Real-time Object Detection for Disaster Management from Aerial Imagery

FINAL YEAR PROJECT REPORT

By

Maryam Sana

Siraj Qazi

In Partial Fulfilment

Of the Requirements for the degree

Bachelor of Engineering in Electrical Engineering (BEE)

School of Electrical Engineering and Computer Science

National University of Sciences and Technology

Islamabad, Pakistan

May 10, 2021

Declaration

We hereby declare that this project report entitled *Real-time Object Detection for Disaster Management from Aerial Imagery* submitted to the Department of Electrical Engineering, is a record of an original work done by us under the guidance of Prof. Dr. Faisal Shafait and Dr. Adnan ul Hassan, and that no part has been plagiarized without citations. This project work is submitted in the partial fulfilment of the requirements for the degree of Bachelors of Engineering in Electrical Engineering (BEE)

Team Members

Maryam Sana

Maryam Sana

Siraj Qazi

Supervisors

Dr. Faisal Shafait

_____

Dr. Adnan ul Hassan

_____

Dedication

We dedicate this work to Allah Almighty, our parents, and our advisors Dr. Faisal Shafait and Dr. Adnan ul Hassan.

Acknowledgements

We are sincerely thankful to our advisor Dr. Faisal Shafait and co-advisor Dr. Adnan ul hassan for their valuable suggestions, insights and collaboration throughout the course of this research project. It would not have been possible to comprehensively conclude this research without their expert advice and unconditional support. We would especially like to express our deep and sincere gratitude towards Dr. Faisal Shafait for inspiring us with his dynamism, vision and motivation towards deep learning research.

We would also like to acknowledge the support of Dr. Mohsin Ghaffar (Technische Universität Kaiserslautern, Kaiserslautern, Germany) and thank him for help, collaboration and guidance in our research activities.

Contents

ABSTRACT

Each year, natural calamities and disasters like earthquakes, landslides, flood and typhoon impact broad areas of the world, especially residential areas, and man-made structures. In order to decrease the negative impacts of such events, accessibility to real-time and accurate geospatial information of degraded areas is vital at early stages of response. Unmanned Aerial Vehicles (UAVs) are recognized as an efficient data acquisition platform. UAVs are able to gather high resolution imagery because of its cost-effectiveness, ability to fly at lower altitudes, and ability to enter a hazardous area. High resolution imagery of a disaster affected area at an early stage could be used to extract indispensable geospatial information for disaster management and response.

With deep learning-based image classification becoming more powerful each year, it is apparent that its introduction to disaster response will increase the efficiency that responders can work with. We seek to apply deep learning (DL) techniques in disaster management and response. Real time and accurate geospatial information gathered from the analysis of the affected areas using CNNs could be a crucial step towards improvement in disaster response systems. A dedicated Aerial Image Database for Emergency Response (**AIDER**) is used for Image classification. Furthermore, A novel dataset named as **AIDER-Detect** is introduced for object detection in emergency response situations.

To expand research concerning deep learning algorithms for remote sensing with state-of-the-art accuracy and accelerated inference for classification, we have proposed two custom architectures. In our approach, we use dilated convolutions with concatenation fusion to overcome the resolution imbalance in data. The deep learning models trained in PyTorch with label smoothing are calibrated with different quantization schemes to build an optimized computation graph using NVIDIA TensorRT Deep Learning SDK. A custom engine is used for accelerated inference of our optimized graph on embedded devices with CUDA-capable GPUs like the NVIDIA Jetson TX2 Module.

Numerous object detection algorithms have been proposed in the last decades, though this work is limited to the study of region proposal networks and end-to-end object detection algorithms. Based on the statistical results gathered by our experiments with the AIDER Detect dataset, we conclude that the YOLOv4-Tiny architecture provides the best trade-off between detection score (**44.1% mAP**) and performance (**14.3 FPS**) on the NVIDIA Jetson TX2 embedded module.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Project Overview

Unmanned Aerial Vehicles (UAVs), equipped with camera sensors can facilitate enhanced situational awareness for many emergency response applications since they are capable of operating in remote and difficult to access areas. “Real-time object detection for disaster management on aerial imagery using atrous convolutions " aims to measure impact of disaster and localize disaster victims to provide rapid assistance using aerial imagery.

1.2 Purpose

The aim of this project is to overcome challenges in disaster management:

- **Improve disaster impact assessment using aerial imagery**

Every year disasters cause widespread social, economic, human and infrastructure damage. Timely and accurate assessment of impacts of disaster can help levitate effects of disaster on an area.

- **Avoid connectivity issues**

Communication between rescue teams discontinues due to breakdown caused by disaster. Networking between rescue teams is essential for effective management for disaster response.

- **Locate critically injured victims**

Critically injured victims need rapid medical assistance, rescue teams fail to locate injured disaster victims timely due to hazardous conditions and debris spread across the area.

1.3 Objectives

Rapid disaster management by computer vision techniques is the main goal of this project. Following are listed as objectives of the project:

- **Real time classification**

One of the main objectives of this project is to classify disaster in a set of specified classes. Identification of the disaster using aerial imagery with an accelerated deep neural network in order to faster mitigate their effects on environment and on human population.

- **Real time Object Detection**

Object detection on aerial imagery of disaster struck area can raise enhanced situational awareness that can provide valuable tool for emergency response and disaster management applications. Detecting humans and vehicles with substantial accuracy and mean average precision is one of the main objectives of this project.

- **Accelerated Computing**

High performance state of the art classification and object detection models cost a lot of computational resources and time. Our aim is to reduce resource requirements for these models while attaining state of the art results. Accelerated performance can help us meet the constraints of real time and provide information to rescue teams for rapid assistance.

- **Hardware Deployment**

Integration and deployment of proposed accelerated models on drone compatible embedded platforms for on board processing.

1.4 Emergency Response System Overview

We propose an emergency response system based on the state-of-the-art deep learning approach to effectively respond to emergency situation created by disasters. A comprehensive overview of the system is shown below in Figure 1.1.

