UAV-BASED POST DISASTER SURVIVOR DETECTION MECHANISM USING A GAN-AIDED ENSEMBLE NETWORK

A PROJECT REPORT

Submitted by

ABHISHEK MANOHARAN (2019503502)

NISHANTHINI S (2019503537)

VAMSI RAJU M (2019503568)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER TECHNOLOGY

ANNA UNIVERSITY, MIT CAMPUS

MAY 2023

BONAFIDE CERTIFICATE

Certified that this project report **"UAV-BASED POST DISASTER SURVIVOR DETECTION MECHANISM USING A GAN-AIDED ENSEMBLE NETWORK"** is the bonafide work of **ABHISHEK MANOHARAN (2019503502), NISHANTHINI S (2019503537), VAMSI RAJU M (2019503568)** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr P. JAYASHREE	Dr. R. GUNASEKARAN
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Professor and Head	Professor
Department Of Computer Technology	Department Of Computer Technology
Anna University, MIT Campus	Anna University, MIT Campus
Chromepet, Chennai-600044	Chromepet, Chennai-600044

ABSTRACT

Post-disaster scene understanding frameworks are becoming increasingly crucial in search and rescue operations and damage assessment initiatives. The use of Unmanned Aerial Vehicles (UAVs) provides an efficient method to complete the task of scene understanding. However, complex environments in post-disaster scenarios make it difficult for UAVs to accurately detect humans or objects. Moreover, inefficient object detection mechanisms lead to low accuracy and a long time for object detection tasks. Hence, to mitigate these issues, we propose a UAV-based GAN-aided Ensemble Network for efficient post-disaster survivor detection, involving a semantic segmentation mechanism. This approach deploys a Context-Conditional Generative Adversarial Network (CCGAN)-based model to remove occlusion in the images obtained from the UAVs. The framework classifies entities present in the visual scope of the UAV using a semantic segmentation framework deployed on the CCGAN-enhanced images, resulting in a pixel-level prediction of entities present in the post-disaster images. Furthermore, an ensemble network comprising a combination of single-stage and multi-stage detectors detects survivors present in the post-disaster scenario, thereby combining the benefits of both architectures, resulting in a reduced false negative rate and improved performance. The proposed model achieved a survivor detection accuracy of 96.4%.

ACKNOWLEDGMENT

We would like to record our sincere thanks to **Dr. J. PRAKASH,** Dean, Madras Institute of Technology, for having given consent to carry out the project work.

We wish to express our sincere appreciation and gratitude to **Dr. P. JAYASHREE,** Professor and Head, Department of Computer Technology, Madras Institute of Technology campus, Anna University.

With esteem respect, we express our sincere appreciation and gratitude to our project guide **Dr. R. GUNASEKARAN** Professor, Department of Computer Technology, MIT campus, Anna University, for his guidance, constant encouragement and support. His extensive vision and practical thinking has been a source of inspiration throughout this project.

We sincerely thank our Panel members, **Dr. P. JAYASHREE, Dr. R. KATHIROLI** and **Dr. V. P. JAYACHITRA,** Department of Computer Technology, Madras Institute of Technology campus, Anna University, for their kind support and suggestions to carry out the project work.

ABHISHEK MANOHARAN

NISHANTHINI S

VAMSI RAJU M

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
FN	False Negative
FP	False Positive
ML	Machine Learning
TN	True Negative
TP	True Positive
TPR	True-Positive Rate
UAV	Unmanned Aerial Vehicle
GAN	Generative Adversarial Network
CCGAN	Context-Conditional Generative Adversarial Network
YOLOv5	You Only Look Once version 5
RCNN	Region-based Convolutional Neural Network
IOU	Intersection Over Union
RPN	Region Proposal Network
NMS	Non-Maximum Suppression
SSD	Single-Shot Detector
mAP	Mean Average Precision
CUDA	Compute Unified Device Architecture

LIST OF SYMBOLS

SYMBOL	ABBREVIATION
L_G	Generator Loss
$\lambda_{L_{\{1\}}}$	L1 distance
$\lambda_{\{adv\}}$	GAN Loss function
$\Gamma_{\{rec\}}$	Reconstruction Loss
Θ	Network Parameters
L_{rpn}	RPN loss function
L_{obj}	Binary cross-entropy loss
L_{reg}	Smooth L1 loss
λ_{reg}	Relative importance hyperparameter
L_D	Discriminator Loss

CHAPTER 1

INTRODUCTION

1.1 POST-DISASTER SURVIVOR DETECTION

Natural disasters such as earthquakes, hurricanes, and floods can cause immense damage to infrastructure and human lives. In the aftermath of such disasters, identifying and rescuing survivors is of utmost importance, as it can significantly increase the chances of survival.

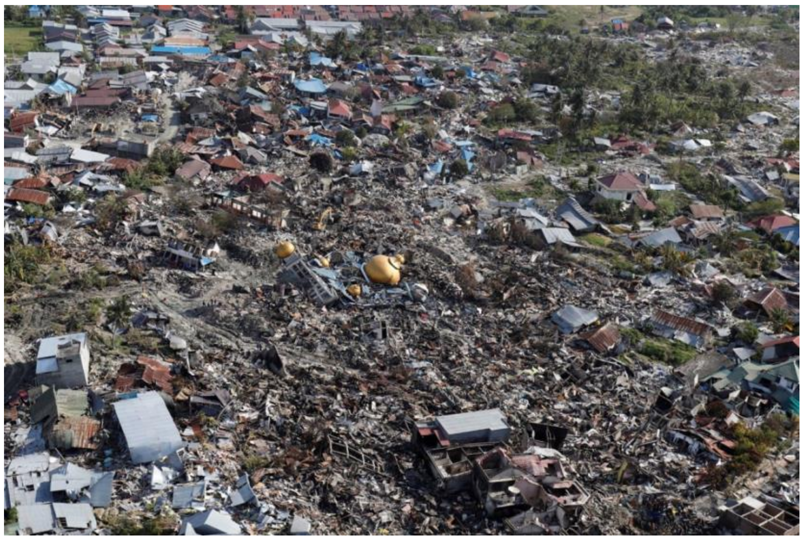


Fig 1.1 Post-Disaster Scene

Fig 1.1 depicts a post-disaster scene, wherein successful detection of survivors is a daunting task due to the small size of human survivors. Hence, post-disaster scene understanding frameworks are becoming increasingly crucial in search and rescue operations and damage assessment initiatives [1]. As the number of natural disasters continues to rise, the importance of efficient and accurate disaster response has become paramount [2]. Recent advancements in Unmanned Aerial Vehicles (UAVs) have shown great potential for post-disaster search and rescue operations. UAVs can rapidly cover large areas, provide real-time information, and reduce the risks to human rescuers. Hence, the use of Unmanned Aerial Vehicles (UAVs) provides an efficient and cost-effective method to complete the task of scene understanding [3]. Moreover, many deep learning strategies have been deployed to effectively execute visual detection from camera sensors mounted on UAVs [4].

However, existing methods of detecting survivors using UAVs can be limited by environmental conditions, such as low visibility, debris, and the presence of multiple objects that can make it difficult to differentiate between survivors and non-survivors. The traditional methods of search and rescue operations can be time-consuming, expensive, and dangerous for the rescuers, particularly in the case of large-scale disasters. The complex environments present in post-disaster scenarios make it difficult for UAVs to accurately detect humans or objects. Additionally, inefficient object detection mechanisms lead to low accuracy and a long inference time for object detection tasks, which can be particularly problematic in urgent search and rescue situations. Survivors being small objects in post-disaster UAV images makes the task of survivor detection using traditional techniques daunting [6]. Furthermore, survivors who are occluded in the image due to debris or damaged buildings covering them will make the survivor detection task further challenging.

1.2 UNMANNED AERIAL VEHICLES (UAV)

Unmanned Aerial Vehicles (UAVs), also known as drones, are remotely piloted aircraft that can be operated without an on-board human pilot. UAVs are typically controlled using a ground-based controller, and they can be equipped with a variety of sensors and payloads, including cameras, infrared sensors, and other specialized equipment.

Fig 1.2 Unmanned Aerial Vehicle

Fig 1.2 shows a UAV with a camera sensor mounted on it. UAVs have a wide range of applications, including military operations, surveillance and monitoring, aerial photography and videography, and search and rescue operations. In recent years, the use of UAVs for search and rescue operations has shown great potential, particularly in post-disaster scenarios where traditional search and rescue methods may be limited or ineffective. UAVs can rapidly cover large areas and provide real-time information, which can help identify survivors, assess damage, and provide situational awareness to rescue teams. Additionally, UAVs can reduce the risks to human rescuers by performing tasks that may be dangerous or inaccessible to humans, such as searching in hazardous areas or at high altitudes.

UAV technology is rapidly advancing, with new developments in sensors, communication systems, and autonomy. As a result, UAVs are becoming increasingly versatile and capable of performing a wide range of tasks. However, there are also challenges associated with UAV operations, such as regulatory requirements, safety concerns, and ethical considerations.

1.3 MACHINE LEARNING

Machine learning is an associated application of artificial intelligence that builds a mathematical model, using sample data which is known as training data to make predictions or decisions and provides systems the capability to learn automatically. However, the machine can learn without being programmed explicitly by any sources. The process gets started by feeding good quality data and training systems by building machine learning models using data and different algorithms. Every machine algorithm consists of three components, namely Representation, Evaluation, and Optimization. Supervised and Unsupervised algorithms are the two main categories in machine learning methods. Machine learning integrates several approaches such as combinatorial optimization, search, statistics, reinforcement learning, probability theory, logic, and control theory.

1.3.1 ENSEMBLE NETWORK

An ensemble network, also known as an ensemble model, is a technique in machine learning that involves combining multiple models to improve the overall accuracy and robustness of the predictions. An ensemble network is a type of neural network that uses ensemble learning to improve the performance of the model. The basic idea behind ensemble learning is that combining the predictions of multiple models can produce more accurate predictions than any individual model. This is because each model may have different strengths and weaknesses and may be better suited to different parts of the data. By combining multiple models, the ensemble network can leverage the strengths of each model to produce a more accurate and robust prediction. Fig 1.3 describes the process of ensembling.

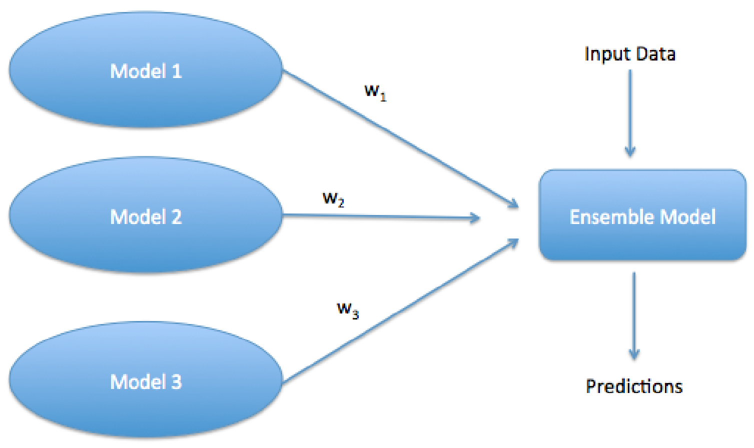


Fig 1.3 Ensemble Model

One popular method for ensembling is simple averaging, which involves taking the average of the predictions of multiple models. Simple averaging is a straightforward and effective way to combine the predictions of multiple models. In simple averaging, each model is trained independently on the same data, and then the predictions of each model are averaged together to form the final ensemble prediction. The advantage of simple averaging is that it is easy to implement and computationally efficient, making it a popular choice for ensemble learning. Additionally, simple averaging can reduce the impact of individual model biases and errors, which can help to improve the overall performance of the ensemble model.

