In: *2018 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS).* s.l.:s.n., pp. 106-110.

Khosla, C. & Saini, B. S., 2020. Enhancing Performance of Deep Learning Models with different Data Augmentation Techniques: A Survey. In: *2020 International Conference on Intelligent Engineering and Management (ICIEM).* s.l.:s.n., pp. 79-85.

Kingma, D. P. & Ba, J., 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

Kitzen, M., 2020. Operations in Irregular Warfare. *Handbook of Military Sciences,* pp. 1-21.

Learned-Miller, E. G., 2011. Introduction to computer vision. *University of Massachusetts, Amherst.*

Lu, N. et al., 2021. Binary classification from multiple unlabeled datasets via surrogate set classification. In: *International Conference on Machine Learning.* s.l.:s.n., pp. 7134-7144.

Manna, S. O., 2022. *DEEP LEARNING FOR BINARY CLASSIFICATION.* [Online]   
Available at: http://machinelearningbusiness.com/index.php/2022/03/16/deep-learning-for-binary-classification/  
[Accessed 10 August 2022].

Miller, I., 2019. *Computer Vision on the Battlefield: Can Machines Distinguish Between Enemy and Civilian in Military Urban Operations?.* [Online]   
Available at: https://papers.ssrn.com/sol3/papers.cfm?abstract\_id=3310951  
[Accessed 19 June 2022].

Morocho-Cayamcela, M. E. & Lim, W., 2021. Pattern recognition of soldier uniforms with dilated convolutions and a modified encoder-decoder neural network architecture. *Applied Artificial Intelligence,* 35(6), pp. 476-487.

Peng, H., Zhang, Y., Yang, S. & Song, B., 2020. Battlefield Image Situational Awareness Application Based on Deep Learning. *IEEE Intelligent Systems,* 35(1), pp. 36-43.

Perez, L. & Wang, J., 2017. The Effectiveness of Data Augmentation in Image Classification using. *CoRR,* Volume abs/1712.04621.

Qassim, H., Verma, A. & Feinzimer, D., 2018. Compressed residual-VGG16 CNN model for big data places image recognition. *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC),* pp. 169-175.

Richter, S. R., Vineet, V., Roth, S. & Koltun, V., 2016. Playing for Data: Ground Truth from Computer Games. In: B. Leibe, J. Matas, N. Sebe & M. Welling, eds. *Computer Vision -- ECCV 2016.* s.l.:Springer International Publishing, pp. 102-118.

Shafaei, A., Little, J. J. & Schmidt, M., 2016. Play and Learn: Using Video Games to Train Computer Vision Models. *CoRR,* Volume abs/1608.01745.

Talas, L., Baddeley, R. J. & Cuthill, I. C., 2017. *Cultural evolution of military camouflage.* [Online]   
Available at: https://royalsocietypublishing.org/doi/full/10.1098/rstb.2016.0351  
[Accessed 20 June 2022].

Taylor, L. & Nitschke, G., 2018. Improving Deep Learning with Generic Data Augmentation. In: *2018 IEEE Symposium Series on Computational Intelligence (SSCI).* s.l.:s.n., pp. 1542-1547.

Toet, A. & Hogervorst, M., 2013. Urban camouflage assessment through visual search and computational saliency. *Optical Engineering,* 52(4), p. 041103.

Vacca, W. A. & Davidson, M., 2011. The regularity of irregular warfare. *The US Army War College Quarterly: Parameters,* 41(1), p. 15.

Wang, W. et al., 2014. Face Recognition Based on Deep Learning. In: *International Conference on Human Centered Computing.* s.l.:s.n., pp. 812-820.

Wu, M. & Chen, L., 2015. Image recognition based on deep learning. In: *2015 Chinese Automation Congress (CAC).* s.l.:s.n., pp. 542-546.

Xiao, H. et al., 2020. Fast Self-Adaptive Digital Camouflage Design Method Based on Deep Learning. *Applied Sciences,* 10(15).

Zhang, W. J. et al., 2018. On Definition of Deep Learning. In: *2018 World Automation Congress (WAC).* s.l.:s.n., pp. 1-5.

Zheng, G., Wu, X., Hu, Y. & Liu, X., 2019. Object Detection for Low-resolution Infrared Image in Land Battlefield Based on Deep Learning. In: *2019 Chinese Control Conference (CCC).* s.l.:s.n., pp. 8649-8652.

Zhou, F. et al., 2020. Deep learning fault diagnosis method based on global optimization GAN for unbalanced data. *Knowledge-Based Systems,* Volume 187, p. 104837.