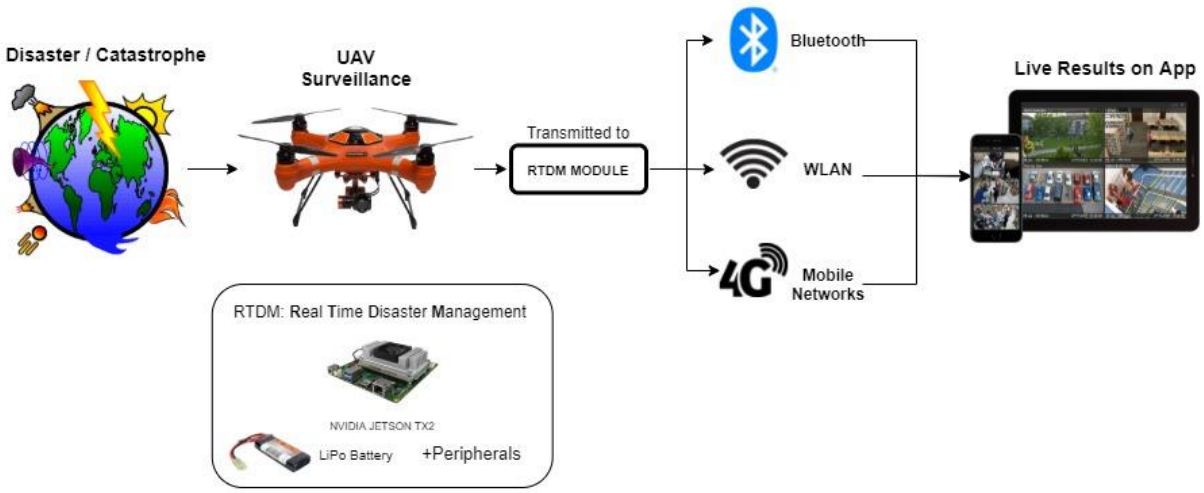

Figure 1.1 Emergency response system overview

As shown in the Figure1.1 our proposed system will create a disaster aware map for users after the processing of aerial imagery. The aerial imagery is processed and analysed via a RTDM – Real Time Disaster Management module on Nvidia jetson TX2.

The architecture of RTDM is shown in the Figure1.2

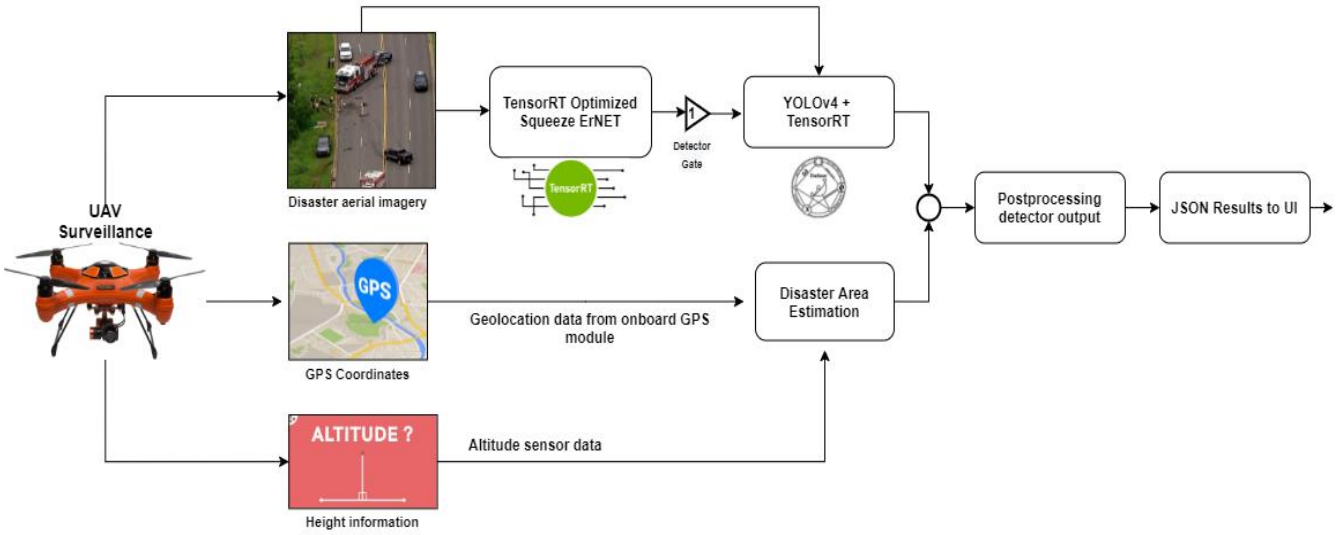

Figure 1.2 RTDM-Real Time Disaster Management module flow diagram

1.5 Components of Emergency Response System

Our emergency response system primarily performs two functions for creating a disaster aware map indicting the area affected and number of disaster victims.

- Disaster-based classification of area
- Victim localization in disaster area

Chapter 2

Disaster-Based Classification of Area

Efficient visual processing of aerial data captured by UAV, can classify the area as a disaster zone and measure the impact of disaster depending upon the area it has affected. Disaster based classification of an area has to be efficient for real time processing as well as compact in terms of space and memory requirements to be able to operate on an embedded platform mounted on UAV. UAVs functioning in difficult to access areas cannot transmit large amounts of visual data for classification and detection so on-board processing is required. We have used convolutional neural networks for disaster detection and classification in aerial imagery.

Chapter 3

Literature Review on Disaster Detection and Classification

3.1 Image Classification Model

Image classification plays an important role in remote sensing images and is used for various applications such as environmental change, agriculture, land planning, urban planning, surveillance, geographic mapping, disaster control and object detection. CNNs are an often-used architecture for deep learning and have been widely used in computer vision. Deep convolutional neural networks provide better results than existing methods in the literature due to advantages such as processing by extracting hidden features, allowing parallel processing and real time operation. Figure3.1 illustrates a simple CNN architecture consisting of convolutional and fully connected layers. The concept of convolutions in the context of neural networks begins with the idea of layers consisting of neurons with a local receptive field, i.e., neurons which connect to a limited region of the input data and not the whole.

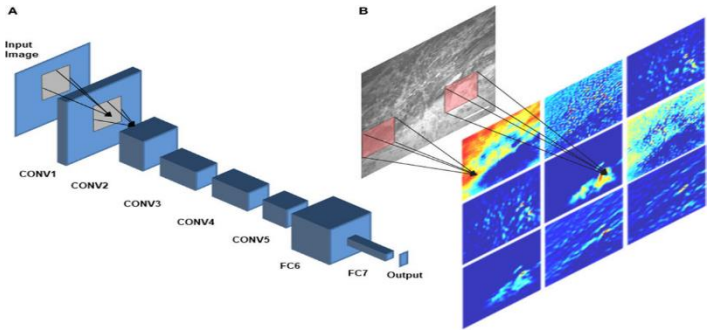

Figure 3.1 Deep Convolutional Neural Network

Fuelled by the need to perform even better in image classification tasks such as the ImageNet Large Scale Visual Recognition Competition (ILSVRC), some of the most important architectures are highlighted next, whose components and ideas will be used to develop an efficient CNN for embedded aerial scene classification with UAVs.