1.4 DEEP LEARNING

Deep learning is a cutting-edge field in artificial intelligence that has the potential to transform many industries. At its core, deep learning involves training artificial neural networks to learn from large amounts of data. This allows the networks to make predictions, classify data, or make decisions with high accuracy. The key advantage of deep learning is its ability to learn complex representations from raw data. This eliminates the need for manual feature engineering, which is a time-consuming and error-prone process. Additionally, deep learning models can generalize well, meaning that they can make accurate predictions on new data that they have never seen before.

1.4.1 CONVOLUTIONAL NEURAL NETWORK (CNN)

Convolutional neural networks, or CNNs, are a type of artificial neural network that have revolutionized the field of computer vision. CNNs are feed-forward networks designed to automatically extract relevant features from raw input data, such as images, and use these features to make accurate predictions or classifications. The convolutional layer, pooling layer, and fully-connected layer are some of the layers of CNN. The convolutional layer is the first layer in any CNN architecture that extracts features from the image data. The feature map is obtained as the output from the convolutional layer, which is then passed as input to the pooling layer. The pooling layer performs dimensionality reduction, i.e., reduces the number of parameters in the feature map from the convolutional layer. Pooling helps to reduce the complexity and limits the risk of overfitting. Fully connected layers are the final layers of the network. The output from the final pooling or convolutional layer is flattened and then fed into the fully connected layer.

1.4.2 GENERATIVE ADVERSARIAL NETWORK (GAN)

GAN stands for Generative Adversarial Network, which is a type of artificial neural network that is composed of two parts: a generator and a discriminator. The generator generates synthetic data that aims to look like real data, while the discriminator is trained to distinguish between real and fake data.

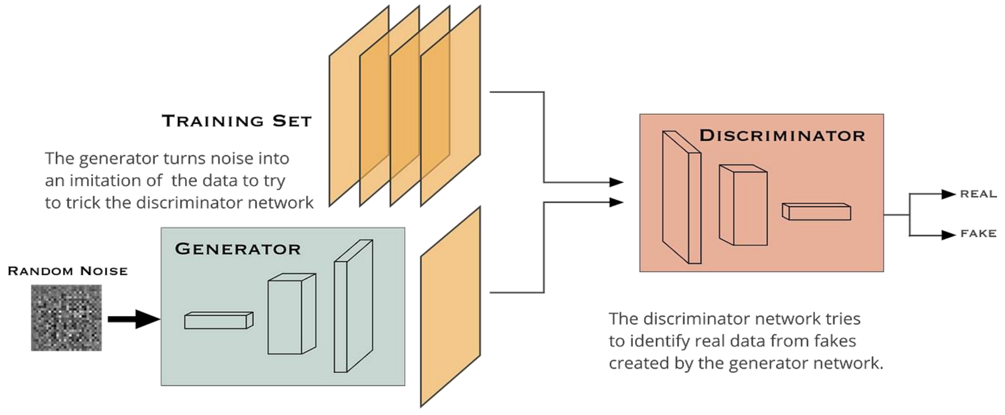

Training Set
The generator turns noise into an imitation of the data to try to trick the discriminator network
Random Noise
Generator
Discriminator
Real
Fake
The discriminator network tries to identify real data from fakes created by the generator network.

Fig 1.4 High-level architecture of GAN

Fig 1.4 depicts the high-level architecture of a GAN model. The training process of a GAN involves a competition between the generator and the discriminator. The generator tries to create synthetic data that can fool the discriminator into thinking it is real, while the discriminator tries to correctly classify the real and fake data. As the training progresses, both the generator and discriminator get better at their tasks, leading to the creation of high-quality synthetic data that can be used for various applications such as image and speech synthesis, data augmentation, and anomaly detection. GANs have been widely used in computer vision applications, particularly in image synthesis, where they can generate realistic images that are almost indistinguishable from real images.

1.4.3 SEMANTIC SEGMENTATION

Semantic segmentation is a computer vision technique that involves dividing an image into multiple segments or regions and assigning a class label to each segment based on its semantic meaning. The goal of semantic segmentation is to understand the content of an image at a pixel level, by identifying and labeling each object or region within the image. Unlike image classification, which assigns a single label to an entire image, semantic segmentation assigns labels to each pixel of an image. This fine-grained approach allows for a more detailed understanding of the image and can be used in various applications such as object detection, autonomous driving, and medical image analysis. Fig 1.5 depicts semantic segmentation executed on post-disaster images obtained from a UAV.

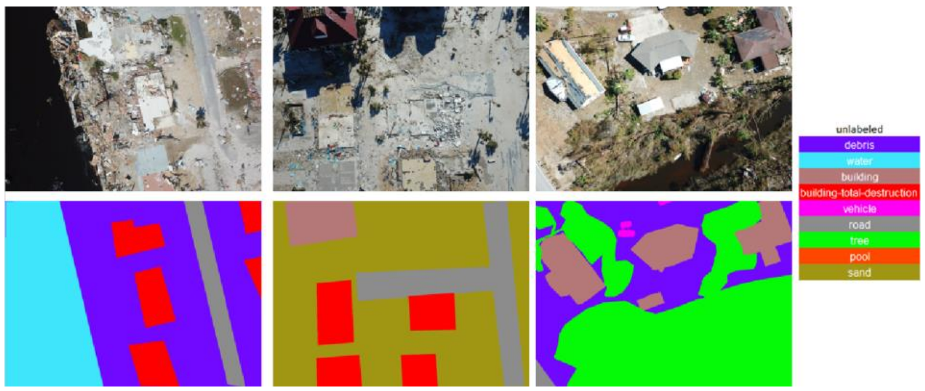


Fig 1.5 Semantic Segmentation

1.4.4 COMPUTER VISION

Computer vision is an interdisciplinary field that focuses on enabling computers to interpret and understand visual information from the world around us. It involves the use of various methods and techniques from artificial intelligence, machine learning, and computer science to enable machines to perform tasks that would normally require human-level visual perception. It enables computers and systems to derive meaningful information from digital images, videos and other visual inputs and take actions or make recommendations based on that information. Computer vision uses real-time images to create a 3D map. With the usage of 3D maps, autonomous vehicles can decipher the driving space for risk-free driving and also opt for alternate routes in case of projected collision. This makes driving easy and accident-free for its passengers.

1.5 PROBLEM STATEMENT

Post-disaster scene understanding frameworks are becoming increasingly crucial in search and rescue operations and damage assessment initiatives. Though UAVs provide an efficient means to detect survivors, complex environments present in post-disaster scenarios make it difficult for them to detect humans accurately. Moreover, inefficient object detection mechanisms lead to low accuracy and a long inference time. Hence, to mitigate these issues, we propose a UAV-based survivor detection scheme involving a GAN-aided semantic segmentation mechanism. This approach deploys a Generative Adversarial Network (GAN)-based model to remove occlusion in the images obtained from the UAVs. The framework classifies objects present in the visual scope of the UAV using a semantic segmentation framework, resulting in pixel-level prediction and classification of entities. Furthermore, an ensemble network consisting of a combination of single-stage and multi-stage detectors is used to improve the performance of the survivor detection model. This helps reduce the false negative rate and improve the overall accuracy of the system.

1.6 OBJECTIVE

- To develop an efficient post-disaster survivor detection framework using UAVs for efficient Search-And-Rescue (SAR) operations and deploy a semantic segmentation mechanism on the 3D model, leading to a pixel-level prediction of various entities or objects in the 3D model which will improve the detection of survivors present in the post-disaster scene.

- To deploy a Generative Adversarial Network (GAN) framework to improve the detection of survivors, who are generally present as small and dense objects in post-disaster UAV images. A GAN denoiser will result in images having lower occlusion and optimal brightness, thereby highlighting the important features of the object.
- To implement a hybrid single-stage and multi-stage ensemble network for survivor detection on the previously obtained semantic entities. The model will comprise the YOLOv5 and Faster R-CNN mechanisms to combine the benefits of both, thereby decreasing the high false negative rate of multi-stage mechanisms and improving the performance of single-stage detectors.

1.7 ORGANIZATION OF THESIS

Chapter 2 encompasses a survey of papers that are related to the topics discussed above. Chapter 3 describes the proposed system and outlines the architecture of the proposed work. The objective of each module along with the algorithms is explained. Chapter 4 explains the experimental implementation details of each module in the framework. Chapter 5 presents the results of the proposed system along with the visualization of the performances before and after optimization techniques. The result of the proposed model is tabulated along with the performance of other novel works. Chapter 6 outlines the conclusion of the thesis and the future work that can be done to enhance the framework.

CHAPTER 2

LITERATURE REVIEW

2.1 ENSEMBLE NETWORKS FOR OBJECT DETECTION

Post-disaster survivor detection is an important task in search and rescue operations. However, survivors being hard to detect objects, especially from sensors mounted on UAVs, require specialized techniques to be effectively detected. Various Convolutional Neural Network (CNN)-based models have been developed for efficient object detection.

With the goal of lowering the high false negative rate of multi-stage detectors and improving the quality of the single-stage detector proposals, [7], [8] propose different ensemble networks that combine a multi-stage method with a single-stage detector through ensembling for effective object detection. But according to the investigation, detecting objects in drone images is more challenging than detecting them in images that were taken from the ground. Hence, the accuracy of the model trained on UAV images is still low compared to models trained on ground images.

Several other deep learning techniques used for the detection of objects from UAVs have been studied in [9], wherein various deep learning architectures have been outlined, including generative adversarial networks, autoencoders, recurrent neural networks, and convolutional neural networks, and their contributions to the challenge of improving vehicle detection. However, videos captured in the UAVs are sent to on-ground workstations or to the cloud for processing rather than being implemented on the UAV itself, thereby leading to the absence of a lightweight system for efficient real-time detection.

2.2 GAN FOR OCCLUSION REMOVAL

GANs are being increasingly deployed in many modern detection algorithms due to their extensive applications, like the removal of occlusion present in images. [10] discusses the regeneration of images obtained from a UAV for effective disaster scene recovery and the detection of survivors. It involves the use of a Partial Convolutional GAN (PC-GAN) for the removal of occlusion through inpainting. Furthermore, the authors of [11] discuss denoising and occlusion-removal strategies empowered by GAN-based systems for better image recognition, wherein image regeneration with efficient removal of unnecessary entities present in the image leads to better object detection performance.

2.3 SEMANTIC SEGMENTATION

Semantic segmentation classifies all entities present in an object through pixel-level prediction. This enables the detection model to easily recognize objects present in an image. The authors of [12], [13] propose and evaluate self-attention segmentation models on new high-resolution datasets, namely HRUD and UAVid. However, HRUD is a very challenging dataset due to its variable-sized classes along with similar textures among different classes.

Debris, textures of debris, sand, and buildings with destruction damage make a great impact on the segmentation performance of the evaluated network models. Furthermore, [14], [15] discusses in detail various semantic segmentation frameworks used for entity separation and classification in UAV-driven images. 3D rendering of images is also being done in an extensive manner for better detection of objects in a scene. [16], [17] implement 3D imaging mechanisms using 2D images obtained from a swarm of UAVs, wherein a 3D imaging of a scene by the usage of 2D images obtained from several UAVs present in the swarm is implemented at different perspectives with a few points of overlap. But a considerable amount of data must be transmitted from the UAV swarm, as images obtained from each node in the swarm are used to produce the 3D rendering. Multiple UAVs also need to exchange information in order to efficiently collect data on the scenario.

2.4 DATASETS REVIEWED

Many UAV-driven and synthetic datasets have been generated for analyzing efficient methods for survivor detection using UAVs. [18] introduces a high-resolution post-disaster UAV dataset named RescueNet, which contains comprehensive pixel-level annotation of images for semantic segmentation to assess the damage and detect survivors after a natural disaster. However, smaller objects like “vehicles” and “pools” make it difficult to get a good segmentation compared to larger objects like buildings and roads.