The VGG16 [1]has become a popular choice when extracting CNN features from images. This network contains 16 CONV/FC layers and appealingly, is characterized by its simplicity. It is comprised only of 3x3 convolutional layers stacked on top of each other with an increasing depth of 2 with pooling layers in between to reduce the feature map size by a factor of 2; and with 2 fully connected layers at the end, each with 4096 neurons. A final dense layer is equal to the number of classes is used for the final classification. A downside of the VGGNet is that it is more expensive to evaluate and uses a lot of parameters and consequently memory (~140MB).

ResNet [2] introduced the idea of residual learning to train even deeper CNNs, where the input to a convolution layer is propagated and added to the output of that layer after the operation, thus the network effectively learns residuals. However, it's gain in accuracy comes at a cost of each memory demands as well as execution time. (~102MB)

Utilizing the idea of separable convolutions, MobileNets [3] manage to offer reduced computational cost with slight degradation in classification accuracy. It applies a single filter at each input channel and then linearly combines them. Thus, its design can be easily parametrized and optimized for mobile applications.

Various CNN architectures can be used for disaster image classification. Figure3.2 illustrates the various deep learning architectures including ResNET and Inception. The overall objective of this process is to explore the performance and accuracy trade-offs between these networks.

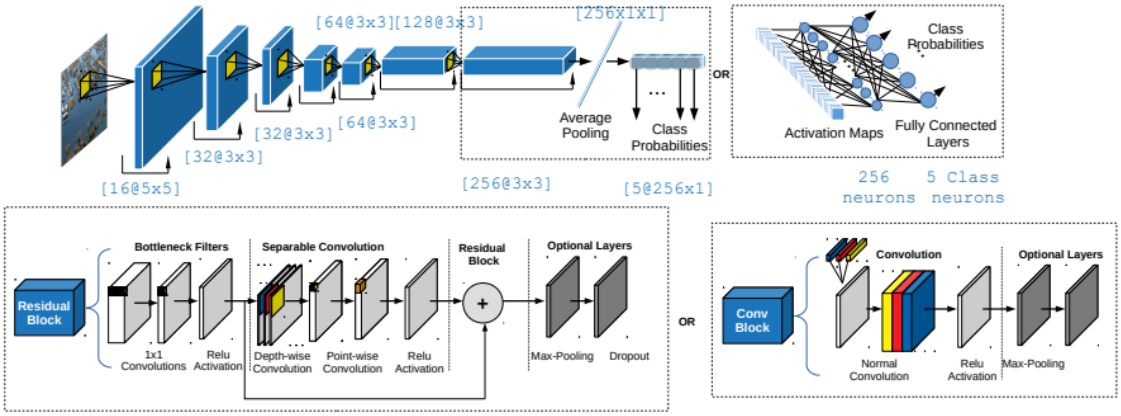

Figure 3.2 Different configurations for setting up a CNN model for aerial disaster classification

3.2 UAV data classification approach

Many image processing and analysis techniques have been developed to aid the interpretation of remote sensing images and to extract as much information as possible from the images. Prior to data analysis, initial processing on the raw data is usually carried out to correct for any distortion due to the characteristics of the imaging system and imaging conditions. The ability of deep convolutional neural networks to learn complex visual characteristics offer a method to classify remotely sensed images via a UAV of different spatial resolution. RGB Imagery of planes, rivers, flooded areas, collapsed buildings, road accidents and burnt areas have complex set of features to be considered for disaster classification. Convolutional Neural Networks with substantial depth and residual connections can be used to learn the pattern for disaster detection.

3.3 Countering class imbalance approach

Imbalanced classification is the problem of classification when there is an unequal distribution of classes in the training dataset. Effective classification with imbalanced data is an important area of research, as high-class imbalance is naturally inherent in many real-

world applications, e.g., fraud detection and cancer detection. Moreover, highly imbalanced data poses added difficulty, as most learners will exhibit bias towards the majority class, and in extreme cases, may ignore the minority class altogether. Methods for handling class imbalance in machine learning can be grouped into three categories: data-level techniques, algorithm-level methods, and hybrid approaches. Data-level techniques attempt to reduce the level of imbalance through various data sampling methods [4]. Algorithm-level methods for handling class imbalance, commonly implemented with a weight or cost schema, include modifying the underlying learner or its output in order to reduce bias towards the majority group. Finally, hybrid systems strategically combine both sampling and algorithmic methods

3.4 Feature Fusion for variable spatial resolution

Variable spatial resolution in training data makes it hard for neural network to generalize well on data. Receptive field of disaster affected area in the dataset is variable and achieving a high accuracy on this dataset is hard. Using variable receptive fields for this in parallel can help solve the problem of generalizing well on variable spatial resolution images. Feature fusion of output of image with different receptive fields can help us retain spatial information with different perspectives. Though feature fusion may lead to extensive computation and memory requirements due to increased parameters.

3.5 Dilated convolutions for parameter reduction

Dilated convolutions were introduced in the paper "Multi-Scale Context Aggregation by Dilated Convolutions [5]. Dilated convolution is a way of increasing receptive view (global view) of the network exponentially and linear parameter accretion. With this purpose, it finds usage in applications that care more about integrating knowledge of the wider context with less cost.

A **KxK** convolution with stride S is the usual sliding window operation, but at every step the window is moved by **S** elements. The elements in the window are always adjacent elements in the input matrix. For S=1, it is the standard convolution. For S>1 a down-sampling effect is obtained. This operation can also be generalized to 0<S<1 (fractionally strided convolution) in which case, an up-sampling effect is obtained.

A **D-dilated KxK** convolution is different, called "convolution with dilated filter", because it is equivalent to dilating the filter before to do the usual convolution. Dilating the filter means expanding its size filling the empty positions with zeros. Figure3.3 shows effect of different dilations on receptive field of a kernel. In practice, no expanded filter is created; instead, the filter elements (the weights) are matched to distant (not adjacent) elements in the input matrix. The distance is determined by the dilation coefficient D. The image below shows how the kernel elements are matched to input elements in a D-dilated 3x3 convolution (when the

canter of the kernel is aligned to the centre of the input matrix). Note that for D=1 the standard convolution is obtained.