[19] proposes the UAV-Human dataset for understanding human action, pose, and behavior. The proposed UAV-Human contains 67,428 multi-modal video sequences, 119 subjects for action recognition, 22,476 frames for pose estimation, 41,290 frames, 1,144 identities for person re-identification, and 22,263 frames for attribute recognition which encourages the exploration and deployment of various data-intensive learning models for UAV-based human behavior understanding. However, The UAV-Human dataset poses a limitation for attribute recognition because the dataset is captured over a relatively long period of time.

2.5 INFERENCE

UAVs provide a significant advantage of having a larger field of view while assessing disaster-it regions for survivors. However, existing survivor detection mechanisms have low performance while being used in real-time systems due to the need for sending huge amounts of data to and from ground stations. Moreover, current detection models either have low accuracy, thereby leading to poor survivor detection performance, or have very long inference time, making such mechanism unsuitable for real-time usage, as survivor detection is to be executed at a fast manner. Hence, the usage of an ensemble model combining single-stage and multi-stage networks provide means to improve survivor detection performance as well as have lower inference time, thereby making the system suitable for real-time use.

Another main issue faced by current mechanisms is the presence of occlusion or coverage of survivors present in images taken by a UAV. This poses a serious problem to the performance of the detection model. Hence, GANs seem to be suitable for the removal of occlusion present in the images, leading to better performance of the survivor detection model. Furthermore, semantic segmentation will lead to better detection of survivors through a color coding generated on the entities present in an image.

CHAPTER 3

UAV-BASED POST-DISASTER SURVIVOR DETECTION MECHANISM USING A GAN-AIDED ENSEMBLE NETWORK

3.1 INTRODUCTION

Natural disasters such as earthquakes, floods, and hurricanes can have devastating effects on human lives and infrastructure. In the aftermath of such disasters, search and rescue operations are essential for locating and rescuing survivors. Traditional methods of search and rescue are time-consuming and can be risky for rescue personnel. In recent years, unmanned aerial vehicles (UAVs) have emerged as a valuable tool for conducting search and rescue operations in disaster areas. UAVs can quickly and safely capture high-altitude images of disaster areas, allowing rescue teams to identify survivors and plan rescue operations more efficiently.

However, identifying survivors from high-altitude images captured by UAVs remains a challenge. The images can be noisy and cluttered, and survivors may be difficult to distinguish from other objects such as debris. In recent years, machine learning approaches such as deep neural networks have shown promising results in survivor detection from UAV images. However, these approaches require large amounts of training data, which may not be available for some disaster scenarios.

To address these challenges, a UAV-based post disaster survivor detection mechanism that uses an ensemble of single-stage and multi-stage detectors is proposed, aided by a Generative Adversarial Network (GAN), to identify survivors in disaster areas. The proposed mechanism uses a GAN to remove occlusion present in the images captured by the UAVs, which are then passed through a semantic segmentation model to classify entities present, thereby effectively training the ensemble network for survivor detection. The proposed mechanism is evaluated, and the results show that it outperforms existing approaches in terms of accuracy and robustness.

3.2 RESCUENET DATASET

RescueNet is a newly introduced dataset for the task of search and rescue in disaster scenarios. This dataset comprises 10,000 synthetic images and 100 real-world images, designed to provide a comprehensive training and evaluation platform for researchers working on computer vision and robotics applications in the domain of search and rescue.

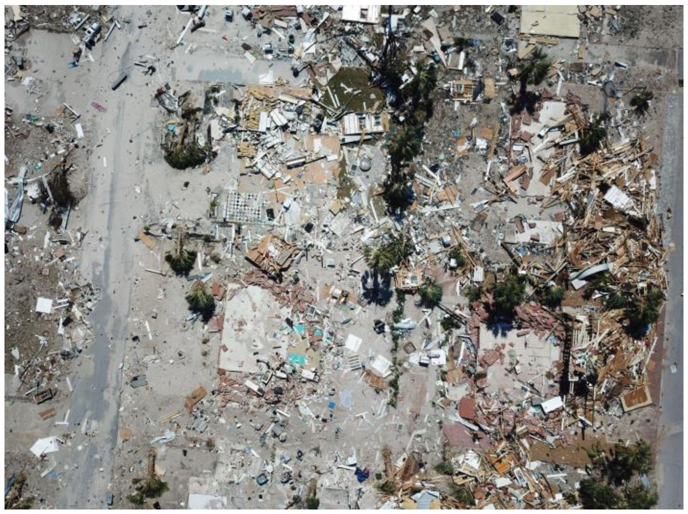


Fig 3.1 Sample from the RescueNet dataset

The RescueNet dataset is unique in that it captures a range of challenges and conditions that are commonly encountered in disaster scenarios, including occlusions, debris, and poor lighting conditions. The synthetic images are generated using a 3D environment modeling tool, which allows for the creation of realistic disaster scenarios with varying levels of complexity and difficulty. The real-world images in the dataset were captured using drones and ground-based cameras in disaster zones, including areas affected by earthquakes and floods. The RescueNet dataset was augmented and annotated using the RoboFlow software.

3.3 SURVIVOR DETECTION FRAMEWORK

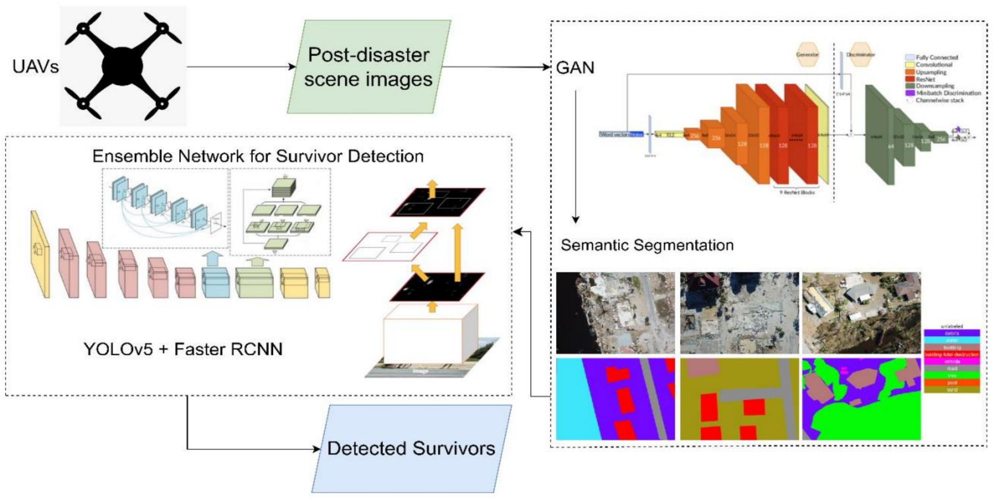


Fig 3.2 Survivor Detection Framework Architecture

Figure 3.2 depicts the survivor detection framework of the proposed work.

3.4 GAN-BASED OCCLUSION REMOVER

3.4.1 CONTEXT-CONDITIONAL GAN (CCGAN)

Context-Conditional Generative Adversarial Networks (CCGANs) are a type of GAN that use contextual information to generate data samples that are specific to a given context. CCGANs can be trained to generate realistic and diverse data samples based on a combination of conditioning variables and contextual information, such as the surrounding environment or the user's behavior. The architecture of a CCGAN, which is depicted by Fig 3.3, consists of three main components: a generator network, a discriminator network, and a context encoder network. The generator network takes two inputs, a random noise vector and a set of conditioning variables, and produces a synthetic sample.

$$L_{\{G\}} = \lambda_{\{L_{\{1\}}\}L_{\{1\}}} + \lambda_{\{adv\}L_{\{adv\}}}$$

(Eq. 3.1)

where $L_{\{1\}}$ is the L1 distance between the generated output image and the real output image, $L_{\{adv\}}$ is the GAN loss function, $\lambda_{\{L_{\{1\}}\}}$ and $\lambda_{\{adv\}}$ are the hyperparameters that control the relative importance of the L1 loss and the GAN loss, respectively.

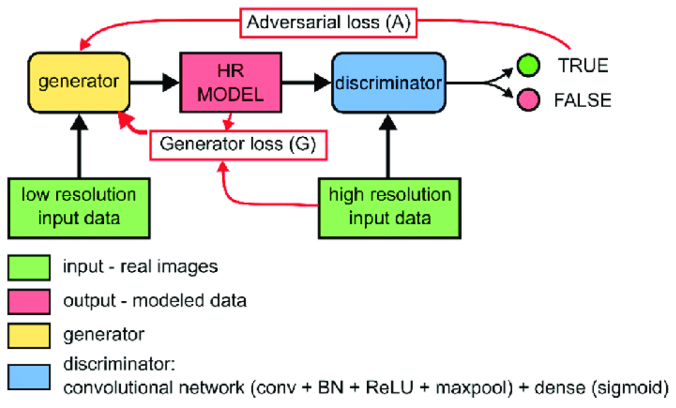


Fig 3.3 High level architecture diagram of CCGAN

For Post-Disaster UAV images, the CCGAN framework was used to remove occlusion or coverage of entities present in the images through image in painting.

In CCGAN, The Generator G is trained to fill in a missing image patch and the Generator G and Discriminator D are conditioned on the surrounding pixels. The model determines if a part of an image is real or fake given the surrounding context.

$$\Gamma_{\{rec\}} = ||P - CE(X'))||_2^2$$

(Eq. 3.2)

where $\Gamma_{\{rec\}}$ is the reconstruction loss, P is the original region before damage, CE is the model and X’ is the entire image that needs to be inpainted.

Algorithm for GAN-based Occlusion Removal Mechanism

GAN-based Occlusion Removal

```
Input:
  • generator: A PyTorch generator model trained to remove occlusions.
  • images: A list of images with occlusions.
  • batch_size: An integer specifying the batch size for inference.
Output: results: A list of images with occlusions removed.
Procedure:
Initialize an empty list results.
For each batch of size batch_size in images, do the following:
    Preprocess the images by resizing, normalizing and converting them to
    tensors.
    Feed the batch of images to the generator and obtain the occlusion-
    removed images.
    Convert the occlusion-removed images back to numpy arrays and append
```

```
        them to results.
Return the results list.
Preprocessing:
    • Resize the input images to the required size for the generator model.
    • Normalize the pixel values of the images to be in the range [-1, 1].
    • Convert the images to tensors.
Occlusion Removal:
    • Feed the pre-processed batch of images to the generator model.
    • The generator model generates occlusion-free versions of the input images.
    • Convert the generated occlusion-free images back to numpy arrays.
```

We first import all necessary libraries to implement the CCGAN architecture, followed by which the configuration for setting up and training the generator and the discriminator are defined. The ImageDataset class is then defined, which enables the loading and usage of a custom dataset to be used for training purposes. Followed by this, the testing and training data loaders are defined for loading the dataset using the ImageDataset class. The generator and the discriminator classes are defined, enabling the creation of the CCGAN model for occlusion removal. The configurations defined earlier are made use of in this stage. The CCGAN model is then trained by first instantiating a model using the above declared classes and training the same on the RescueNet dataset for occlusion removal. The training results and performance evaluation are then printed as output to visualize the efficiency of the trained CCGAN model for occlusion removal.

$$\Gamma = \lambda_{\{adv\}}\Gamma_{\{adv\}} + \lambda_{\{rec\}}\Gamma_{\{rec\}}$$

(Eq. 3.3)

where Γ is the overall loss, $\lambda_{\{\mathrm{adv}\}}$ and $\lambda_{\{\mathrm{rec}\}}$ are the coefficients for tuning the influence of each of the losses. The loss of the Generator and the Discriminator during training are plotted, upon which we ascertain that the CCGAN model was trained with minimal training loss of 0.1413. The occlusion-removed images are then stored in the Segmentation folder to be used in the next module.