The intuition is to take advantage of the different dilation rates since one path may peek up features that another may have missed due to changes in region resolution. It is important to note that the weights are not shared between paths and each learns different weights that may be more useful.

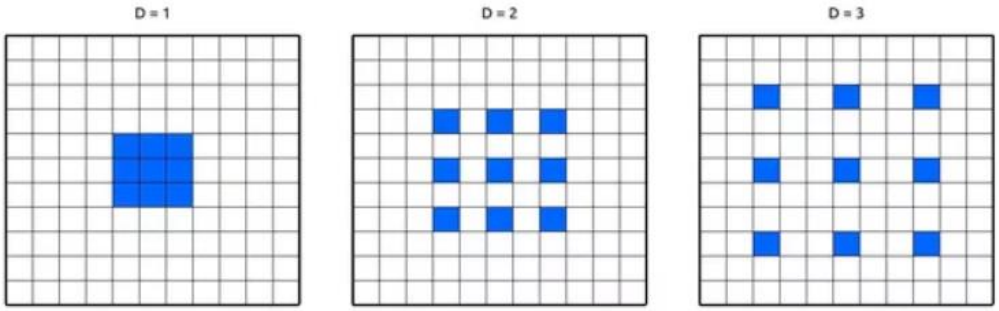

Figure 3.3 a) Dilation 1 b) Dilation 2 c) Dilation 3

Receptive field

3.6 Quantized Neural Networks

Neural networks in python frameworks are generally trained in floating point of bit width 32. These floating-point calculations take a lot of computational resources as well as contain a substantial amount of redundant information. By converting weights and activations into reduced precision we can convert networks into quantized neural networks. Quantized neural networks can accelerate a neural network to a very large extent with a little accuracy drop. QNNs drastically reduce memory size and accesses and replace most arithmetic operations with bit-wise operations [6]. Such networks are very useful when we have to deploy our deep learning solutions to embedded platforms with limited resources.

Chapter 4

Disaster Classification Dataset

4.1 Aerial Imagery Dataset for Emergency Response (AIDER)

A dedicated dataset for disaster classification has been used, referred to as AIDER (Aerial Image Dataset for Emergency Response Applications) [7]. The dataset construction involved manually collecting all images for four disaster events, 320 images of Fire/Smoke, 370 images for Flood, 320 images for Collapsed Building/Rubble, and 335 images for Traffic Accidents, as well as 1200 images for the Normal case. Visually similar images such as for example active flames and smoke are grouped together. The aerial images of 4 disaster classes were collected from multiple sources such as the world-wide-web (e.g., google images, Bing images, YouTube, news agencies websites, etc.), other databases of general aerial images, and images collected using a UAV platform (DJI Matrice 100 UAV). During the data collection process, the various disaster events were captured with different resolutions and under various conditions with regards to illumination and viewpoint. Figure4.1 shows an overview of different classes of AIDER.

No table of figures entries found.

Figure 4.1 Aerial Imagery Dataset for Emergency Response Random Samples

4.1.1 Specifications

Specifications of AIDER are listed below:

Total Images: **6,433**

Total Classes: **5**

Collapsed Building, Fire, Flood, Traffic Accidents, Normal

Individual class image count:

Collapsed Building: **511**

Fire: **521**

Flood: **526**

Traffic Accidents: **484**

Normal: **4390**

Resolution: **Variable**

Total Size: **263 MiB**

Chapter 5

Custom Architecture Development

After taking into account the previously describe state of the art deep learning image classification approaches, we extend the design space and focus on new features. A custom architecture – variant of EmergencyNet proposed in [7] has been developed for improved performance. Our image classification architecture consisted of atrous convolution blocks with concatenation fusion, Dense – Fully connected layers and label smoothing at input level. Custom architecture was trained with different configurations for further improved performance.

5.1 Atrous Convolution Feature Fusion (ACFF) Module

Atrous convolutions can capture and transform images at different resolutions depending on the dilation rate which determines the spacing between the kernel points, effectively increasing their receptive field without increasing the parameter count. The proposed ACFF block shown in Figure5.1 computes multiple atrous convolutions for the same input map across different dilation rates d. Each atrous convolution is factored into depth-wise convolution that performs light-weight filtering by applying a single convolutional kernel per input channel to reduce the computational complexity. Then, some form of fusion takes place to merge the different features together.

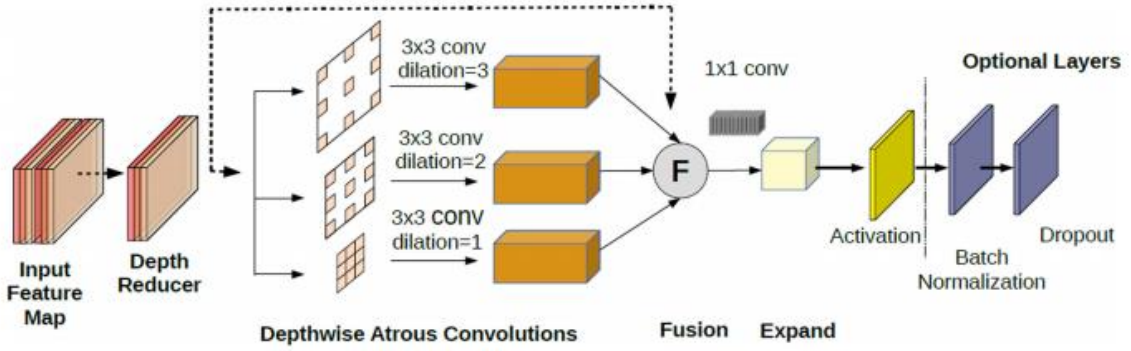

Figure 5.1 ACFF Block basic architecture

5.2 Dense Layer

Fully Connected Layer is simply, feed forward neural networks. Fully Connected Layers form the last few layers in the network. The input to the fully connected layer is the output from the final Pooling or Convolutional Layer, which is flattened and then fed into the fully connected layer.

After passing through the fully connected layers, the final layer uses the SoftMax activation function (instead of ReLU) which is used to get probabilities of the input being in a particular class (classification).