3.5 SEMANTIC SEGMENTATION

In semantic segmentation, IoU is used to measure how well the predicted segmentation map aligns with the ground truth segmentation map, i.e., how accurately the model identifies the object boundaries and regions in the image. The IoU score ranges from 0 to 1, where a score of 1 indicates a perfect match between the predicted and ground truth segmentation maps, while a score of 0 indicates no overlap between the two maps.

$$Y = f(X;\Theta) = W^T * g(V;\theta) + b$$

(Eq. 3.4)

where Y is the output segmentation map, X is the input image, Θ is the set of network parameters for the convolutional layers, V is the input to the decoder, θ is the set of network parameters for the decoder, W is the weight matrix, b is the bias term, and g(.) represents the upsampling layers in the decoder.

Fig 3.4 depicts the high-level architecture of semantic segmentation.

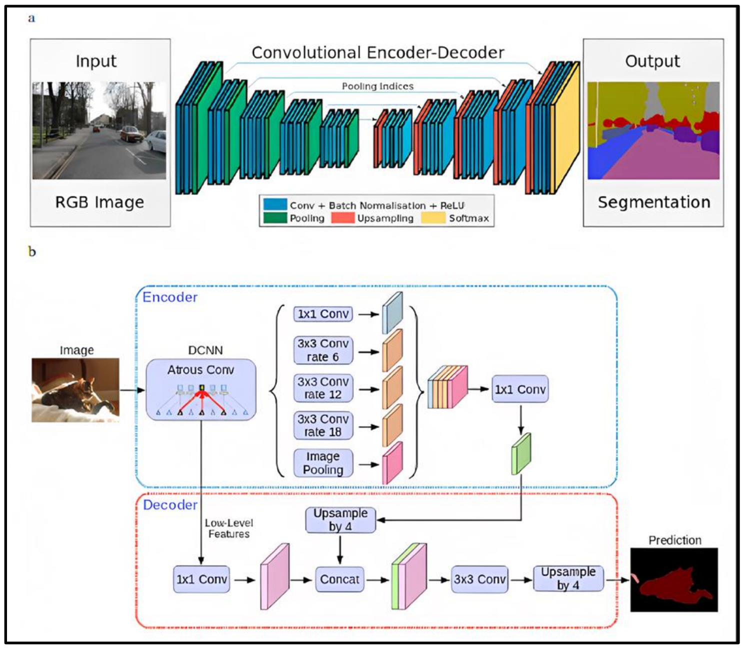


Fig 3.4 High level architecture diagram of semantic segmentation

3.5.1 SEGFORMER

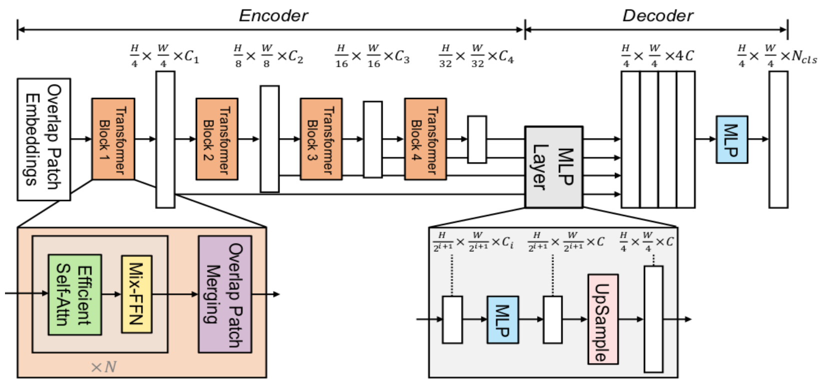
SegFormer is a novel deep learning architecture for semantic segmentation, which aims to accurately classify each pixel in an image into a specific class. The SegFormer architecture is designed to process high-resolution images efficiently while maintaining state-of-the-art accuracy.

Algorithm for Semantic Segmentation of GAN-Enhanced Images

Semantic Segmentation
Input: • model: A PyTorch model trained for semantic segmentation. • images: A list of images to perform segmentation on. • batch_size: An integer specifying the batch size for inference. Output: results: A list of dictionaries containing the segmentation results for each

```
image.
Initialize an empty list results.
For each batch of size batch_size in images, do the following:
        Preprocess the images by normalizing and converting them to tensors.
        Feed the batch of images to the model and obtain the segmentation results.
        Convert the segmentation results to a dictionary format with keys 'mask'
        and 'class', and append the dictionary to results.
Return the results list.
```

SegFormer utilizes a transformer-based encoder that extracts high-level features from the input image, followed by a decoder that generates a pixel-wise segmentation map. Unlike traditional encoder-decoder architectures, SegFormer employs a self-attention mechanism in both the encoder and decoder modules, which enables the model to capture long-range dependencies and contextual information across the image.



Encoder
Decoder
$\frac{H}{4}\times\frac{W}{4}\times C_1$
$\frac{H}{8}\times\frac{W}{8}\times C_2$
$\frac{H}{16}\times\frac{W}{16}\times C_3$
$\frac{H}{32}\times\frac{W}{32}\times C_4$
$\frac{H}{4}\times\frac{W}{4}\times 4C$
$\frac{H}{4}\times\frac{W}{4}\times N_{cls}$
Overlap Patch Embeddings
Transformer Block 1
Transformer Block 2
Transformer Block 3
Transformer Block 4
MLP Layer
MLP
Efficient Self-Attn
Mix-FFN
Overlap Patch Merging
$\times N$
$\frac{H}{2^{i+1}}\times\frac{W}{2^{i+1}}\times C_i$
$\frac{H}{2^{i+1}}\times\frac{W}{2^{i+1}}\times C$
$\frac{H}{4}\times\frac{W}{4}\times C$
MLP
UpSample

Fig 3.5 Architecture of SegFormer

Additionally, SegFormer employs a hybrid attention mechanism that combines self-attention with spatial attention, which allows the model to focus on relevant spatial regions and enhance its ability to handle complex images with fine-grained details.

$$L = -\frac{1}{N} \sum_{i} \sum_{j} [y_{ij} * \log(\hat{y}_{ij}) + (1 - y_{ij}) * \log(1 - \hat{y}_{ij})]$$

(Eq. 3.5)

where N is the total number of pixels in the image, y_{ij} is the ground truth label for pixel (i,j), $\hat{y}_{ij}$ is the predicted label for pixel (i,j), and the summation is taken over all pixels in the image. Deploying the SegFormer framework on the GAN-enhanced post-disaster images results in pixel-wise prediction of various entities present in an image. This mechanism generates a color-coding, wherein each entity present in the image is associated with a unique color, thereby making it easy for a detection model to classify between various entities present in the image. Classes of importance, namely car and people, were annotated for semantic segmentation and used to train the SegFormer model. The necessary libraries were imported to be able to implement the SegFormer framework. The images obtained as output from the GAN module are transferred to the GAN folder to be able to make use of the occlusion-removed images for semantic segmentation. The SemanticSegmentationDataset class is defined to load and process the dataset for the semantic segmentation task.

The SegFormer model is then defined and implemented to carry out the task of semantic segmentation on the images obtained from the previous module. The SegformerFinetuner class is defined to prune and finetune the parameters used for implementing the model. The model is then trained for 300 epochs. The Early Stopping mechanism was incorporated to reduce the chances of overfitting of the model. The model was trained by utilizing the GPU provided by Google Colab using the CUDA library. Tensorboard was instantiated to print the performance evaluation metrics that were witnessed during the training period. The model had minimal training loss, recorded at 0.0233.

3.6 HYBRID ENSEMBLE NETWORK

3.6.1 ENSEMBLE NETWORK

By combining multiple models that may have different strengths and weaknesses, an ensemble network can produce more accurate and stable predictions. Additionally, ensemble networks can improve the interpretability of models by providing a measure of the uncertainty in the predictions, which is particularly important in applications where the consequences of incorrect predictions are severe.

Algorithm for the Hybrid Ensemble Network Of YOLOv5 and Faster RCNN models

Hybrid Ensemble Network

Input:

- single_stage_model: A PyTorch model trained with a single-stage detector architecture.
- multi_stage_model: A PyTorch model trained with a multi-stage detector architecture.
- images: A list of images to detect objects in.
- conf_threshold: A confidence threshold below which predictions are suppressed.

```
    • iou_threshold: An IoU threshold above which overlapping predictions are
      suppressed.
Output: results: A list of dictionaries containing the detection results for each
image.
Initialize an empty list of results.
For each image in images, do the following:
    Use the single-stage model to detect objects in the image and store the
    detection results in a list ‘single_stage_results’.
    Use the multi-stage model to detect objects in the image and store the
    detection results in a list ‘multi_stage_results’.
    Concatenate the detection results from the single-stage and multi-stage
    models into a single tensor.
    Apply non-maximum suppression to the tensor with confidence threshold
    conf_threshold and IoU threshold iou_threshold.
    Convert the detection results to a dictionary format with keys 'boxes',
    'scores', and 'labels', and append the dictionary to results.
Return the results list.
```

The proposed work aims to combine a single-stage and multi-stage object detection frameworks. An ensemble model comprising the YOLOv5 single-stage detector and the Faster RCNN multi-stage detector was implemented. The main advantage of the proposed combination of single-stage and multi-stage detectors is the improved object detection performance over an independent single-stage detector and lower false negative rate than an independent multi-stage detector. YOLOv5, being one of the most popular and powerful single-shot detectors, is a stable version of the ‘You Only Look Once’ family of CNN-based object detection models. Faster RCNN is a very powerful two-stage detector that is widely used for object detection mechanisms.

$$Y = \frac{1}{N} \Sigma_i \left[f_{i(X;\Theta_i)} \right]$$

(Eq. 3.6)

where Y is the set of predicted bounding boxes and class labels, N is the number of models in the ensemble, $f_{i(.)}$ represents the i-th model in the ensemble, X is the input image, and θ_i is the set of network parameters for the i-th model.

3.6.1.1 YOU ONLY LOOK ONCE (YOLOv5)

YOLOv5, which stands for (You Only Look Once version 5) is a state-of-the-art single-stage deep learning model that achieves high accuracy and real-time performance on object detection tasks. YOLOv5 comprises a highly efficient single-stage architecture, which uses a grid-based approach to divide the image into multiple cells, each of which is responsible for detecting and localizing one or more objects. The loss function used for training the YOLOv5 model is as follows:

$$L = \lambda_{\mathrm{r}} L_{\mathrm{r}} + \lambda_{cL_c} + \lambda_{\mathrm{a}} L_{\mathrm{a}}$$

(Eq. 3.7)

where λ_r, λ_c, and λ_a are hyperparameters that control the relative importance of the different components of the loss function, L_r is the regression loss that penalizes the difference between the predicted and ground truth bounding box coordinates, L_c is the classification loss that penalizes the difference between the predicted and ground truth class probabilities, and L_a is the anchor loss that encourages the model to predict anchors that match the ground truth objects. The YOLOv5 model uses anchor boxes to predict bounding box coordinates. The anchor boxes are defined as a set of k clusters of bounding boxes computed using k-means clustering on the ground truth bounding boxes. The anchor loss is used to encourage the model to predict anchor boxes that match the ground truth objects:

$$L_{\mathrm{a}} = \lambda_{a\sum_{\mathrm{j}}}\left(1 - IOU(B_{\mathrm{j}}, A_{\mathrm{mj}})\right)^2$$

(Eq. 3.8)

where A_{mj} is the m-th anchor box that best matches the j-th ground truth object, and $IOU(B_j, A_{mj})$ is the intersection over union (IOU) between the predicted and ground truth bounding boxes. λ_a is a hyperparameter that controls the relative importance of the anchor loss. The YOLOv5 model incorporates several significant improvements over its predecessors, including a new backbone architecture based on cross-stage partial networks (CSPNet), which enhances the model's ability to detect objects at different scales, and a new prediction head that employs anchor-free bounding box regression, which makes the model more robust and reduces the need for manual tuning. Fig 3.6 shows the architecture of YOLOv5.

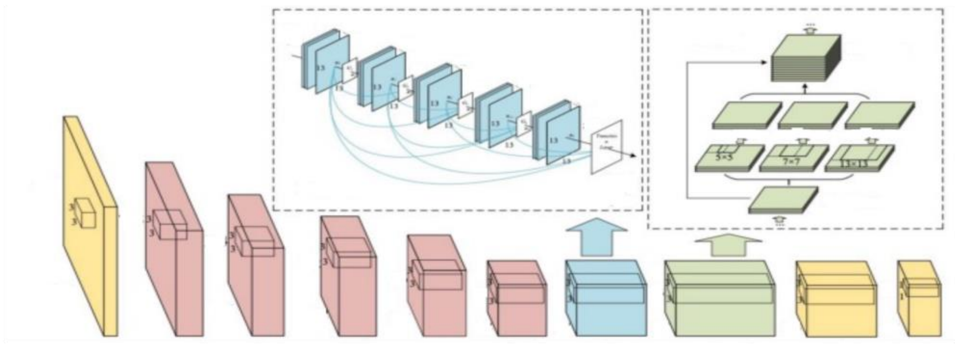


Fig 3.6 Architecture of YOLOv5

3.6.1.2 FASTER REGION-BASED CONVOLUTIONAL NEURAL NETWORK (FASTER R-CNN)

Faster R-CNN is a deep learning model used for object detection that improves upon the original R-CNN model. It consists of two main components: a Region Proposal Network (RPN) and a Fast R-CNN network. The loss function used for training the Faster R-CNN model can be represented as follows:

$$L = L_{rpn} + L_{roi}$$

(Eq. 3.9)

where L_{rpn} is the loss function for the RPN that encourages it to generate accurate proposals, and L_{roi} is the loss function for the second-stage network that classifies the proposals and refines their bounding box coordinates.

The RPN generates a set of object proposals by sliding a small network over the convolutional feature map output by a shared backbone network. The RPN outputs two sets of scores for each anchor box: one that predicts the probability of an object being present and the other that predicts the offsets of the bounding box for each object. These proposals are then refined using non-maximum suppression (NMS) to generate a set of high-quality proposals. The RPN loss function encourages the network to generate accurate scores and coordinates for the positive proposals (those that have high overlap with a ground truth object) and to suppress the scores for the negative proposals (those that have low overlap with any ground truth object):

$$L_{rpn} = L_{obj} + \lambda_{reg} L_{reg}$$

(Eq. 3.10)

where L_{obj} is the binary cross-entropy loss for objectness classification, L_{reg} is the smooth L1 loss for bounding box regression, and λ_{reg} is a hyperparameter that controls the relative importance of the two components.

Faster R-CNN has achieved state-of-the-art performance on several benchmark datasets, including PASCAL VOC and MS COCO. However, the two-stage architecture of Faster R-CNN can be computationally expensive, especially during inference. Therefore, several alternative models such as YOLO and SSD have been proposed to address this issue by using a single-stage architecture. Fig 3.7 depicts the architecture of Faster RCNN.

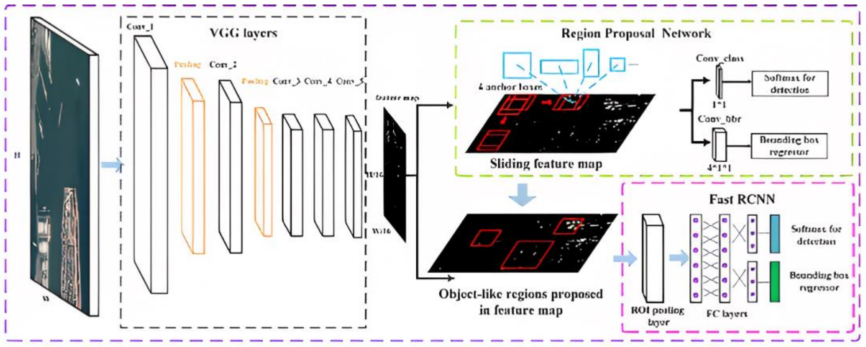


Fig 3.7 Architecture of Faster RCNN

3.6.1.3 SIMPLE AVERAGING

Simple averaging is a popular ensemble method used in machine learning and statistical modeling, which involves combining multiple models or algorithms to improve their predictive performance. Simple averaging works by taking the average of predictions made by multiple models, which can provide a more accurate and robust prediction compared to any individual model. The simplicity and ease of implementation of simple averaging make it a widely used method for ensembling in various domains such as image recognition, natural language processing, and finance.

Let X be the input data, and let f1(X), f2(X), ..., fn(X) be the output of n different models trained on the same dataset. Then the simple averaging ensemble technique combines the outputs of these models as follows:

$$\hat{y} = \left(\frac{1}{n}\right) * \sum_{i} fi(X)$$

(Eq. 3.11)

where ŷ is the final prediction for the input X.

The effectiveness of simple averaging depends on several factors such as the diversity of the models, their prediction errors, and the correlation between their predictions. Simple averaging can work well when the individual models have complementary strengths and weaknesses and are trained on different subsets of data. It reduces the variance of the predictions by averaging out the noise and errors in the individual models. Furthermore, it improves the overall accuracy and robustness of the predictions by reducing the impact of individual errors or biases in each model.

The images obtained as output were saved to the respective folders of YOLOv5 and Faster RCNN models to be trained on enhanced images with entity separation for better performance. The YOLOv5 model is instantiated by downloading all dependencies from the official ‘ultralytics’ github repository. Our dataset of choice, namely RescueNet, is annotated and augmented using the

Roboflow software. Additional pre-processing techniques are incorporated within the Roboflow workspace, and a unique api key is generated for our dataset. This api key is then used to download the dataset and the respective annotations in PyTorch YOLOv5 format. The YOLOv5 model is then trained with batch size 16 for 492 epochs. The configurations previously defined in custom_yolov5s.yaml are also given as input parameters. In our case, Early stopping was observed as the model performance did not vary over a period of epochs, thereby stopping the training process in the 492nd epoch.

The Mean Average Precision (mAP) is calculated for every epoch/iteration and displayed in the output. Finally, the total number of instances that the model visualized under each class is tabulated and presented, and the weights are stored in the yolov5s_results folder as ‘best.pt’. Tensorboard is then made use of to visualize the performance of the model. Tensorboard plots all performance metrics observed while training the model over the range of epochs for which the model was trained. The overall training loss of the model was found to be 0.002 for identifying various classes.

The mean average precision (mAP) of the model after 492 epochs was found to be 0.55. The trained model is then tested by visualizing results obtained when test images are passed through the model. The bounding boxes constructed over detected classes are displayed. The Faster RCNN multi-stage CNN model is implemented using Detectron2, which is a computer vision library that allows the implementation of various CNN models through the usage of PyTorch. Initially, the GPU instantiated in the Google Colab environment is displayed using the nvidia-smi command. The Faster RCNN model is then instantiated and trained using the faster_rcnn_X_101_32x8d_FPN_3x architecture present in Detectron2. Other parameters required for training the model are defined as well. The dataset previously annotated and augmented on Roboflow is downloaded along with annotations in the COCO JSON format. Roboflow being a versatile software allows exporting various annotation formats for the same dataset. The COCO JSON format is used while training a Faster RCNN model. The dataset is then registered to detectron2 to be able to use the dataset for training the Faster RCNN model.

Once the model is trained, Tesnorboard is initialized, thereby giving various performance metrics observed during the training process of the Faster RCNN model. Total loss of the model was observed to be 0.89. The accuracy of detecting various classes was found to be 94.92%. The trained model is then tested using the test set and results are visualized. The model and its corresponding weights are then saved and stored in the ‘Saved_Models’ folder.

The final ensemble model is created using the two trained models obtained in the previous steps. Both the models are executed for the same image and their respective outputs are visualized. The models are then combined using the ensembling technique named ‘simple averaging’, wherein multiple models are trained independently and their predictions are combined to improve overall performance. The ensembled model is then evaluated using the same image used for the previous models and the outputs are compared. Survivors were detected with an accuracy of 96.4%.

CHAPTER 4

EXPERIMENTAL SETUP

4.1 LANGUAGES USED

The proposed system was developed using Python 3. The grammar of Python is straightforward and user-friendly, which makes it the perfect language for quick experimentation and prototyping. Python functions can be accelerated using the Numba package by having them converted to machine code. It supports CUDA, a parallel computing framework and programming model created by NVIDIA for use with its GPUs for general computation. Additionally, the PyTorch and TensorFlow frameworks were utilized to implement the various models involved in the proposed work. Those two libraries are favored by the community and businesses because they can leverage the power of the NVIDIA GPUs. That is very useful, and sometimes necessary, for processing big datasets like a corpus of text or a gallery of images.

4.2 TOOLS AND PACKAGES USED

4.2.1 TENSORFLOW

TensorFlow is an open-source software library developed by Google for machine learning and deep neural networks. In this project, it has been used for instantiating, developing and training the YOLOv5 model.

4.2.2 PYTORCH

PyTorch is an open-source machine learning library based on the Torch library. It is developed and maintained by Facebook's AI research group and is widely used for developing deep learning models in various fields such as computer vision. In this project, it has been used to develop the Faster RCNN model through the usage of Detectron2.

4.2.3 GOOGLE COLAB

Google Colab is a python-based framework used for Deep Learning tasks with free access to Graphical Processing Units (GPUs) and Tensor Processing Units (TPUs). In this project, it has been used to implement the GAN-based occlusion remover, semantic segmentation model and the ensemble network.

4.2.4 CUDA

CUDA is a parallel computing platform that provides an API for developers, allowing them to build tools that can make use of GPUs for general purpose processing. In this project, CUDA is used to enable a GPU for the program, which will improve the efficiency of training of the various models involved.

4.2.5 OPENCV

OpenCV is a huge open-source library for computer vision, ML, and image processing. When it integrates with various libraries, such as NumPy, python can process the OpenCV array structure for analysis. In this project, it has been used for analyzing the images given as input to the survivor detection models.

4.2.6 KERAS

Keras is a powerful and widely-used open-source DL library that provides a high-level API for building and training neural networks. It allows for efficient computation because it is built on top of lower-level libraries like TensorFlow and

Theano. Additionally, Keras provides a wide range of pre-built layers, activation functions, and loss functions that make it simple to build complex models while abstracting away implementation details.

4.2.7 NUMPY

NumPy, or Numerical Python, is a library that contains multidimensional array objects and a set of procedures for manipulating them. It has a number of features, including an N-dimensional array object with a lot of power, advanced (broadcasting) capabilities, C/C++ and Fortran code integration tools, linear algebra, the Fourier transform, and random number abilities.

4.2.8 PANDAS

Pandas is a widely-used open-source Python library for data science, data analysis, and ML activities. It is built on top of NumPy, a library that supports multi-dimensional arrays. It is the ideal tool for dealing with complex data in the real world.

4.2.9 MATPLOTLIB

Matplotlib is a Python package for plotting and visualization that generates production quality graphs. It is designed to be able to create simple and complex plots with only a few lines of code. Matplotlib is written to use NumPy and other extension codes. In this project, it has been used to plot the performance evaluation metrics post training of the survivor detection models.

4.3 EVALUATION METRICS

There are various metrics that can be used to evaluate the performance of a machine learning model, and the choice of metric depends on the specific task and the nature of the data. The metrics used in this project are:

4.3.1 IOU

A typical statistic for assessing how well image segmentation models perform is IoU (IoU). It calculates the degree to which the predicted segmentation map and the actual segmentation map overlap. IoU is calculated as the ratio of the intersection of the predicted and ground truth segmentation maps to their union:

$$IoU = \frac{Intersection}{Union}$$

(Eq. 4.1)

where the number of pixels that appear in both the anticipated and actual segmentation maps is known as the intersection, and the total number of pixels that appear in either segmentation map is known as the union.