Previous approach suggested for classification of AIDER in Emergency NET [6] does not use Fully connected layer. After exploring different design choices, it was evident that with reduced number of convolutional layers and ACFF block accuracy can be maintained by adding a fully connected or dense layer before SoftMax layer.

While taking consideration the above-mentioned design choices we developed two custom architectures for AIDER image classification in PyTorch.

- Squeeze ErNET
- Squeeze ErNET (Reduced Convolutions)

5.3 Squeeze ErNET

We developed Squeeze ErNET architecture with a combination of ACFF modules and convolutional layers to capture spatial information in our dataset. The topology of squeeze ErNET is specified in Table 1.

Table 1 Squeeze ErNET Topology

Layer	Output	Receptive Field	Filter	Stride
Input Image	140 × 140			
Conv 1	69 × 69	3	16	2
ACFF -1	67 × 67	3,5,7	64	1
Pooling -1	33 ×33	2		2
ACFF – 2	31 ×31	3,5,7	96	1
Pooling – 2	15 ×15	2		2
ACFF – 3	13 ×13	3,5,7	128	1
Pooling – 3	6 ×6	2		2
ACFF – 4	4 ×4	3,5,7	256	1
Pooling – 4		1		1
Conv 2	4 ×4	1	5	1
Global Pooling	2 ×2	5		1
FC – 1	5 ×1			
SoftMax	5 ×1			

5.4 Squeeze ErNET (Reduced Convolutions)

We developed a variant of Squeeze ErNET with reduced parameters. Squeeze ErNET Reduced Convolutions (Henceforth referred to as **RedConv**) tends to perform better than Squeeze ErNET.

The topology of Squeeze ErNET (RedConv) is specified in the Table 2.

Table 2 Squeeze ErNET RedConv Topology

Layer	Output	Receptive Field	Filter	Stride
Input Image	140 × 140			
Conv 1	69 × 69	3	16	2
Reduced Conv 1	69 × 69	1	8	1
ACFF -1	67 × 67	3,5,7	64	1
Pooling -1	33 ×33	2		2
ACFF – 2	31 ×31	3,5,7	96	1
Reduced Conv 2	31 ×31	1	48	1
Pooling – 2	15 ×15	2		2
ACFF – 3	13 ×13	3,5,7	128	1
Pooling – 3	6 ×6	2		2
Reduced Conv 3	6×6	1	64	1
ACFF – 4	4 ×4	3,5,7	256	1
Pooling – 4		1		1
Conv 2	4 ×4	1	5	1
Global Pooling	2 ×2	5		1
FC – 1	5 ×1			
SoftMax	5 ×1			

5.5 Training

All the networks are developed and tested through the same training framework so as to have the same conditions and a fair comparison during the inference phase. PyTorch deep learning framework was used for evaluation and training.

Data was pre-processed before training keep the image size to 240 into 240 and centre crop transformations. Image normalization was tested as transform but did not give any significant results.

Training hyperparameters are listed in Table 3:

Parameter	Value
Epochs	500
Learning rate	0.0001
Loss function	Cross entropy with label smoothing
Batch size	128
Optimizer	Adam
Momentum	0.5
Scheduler Step Size	0.0001

Table 3 Image Classification Training Hyper-parameters

Chapter 6

Fine Tuning Disaster Classification Model

Fine tuning is required to attain maximum accuracy. The process of setting right hyper parameters is an extensive process in the deep learning world. Finding the best configuration in the high dimensional hyper parameter space is quite non trivial. We included weight decay, depth of model, momentum, batch size, learning rate in our optimization space.

There are three basic approaches followed for hyper parameter tuning of deep learning models.

- Grid search
- Random search
- Bayesian optimization

6.1 Grid Search

Every possible combination of parameters is searched and applied in grid search. Eventually grid search records the parameters which give the most optimal results. This procedure involves defining a search space for optimization. Grid search is a popular method in hyperparameter tuning of simple ML algorithms; however, high computational complexity in complex ML algorithms such as Deep Neural Networks (DNN) is the main barrier towards its practical implementation [8].

6.2 Random search

Random search involves the process of defining a hyper parameter search space and further randomized sample points are chosen for finding the most optimal configuration. Python frameworks like Optuna, scikit learn provided support for hyper parameter tuning. We have used Optuna randomized search for our hyper parameter tuning. See Figure6.1 for difference between grid and random search space for parameter optimization.

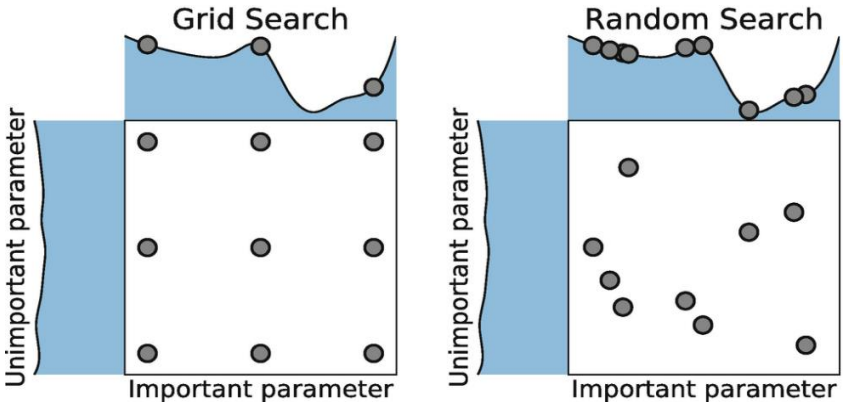

Figure 6.1 Parameter optimization curves and search space with grid search vs random search

6.3 Optuna Hyperparameter Optimization

We have used Optuna randomized search for our hyper parameter tuning. We explored the best set of optimizers, momentum, epochs, batch size and learning rate for the optimization of our proposed architectures. We set loss as our minimizing objective function in the test experiments. Figure6.2 shown below can highlight the results optimization.