4.3.2 PRECISION

A classification model's precision is a metric that assesses the share of accurate positive predictions among all positive predictions. In other words, it assesses how well the model can recognise good examples. Precision is calculated as the number of true positives divided by the sum of true positives and false positives:

$$Precision = \frac{True\ Positives}{(True\ Positives\ +\ False\ Positives)}$$

(Eq. 4.2)

where true Positives are the number of positive cases that the model correctly predicted, whereas False Positives are the number of negative instances that the model mistook for positive.

4.3.3 RECALL

A classification problem's recall metric, also known as sensitivity or the true positive rate, counts the proportion of true positive predictions among all real positive examples. Recall is calculated as the number of true positives divided by the sum of true positives and false negatives:

$$Recall \ = \frac{True\ Positives}{(True\ Positives\ +\ False\ Negatives)}$$

(Eq. 4.3)

where false Negatives are instances of positive data that the model misinterpreted as negative, whereas True Positives are the number of instances of positive data that the model properly forecasted as positive.

4.3.4 F-SCORE

The F-Score, also called the F1-Score, is a measure of a model's accuracy on a dataset. It is used to evaluate binary classification systems, which classify examples into 'positive' or 'negative'.

$$F1 - Score = 2 * \frac{Precison * Recall}{Precison + Recall}$$

(Eq. 4.4)

4.3.5 ACCURACY

A classification model's total performance is measured by accuracy, which is the percentage of accurate predictions the model makes. It is determined by dividing the total number of forecasts by the number of accurate guesses.

$$Accuracy = \frac{(True\ Positives\ +\ True\ Negatives)}{(True\ Positives\ +\ False\ Positives\ +\ True\ Negatives\ +\ False\ Negatives)}$$

(Eq. 4.5)

where False Positives are instances of negative events that were mistakenly predicted as positive, and False Negatives are instances of positive events that were mistakenly predicted as negative. True Positives are the number of positive instances that were correctly predicted, True Negatives are the number of negative instances that were correctly predicted, and False Positives are False Negatives.

4.3.6 GENERATOR ADVERSARIAL LOSS

Generative Adversarial Loss (GAN Loss) is a mathematical function used in training AI models that can generate realistic images, videos, or audio. The function is used to train two networks: a generator network that tries to create realistic data, and a discriminator network that tries to differentiate between real data and generated data. The GAN loss encourages the generator network to generate more realistic data by penalizing it when the discriminator can easily tell that the data is not real. At the same time, the GAN loss encourages the discriminator to be better at telling the difference between real and generated data. Using GAN loss in training has been shown to be effective in creating high-quality data, but it can be challenging to get it right.

The GAN loss equation can be written as:

$$\mathrm{L(G, D) = E\left[\log\left(D(x)\right)\right] + E\left[\log\left(1 - D\left(G(z)\right)\right)\right]}$$

(Eq. 4.6)

where, G is the generator network, D is the discriminator network, x is a real data sample, z is a random noise vector fed as input to the generator to generate fake data. The first term in the equation measures the ability of the discriminator to correctly classify real data samples, and the second term measures the ability of the discriminator to correctly classify fake data generated by the generator. The generator network is trained to minimize this loss, while the discriminator network is trained to maximize it.

4.3.7 GENERATOR PIXEL LOSS

Pixel loss is a measure of the difference between the pixels of two images. Generator pixel loss is a loss function used in image generation tasks, where the generator network is trained to generate images that are as similar as possible to a set of target images.The generator pixel loss equation is based on the Mean Squared Error (MSE) between the generated image and the target image and is defined as:

$$\mathrm{L1 = \left||G(x) - y|\right|^2}$$

(Eq. 4.7)

where, G(x) is the generated image from input x, y is the target image. The goal of the generator network is to minimize this loss function L1 by adjusting its weights during training. This means that the generated image should be as close as possible to the target image in terms of pixel values. The lower the value of L1, the better the generator network has learned to create images that are similar to the target images.

4.3.8 DISCRIMINATOR LOSS

The discriminator loss is typically used in the context of training a Generative Adversarial Network (GAN). In this case, the discriminator's goal is to distinguish between real and fake data, while the generator's goal is to create fake data that is indistinguishable from the real data. The discriminator loss is typically defined as the cross-entropy loss between the discriminator's predictions and the true labels:

$$L_D = -\left(\frac{1}{N}\right) * \ \mathrm{sum}_i\left[\, y_i * \log\left(D(x_i)\right) + \ (1 - y_i) * \log\left(1 - D(x_i)\right)\right]$$

(Eq. 4.8)

where, N is the batch size, x_i is the i-th input data sample, y_i is the true label (1 for real data, 0 for fake data), D(x_i) is the discriminator's output (a probability between 0 and 1) for input x_i. The discriminator tries to minimize this loss by correctly classifying the real and fake data, while the generator tries to maximize it by creating better fake data that can fool the discriminator.

CHAPTER 5

RESULTS AND ANALYSIS

5.1 GAN-BASED OCCLUSION REMOVER

The GAN-based occlusion removal mechanism took as input post-disaster images taken from a UAV with a camera sensor mounted on it, of which several images contained occluded survivors present in them. The model gave as output images with the occluded survivors being regenerated, thereby enabling survivor detection with better performance. The Generator pixel loss, Generator Adversarial loss and the Discriminator loss were plotted against the number of instances seen by the model. Fig 5.1, Fig 5.2, and Fig 5.3 depict the Generator Pixel Loss, Generator Adversarial Loss and the Discriminator Loss respectively.

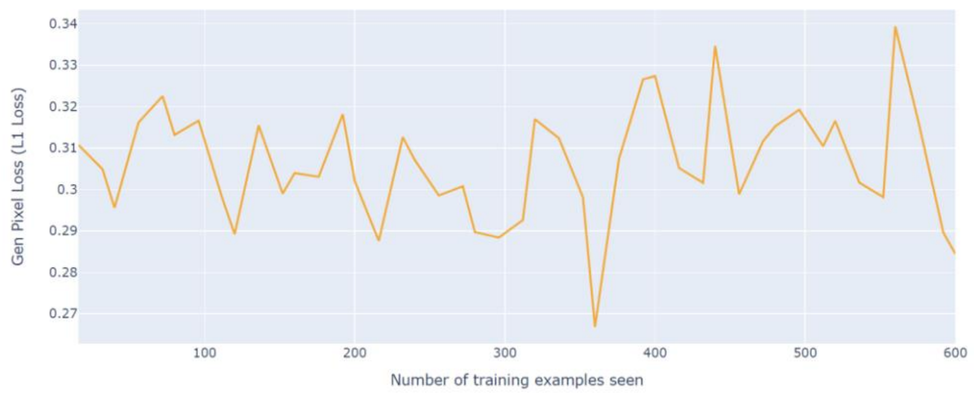


Fig 5.1 Generator Pixel Loss

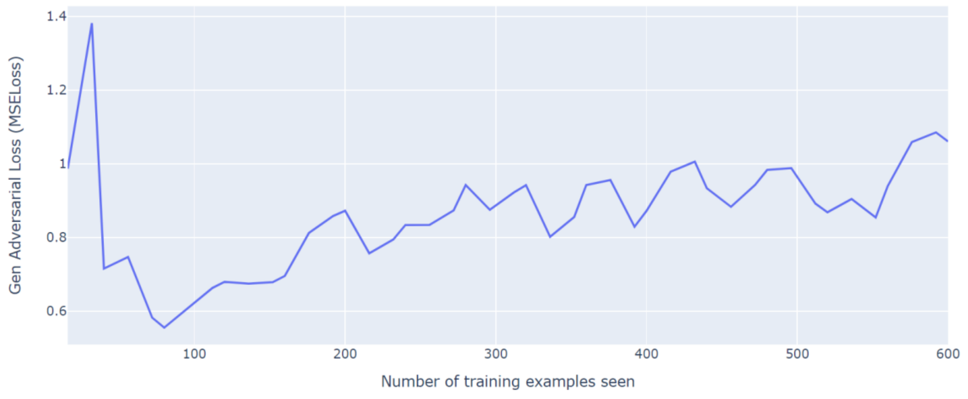


Fig 5.2 Generator Adversarial Loss

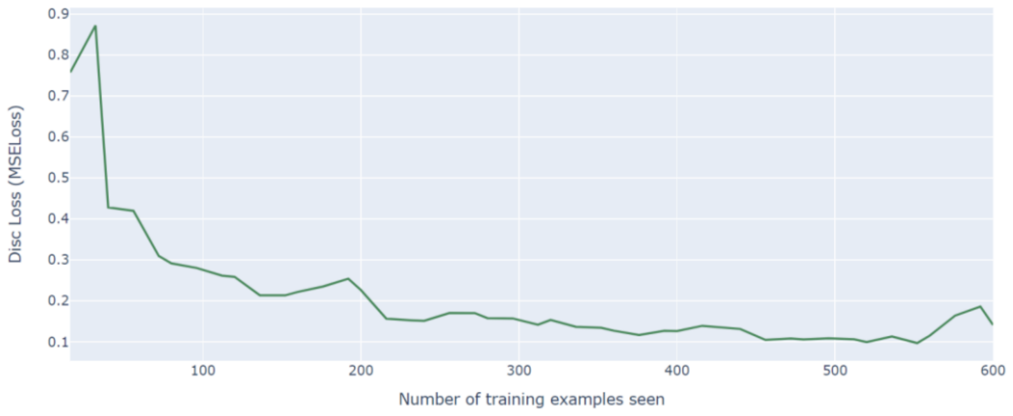


Fig 5.3 Discriminator Loss

CCGAN does the task of image inpainting on top of the occluded images, thereby creating a mask on top of the images. When the GAN model is trained to regenerate the image by removing the mask, it also gets trained to regenerate the occluded survivors present within or outside the mask, thereby improving the performance of the model. Fig 5.4 depicts a sample output obtained from the GAN model.

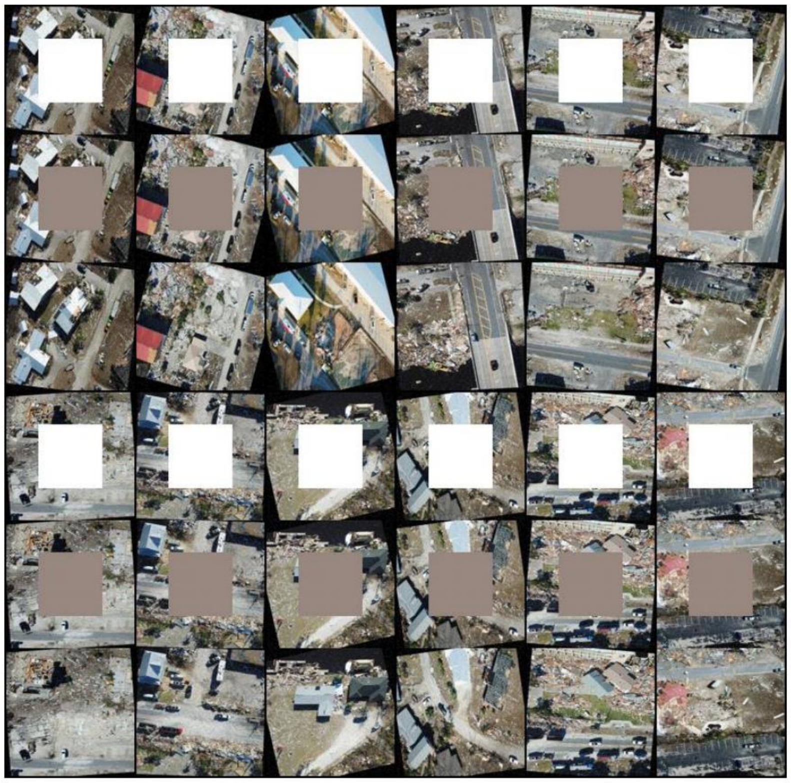
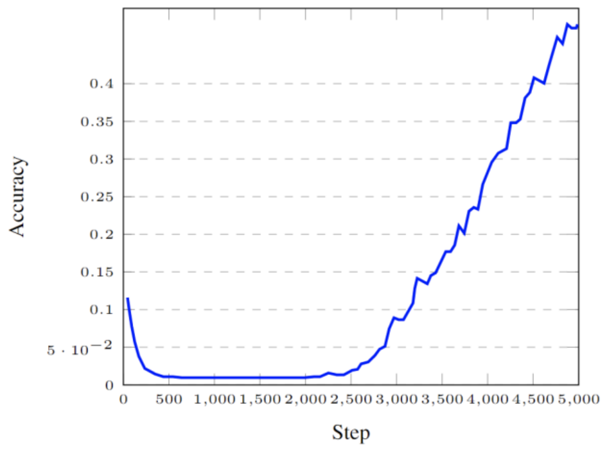


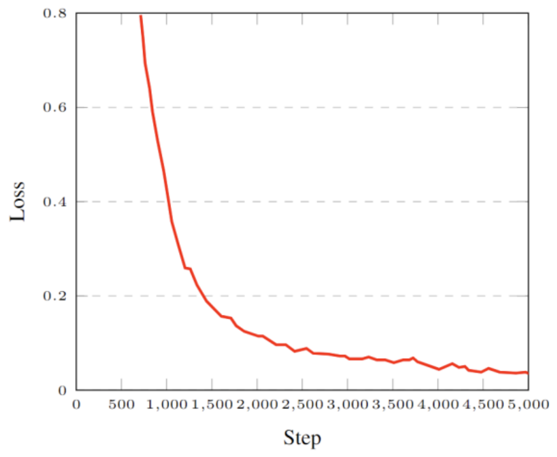
Fig 5.4 Output from the GAN-based occlusion remover

5.2 SEMANTIC SEGMENTATION

The semantic segmentation module takes in as input the enhanced images obtained from the GAN module, and gives as output the entities present in the image classified into separate groups grouped by a color coding generated on top of the entities. The accuracy and the loss obtained while training the SegFormer model for semantic segmentation were recorded and plotted against the steps through which the training process took place. Fig 5.5, Fig 5.6 and Fig 5.7 depict the Accuracy vs Step curve, Loss vs Step curve and the sample output of the semantic segmentation module.



(a) Accuracy vs Step curve



(b) Loss vs Step curve

Fig 5.5 Semantic Segmentation Performance Metrics

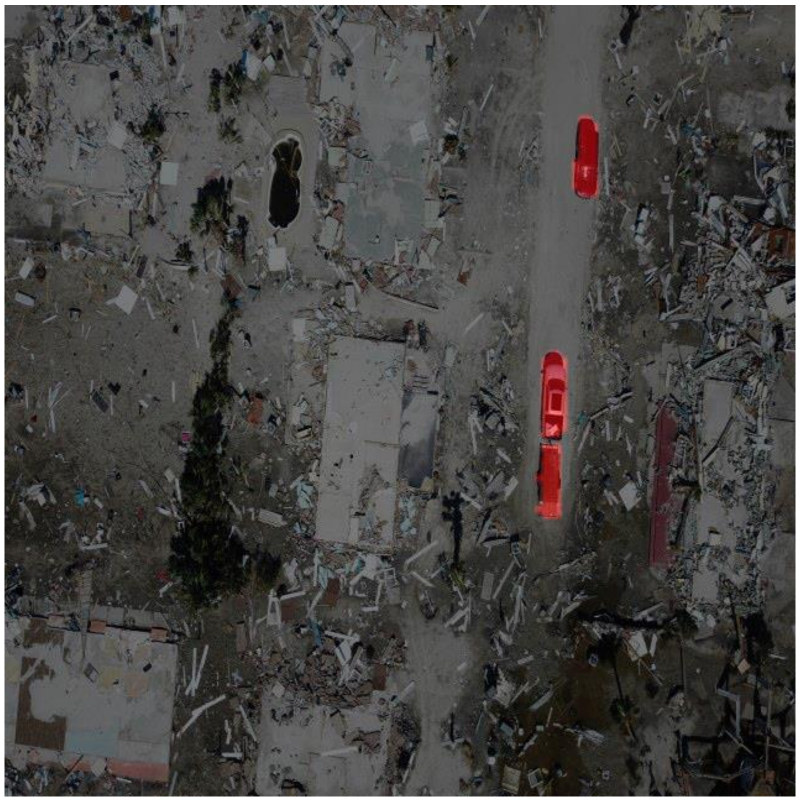


Fig 5.6 Output from the Semantic Segmentation module

5.3 ENSEMBLE NETWORK

5.3.1 YOLOV5 SINGLE-STAGE DETECTOR

The YOLOv5 object detection model was trained on the images obtained as output from the semantic segmentation model. The performance metrics obtained during the training process were obtained and plotted using the TensorBoard tool in Colab. Fig 5.8 displays all the performance metrics plotted during the training process of the YOLOv5 model. The standalone YOLOv5 model obtained a survivor detection accuracy of 55.5%. Fig 5.9 depicts sample output obtained for a test image from the trained YOLOv5 model. The model had low inference time which is desirable, but the accuracy of the model is low.

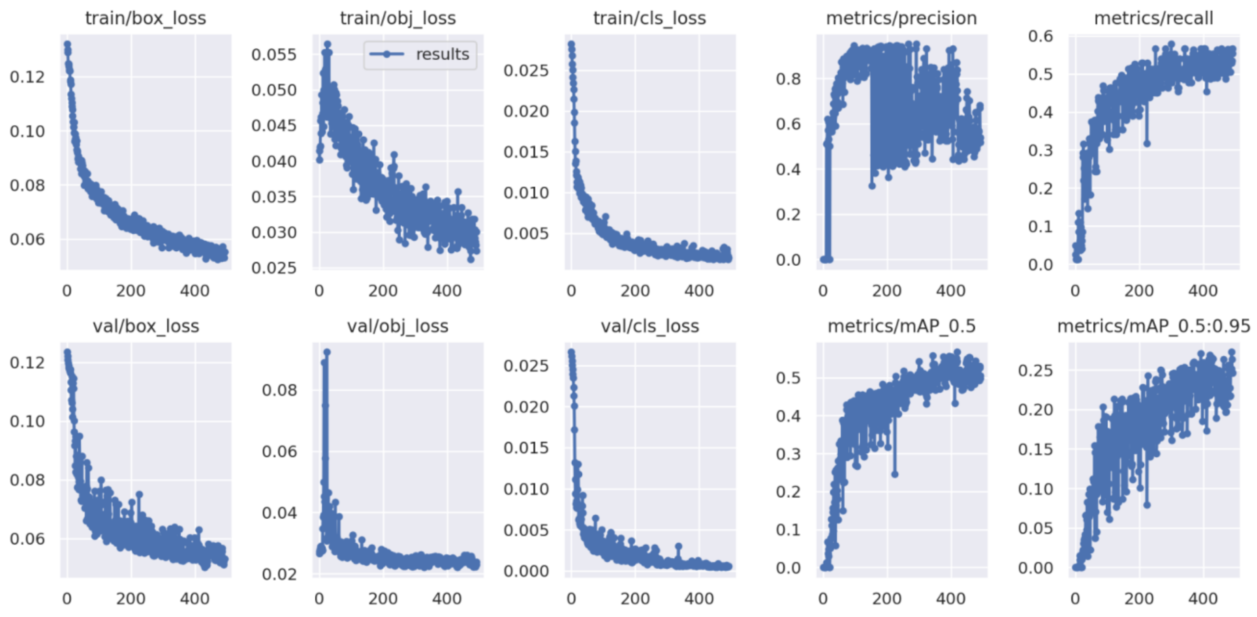


Fig 5.7 YOLOv5 performance metrics

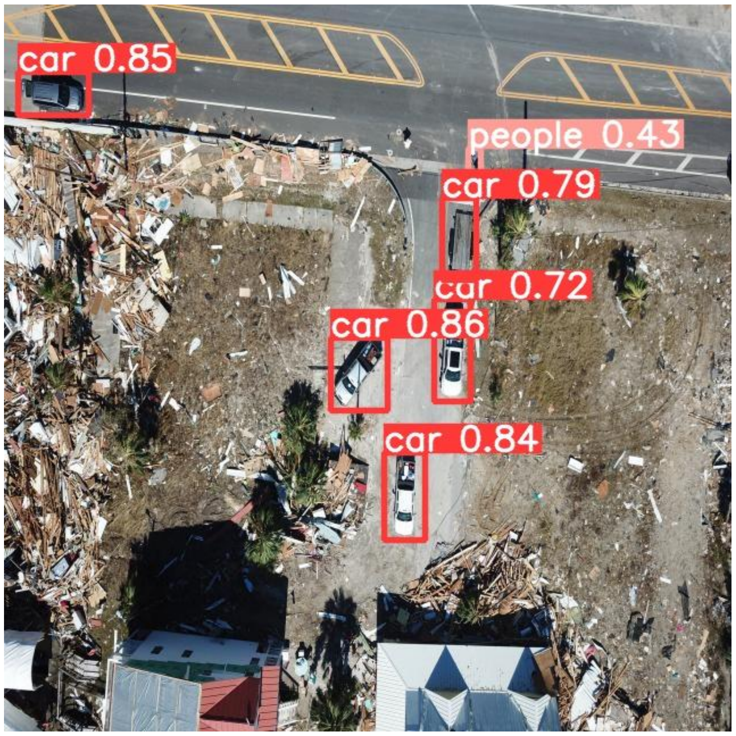
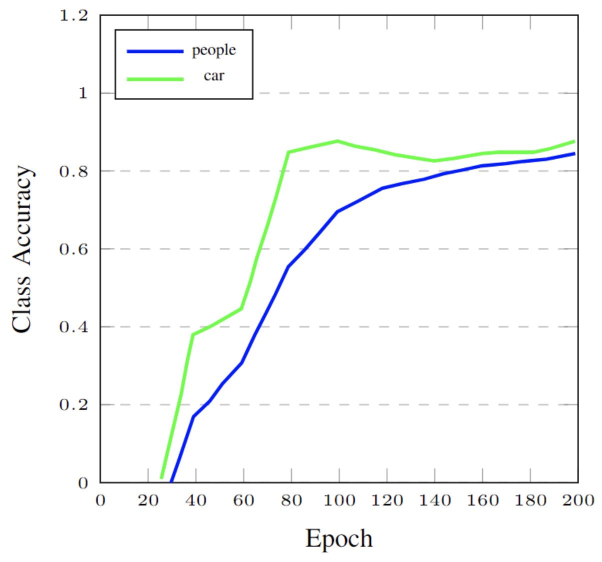
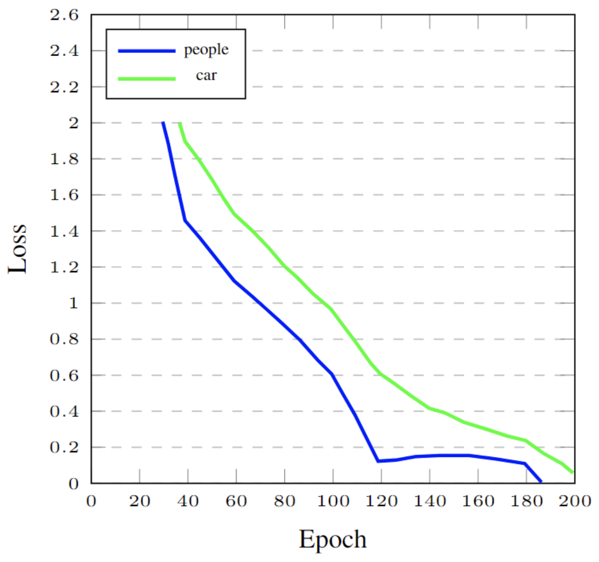


Fig 5.8 Sample YOLOv5 Output

5.3.2 FASTER RCNN MULTI-STAGE DETECTOR

The Faster RCNN object detection model was trained on the images obtained as output from the semantic segmentation model. The performance metrics obtained during the training process were obtained and plotted using the TensorBoard tool in Colab. Fig 5.10 and Fig 5.11 display the Accuracy-Epoch curve and Loss-Epoch curve of Faster RCNN respectively. The standalone Faster RCNN model obtained a survivor detection accuracy of 96.4%. Fig 5.12 depicts sample output obtained for a test image from the trained Faster RCNN model. The model however has very high inference time.