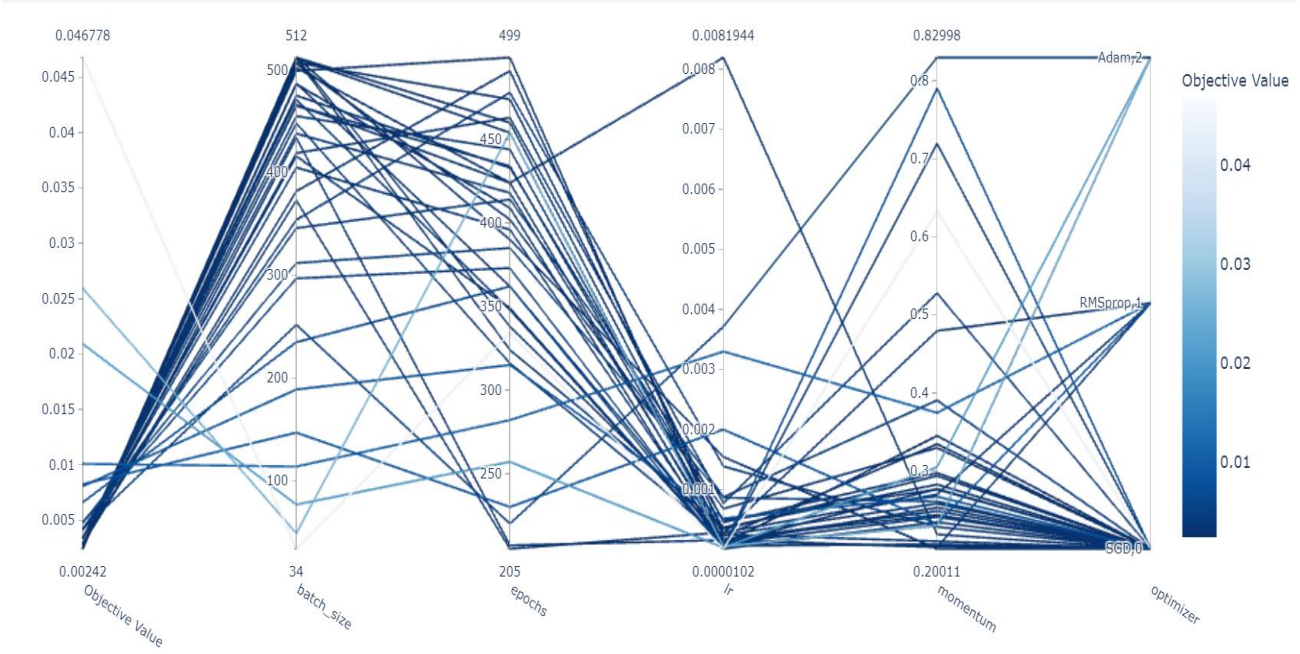

Figure6.2 Parallel Coordinate plot of optimization space

Chapter 7

Quantization

7.1 Reduced Precision Networks

A significant redundancy lies in classification convolutional neural networks. By removing this redundancy, we can accelerate our neural networks and reduce not only time cost but also space cost of many efficient solutions. Reduced precision neural networks are answer to many real time problems in deep learning world. Networks with weights and activation restricted to a low precision are capable of having small memory footprint and performing low precision arithmetic operations.

We have tried our custom architecture with three precision configurations:

- **FP32 (32-bit Floating Point or "Full" Precision)**
- **FP16 (16-bit Floating Point or "Half" Precision)**
- **INT8 (8-bit Integer Quantization)**

INT8 proves to be much faster than other configurations but FP16 is the most optimal configuration with memory space optimization, accelerated inference and maximum accuracy.

7.2 Layer Fusion

Dynamic computation graph of every neural network determines the calculation flow in the forward pass. A compressed dynamic computation graph can improve the speed of inference without affecting the results. Compression of computation graph can be obtained by fusing different layers in the graph.

Example of DCG shown in Figure7.1 and 7.2 illustrate reduced of complexity after horizontal layer fusion.

Different transforms and optimizations are applied to parameters and layers after training to compressed the computation graph into a smaller and compact most structure.

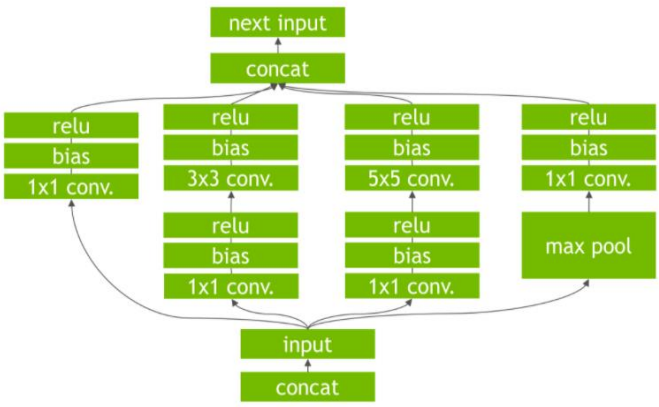

Figure 7.1 A DCG of neural network before layer fusion

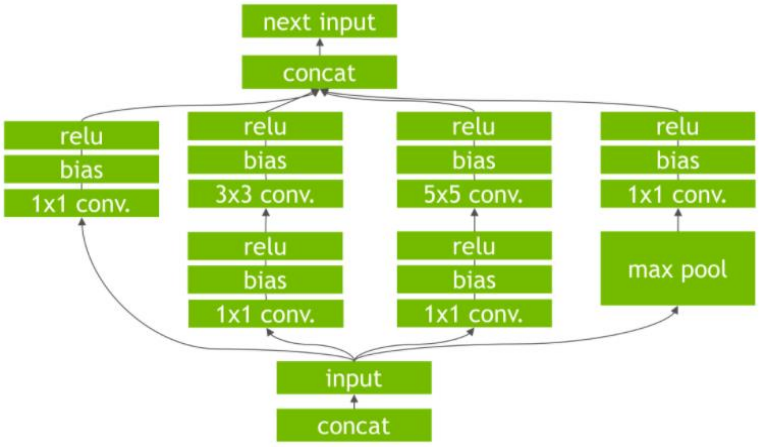

Figure 7.2 DCG after layer fusion

7.3 TensorRT Optimizations

To optimize a model for inference, TensorRT takes the network definition, performs optimizations including platform-specific optimizations, layer optimizations and generates the inference engine. Figure7.3 shows the flow sheet of TensorRT engine generation.

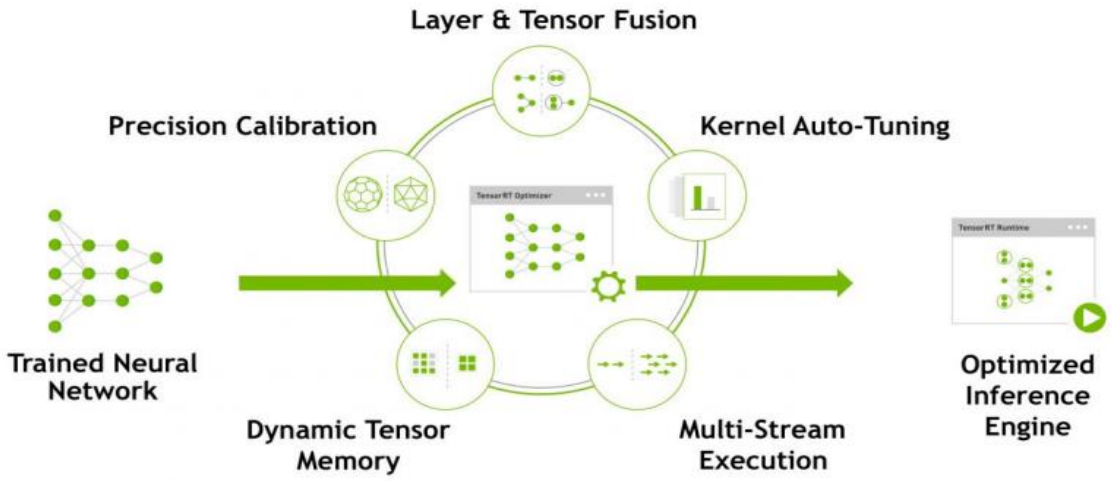

Figure 7.3 TensorRT Optimization Cycle

The build phase performs the following optimizations on the layer graph:

- Elimination of layers whose outputs are not used
- Elimination of operations which are equivalent to no-op
- The fusion of convolution, bias and ReLU operations
- Aggregation of operations with sufficiently similar parameters
- Merging of concatenation layers

TensorRT optimization of our network led to an accelerated inference rate with a little drop in our F1 score and accuracy. Low precision inference by TensorRT reduces power consumption and energy/frame. Energy efficiency is a vital part of remote sensing applications and we have significantly reduced the energy requirements of our application via low precision processing.

Chapter 8

Evaluation & Performance Metrics I

The quality of results from a statistical or a machine learning model is quantified using appropriate evaluation metrics. Commonly used evaluation metrics for classification include **Accuracy**, **Precision**, **Recall** and the **F1 Score**. Among the most popular metrics for testing object detection models are the **Intersection over Union (IoU), Average Precision (AP)** and the **mean Average Precision (mAP) Score**. Furthermore, considering the two specificities of our project – real-time inference and deployment on embedded systems – we also record two additional performance metrics - classification/detection inference rate in **Frames Per Second (FPS)** and the **memory footprint** (in Megabytes) of our deep learning models.

Before we proceed to explain each evaluation metric, following terms are significant:

I. **True Positive (TP):**
A true positive is an outcome where the model *correctly* predicts the *positive* class.

II. **True Negative (TN):**
A true negative is an outcome where the model *correctly* predicts the *negative* class.

III. **False Positive (FP):**
A false positive is an outcome where the model *incorrectly* predicts the *positive* class.

IV. **False Negative (FN):**
A false negative is an outcome where the model *incorrectly* predicts the *negative* class.

8.1 Classification Metrics

i. **Accuracy**

Classification accuracy refers to the ratio of the number of correct predictions to the total number of input samples - the proportion of true results among the total number of cases examined. Its well-suited for binary as well as a multi-class classification problem.

$$Accuracy = \frac{Correct\ Predictions}{All\ Predictions} \quad (1)$$

In terms of positives and negatives, accuracy can be expressed as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (2)$$

ii. Precision

Precision refers to the portion of predicted positives that are truly positive i.e., the fraction of true positives out of all the positives (predicted belonging to a certain class):

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \quad (3)$$

iii. Recall

Recall quantifies the proportion of the actual positives that was predicted correctly i.e., the positives correctly classified by the model out of all the actual positives:

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \quad (4)$$

Therefore, a model that produces no false negatives (does not miss a single positive example) has a recall of 1.0 (or 100%).

iv. F1 Score

The F1 score maintains a balance between precision and recall, and is given as the harmonic mean of the two:

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (5)$$

8.2 Performance Metrics

i. Inference Rate (Frames Per Second)

Inference rate is the rate at which a classifier or detector is capable of processing incoming camera frames, where t_i is the processing time of a single image, is given as:

$$FPS_{model} = \frac{1}{N_{test\_samples} \times \sum_{i=1}^{N_{test\_samples}} (t_i)} \quad (6)$$

ii. Memory Footprint (Model Size)

Memory footprint refers to the size of the deep learning model – the memory it claims from the OS when trying to perform a single forward pass through the trained network. Since we've used PyTorch as our primary deep learning framework, the results reported are generated from tools that evaluate PyTorch models – such as torch-summary.

iii. Energy per frame (mJ/f)

Energy efficiency is a vital part of remote sensing applications. Energy per frame refers to the total amount of energy required by the GPU and CPU cores to process one image.

Here we can calculate energy consumed/frame by the following formula:

$$\frac{E}{F} = \frac{Total\ Energy\ Consumed}{\%test - data \times 6433} \quad (7)$$

Chapter 9

Test Platforms

Our results have been achieved using the following CUDA-accelerated test platforms:

9.1 NVIDIA Jetson TX2 Development Kit

NVIDIA® Jetson™ TX2 is an embedded AI computing device. Each supercomputer-on-a-module brings true AI computing to the edge with an NVIDIA Pascal™ GPU, up to 8 GB of memory and 59.7 GB/s of memory bandwidth, and a wide range of standard hardware interfaces.

Figure 8.1 Nvidia Processing Core

Technical Specifications

- **GPU** 256-core *NVIDIA Tegra X2* - Pascal™ GPU architecture with 256 NVIDIA CUDA cores
  - Dual-Core *NVIDIA Denver 2* 64-bit CPU
  - Quad-Core *ARM® Cortex®-A57* MPCore
- **Memory**
  - *8GB* 128-bit LPDDR4 Memory
  - *1866 MHz* – 59.7 GB/s
- **Storage** *32GB eMMC 5.1*
- **Power** *7.5W* / 15W
- **OS** *Ubuntu 18.04* LTS Flashed with NVIDIA JetPack 4.4.2

9.2 HP Omen 15-dc1064tx

We also tested our models on a high-performance laptop with NVIDIA CUDA-capable GPU for comparison with the TX2 performance.