(a) Accuracy-Epoch curve **(b) Loss-Epoch curve**

Fig 5.9 Faster RCNN Performance Metrics

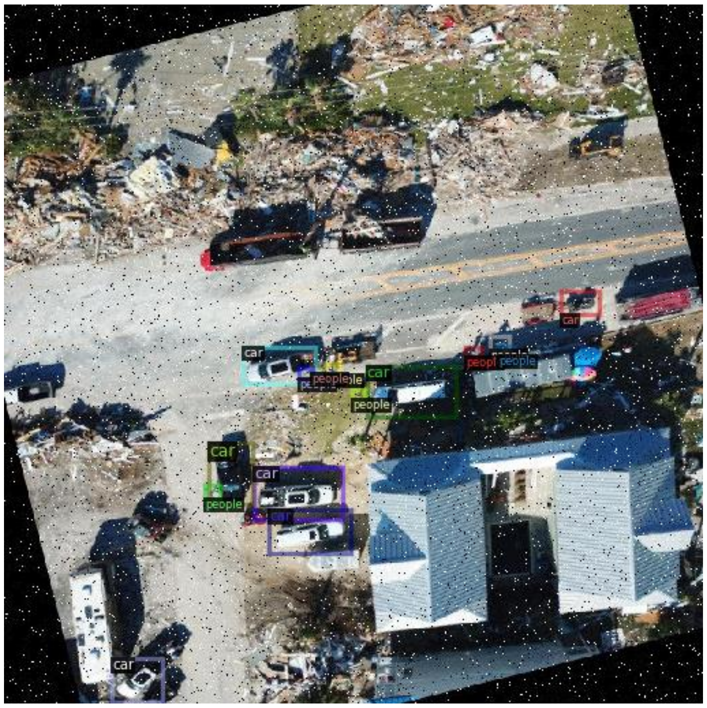


Fig 5.10 Sample Faster RCNN Output

5.3.3 FINAL ENSEMBLE NETWORK

The final ensemble model involved the usage of the trained weights of the standalone YOLOv5 and Faster RCNN models to be combined using simple averaging, thereby giving the output for test images. Fig 5.13 depicts the output received from the ensemble model with lower inference time.

Table 4.1 Performance comparison of standalone object detection models and the proposed hybrid ensemble network

Model	**Accuracy**	**Precision**	**Recall**	**F1 score**
YOLOv5	55.5	47.3	52.8	67.5
Faster RCNN	94.95	89.72	91.25	96.23
Proposed Hybrid Ensemble	96.4	92.5	93.79	97.95

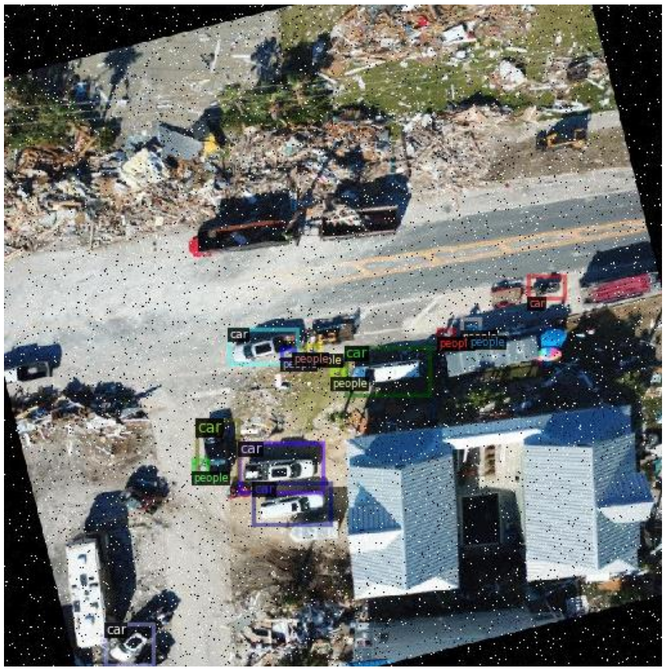


Fig 5.11 Final Ensemble Model Output

CHAPTER 6

CONCLUSION

Natural calamities lead to immense building damage and cause havoc among people who get trapped in the disaster. Since survivors may be present in a disaster-struck area, post-disaster survivor detection is essential to carry out effective Search and Rescue operations. Though UAVs are being widely used to scan the post-disaster area for survivors, inaccurate detection mechanisms lead to many survivors not being detected. Hence to mitigate the issues faced by current survivor detection frameworks, we propose a UAV-based post- disaster survivor detection mechanism that employs a GAN-aided ensemble network. The proposed mechanism aims to improve the accuracy of survivor detection in disaster-stricken areas. The GAN-aided ensemble network is trained using UAV imagery, enabling it to detect survivors in real-time. Furthermore, a novel GAN-aided semantic segmentation pre- processing module has been implemented to enhance the images fed as input to the detection model. The hybrid ensemble model comprising a single-stage YOLOv5 model and a multi- stage Faster RCNN model improves the accuracy of survivor detection, and reduces the inference time, thereby making the framework more suitable for real-time use. The proposed mechanism has shown promising results in terms of accuracy and speed compared to existing methods.In the future, the integration of data obtained from other sensors such as thermal imaging cameras, ground-based sensors, and acoustic sensors can be implemented to improve detection accuracy.

REFERENCES

[1] J. Dong, K. Ota and M. Dong, ”UAV-Based Real-Time Survivor Detection System in Post-Disaster Search and Rescue Operations” in IEEE Journal on Miniaturization for Air and Space Systems, vol. 2, no. 4, pp. 209-219, 2021

[2] M. Rahnemoonfar, T. Chowdhury, A. Sarkar, D. Varshney, M. Yari and R. R. Murphy, ”FloodNet: A High Resolution Aerial Imagery Dataset for Post Flood Scene Understanding,” in IEEE Access, vol. 9, pp. 89644-89654, 2021

[3] M. ˙Zarski, B. W´ojcik, J. A. Miszczak, B. Blachowski and M. Ostrowski, ”Computer Vision Based Inspection on Post-Earthquake With UAV Synthetic Dataset,” in IEEE Access, vol. 10, pp. 108134-108144, 2022

[4] Isaac-Medina, Brian KS, Matt Poyser, Daniel Organisciak, Chris G. Willcocks, Toby P. Breckon, and Hubert PH Shum. ”Unmanned aerial vehicle visual detection and tracking using deep neural networks: A performance benchmark” In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pp. 1223-1232, 2021.

[5] T. Ye, W. Qin, Y. Li, S. Wang, J. Zhang and Z. Zhao, ”Dense and Small Object Detection in UAV-Vision Based on a Global-Local Feature Enhanced Network,” in IEEE Transactions on Instrumentation and Measurement, vol. 71, pp. 1-13, 2022

[6] A. Abdollahi, B. Pradhan, S. Gite and A. Alamri, ”Building Footprint Extraction from High Resolution Aerial Images Using Generative Adversarial Network (GAN) Architecture,” in IEEE Access, vol. 8, pp. 209517-209527, 2020

[7] Albaba, Berat Mert, and Sedat Ozer, ”SyNet: An ensemble network for object detection in UAV images” in 25th IEEE International Conference on Pattern

Recognition (ICPR), pp. 10227-10234, 2021

[8] H. Wang, Y. Yu, Y. Cai, X. Chen, L. Chen and Y. Li, ”Soft-Weighted-Average Ensemble Vehicle Detection Method Based on Single-Stage and Two-Stage Deep Learning Models,” in IEEE Transactions on Intelligent Vehicles, vol. 6, no. 1, pp. 100-109, 2021

[9] A. Bouguettaya, H. Zarzour, A. Kechida and A. M. Taberkit, ”Vehicle Detection From UAV Imagery With Deep Learning: A Review” in IEEE Transactions on Neural Networks and Learning Systems, vol. 33, no. 11, pp. 6047-6067, Nov. 2022

[10] Rui, Xue, Yang Cao, Xin Yuan, Yu Kang, and Weiguo Song., ”Disas-terGAN: Generative Adversarial Networks for Remote Sensing Disaster Image Generation,” in MDPI Remote Sensing 13, no. 21: 4284, 2021

[11] J. Dong, L. Zhang, H. Zhang and W. Liu, ”Occlusion-Aware GAN for Face De-Occlusion in the Wild,” in IEEE International Conference on Multimedia and Expo (ICME), London, UK, pp. 1-6, 2020

[12] T. Chowdhury and M. Rahnemoonfar, ”Attention Based Semantic Seg-mentation on UAV Dataset for Natural Disaster Damage Assessment” 2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS, pp. 2325-2328, 2021

[13] Lyu, Ye, George Vosselman, Gui-Song Xia, Alper Yilmaz, and Michael Ying Yang., ”UAVid: A semantic segmentation dataset for UAV imagery.,” in ISPRS journal of photogrammetry and remote sensing 165, pp. 108-119, 2020

[14] T. Chowdhury, M. Rahnemoonfar, R. Murphy and O. Fernandes,

”Comprehensive Semantic Segmentation on High Resolution UAV Imagery for Natural Disaster Damage Assessment,” in IEEE International Conference on Big Data (Big Data), Atlanta, GA, USA, 2020, pp. 3904-3913, 2020

[15] M. Axelsson, M. Holmberg, S. Serra, H. Ovr ́en and M. Tulldahl, ”Semantic labeling of lidar point clouds for UAV applications,” in IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Nashville, TN, USA, pp. 4309-4316, 2021

[16] H. Ren et al., ”Swarm UAV SAR for 3-D Imaging: System Analysis and Sensing Matrix Design” in IEEE Transactions on Geoscience and Remote Sensing, vol. 60, pp. 1-16, 2022

[17] T. C. Bybee and S. E. Budge, ”Method for 3-D Scene Reconstruction Using Fused LiDAR and Imagery From a Texel Camera” in IEEE Transactions on Geoscience and Remote Sensing, vol. 57, no. 11, pp. 8879-8889, Nov. 2019

[18] Rahnemoonfar, Maryam, Tashnim Chowdhury, and Robin Murphy. ”RescueNet: A High-Resolution Post Disaster UAV Dataset for Semantic Segmentation.” UMBC Student Collection, 2021

[19] Li, Tianjiao, Jun Liu, Wei Zhang, Yun Ni, Wenqian Wang, and Zhiheng Li. ”Uav-human: A large benchmark for human behavior understanding with unmanned aerial vehicles” In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 16266-16275, 2021

[20] M. Dai, T. H. Luan, Z. Su, N. Zhang, Q. Xu and R. Li, ”Joint Channel Allocation and Data Delivery for UAV-Assisted Cooperative Transportation Communications in Post-Disaster Networks,” in IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 9, pp. 16676-16689, 2022