Technical Specifications

- **GPU** *NVIDIA GTX 1660 Ti* – NVIDIA Turing™ GPU architecture with 1536 CUDA cores and ***6GB*** *GDDR6* memory with 192-bit Interface @ 288GB/s 90W
- **CPU** *Intel® Core™ i5-9300H* 9th Generation Hexacore (8M Cache, Up to 4.10 GHz)
- **Memory** *16GB* DDR4-2666 MHz
- **Storage** *128GB NVMe* M.2 SSD + *1 TB SATA* 5400 RPM HDD
- **Power** *200W* AC Adapter
- **OS** *Ubuntu 18.04* LTS with Linux Kernel 5.4.0-generic

Chapter 10

Disaster Classification Results

Our results are categorized into three major divisions:

- Results with PyTorch
- Results with PyTorch + TensorRT (FP32)
- Results with PyTorch + TensorRT (FP16)

10.1 Classification Results

0. State-of-the-art-results (From IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 2020) [7]

MODEL	***ACCURACY***	***F1 SCORE***	***PARAMETERS***	***FPS (Cortex A57)***
EmergencyNET	Not-given	95.6	90,040	25.7

Table 4 EmergencyNET state-of-the-art results

i. Results with PyTorch

MODEL	***ACCURACY***	***F1 SCORE***	***PARAMETERS***	***MEMORY***	***FPS (TX2)***	***FPS (GTX 1660Ti)***
SqueezeRNET	92.5%	95.5%	169,241	21.2 MB	160.72	876.74
SqueezeRNET (RC*)	93%	96%	109,569	20.04 MB	142.11	783.29

Table 5 Custom architecture PyTorch evaluation and performance results

ii. Results with PyTorch + TensorRT (FP32)

MODEL	***ACCURACY***	***F1 SCORE***	***PARAMETERS***	***MEMORY***	***FPS (TX2)***	***FPS (GTX 1660Ti)***
SqueezeRNET	88.45%	92.51%	169,241	21.2 MB	416.46	2951.13
SqueezeRNET (RC*)	94.1%	93.281%	109,569	20.04 MB	497.23	2988.54

Table 6 Custom architecture TensorRT floating point 32 precision results

iii. Results with PyTorch + TensorRT (FP16)

MODEL	***ACCURACY***	***F1 SCORE***	***PARAMETERS***	***MEMORY***	***FPS (TX2)***	***FPS (GTX 1660Ti)***
SqueezeERNET	90.06%	90.62%	169,241	21.2 MB	547.46	3430.17
SqueezeERNET (RC*)	93.79%	94.375%	109,569	20.04 MB	569.37	3196.18

Table 7 Custom architecture TensorRT floating point 16 precision results

10.2 Power and Energy Consumption Results

Low precision inference by TensorRT reduces power consumption and energy/frame. Energy efficiency is a vital part of remote sensing applications and we have significantly reduced the energy requirements of our application via low precision processing.

Figure10.1 and Figure10.2 shows sample graphs of total power consumption for Jetson TX2 while processing

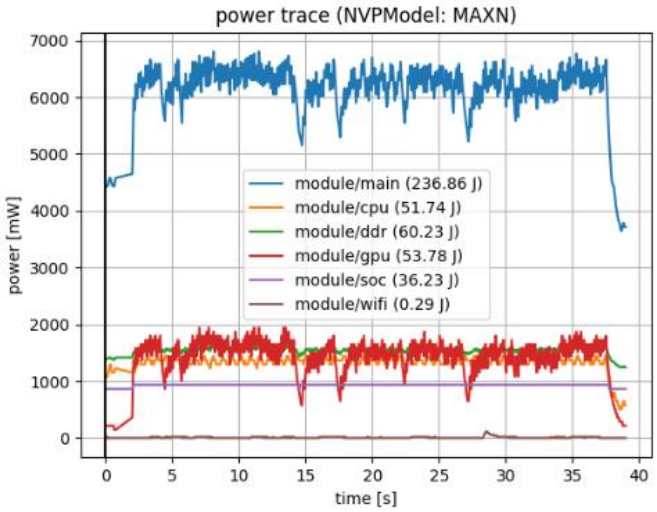

Figure 10.1 Squeeze ErNET FP16 Mode with 45 percent test data Power Consumption Trace

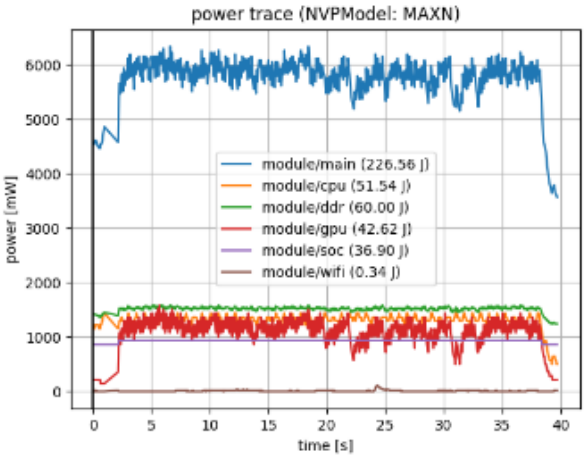

Figure 10.2 Squeeze ErNET RedConv FP16 Mode with 45 percent test data Power Consumption Trace

We have calculated energy consumption for our tested models on TX2 using (7). Table 8 shows energy per frame results of Squeeze Ernet and Squeeze Ernet RedConv.

Model	**Energy(mJ)/Frame**
Squeeze_ErNET	1.857
Squeeze_ErNET (RedConv)	1.472

Table 8 Energy Consumption TX2

Chapter 11

Victim Localization in Disaster Area

One vital component of our rapid emergency response system is victim localization in a disaster struck area. Detecting humans and vehicles in a disaster classified image is used to estimate the number of victims stuck in disaster affected area. Rapid assistance can be provided to disaster victims as soon as their location is identified. We have employed object detection techniques to localize disaster victims in image data.

Chapter 12

Object Detection Literature Review

12.1 Object detection approach for UAV Images

Object detection is the combination of object classification and object localization in an image. These two tasks combined are quite challenging. There are several approaches developed and tested by computer vision community for object detection.

The most commonly used approaches for object detection are region proposal-based networks and you only look once algorithm.

12.2 Object detection using region proposals

Region proposal-based networks have variants named as R-CNN [9] ,Fast R-CNN [10] and Faster R-CNN [11]. Each of them an improved form of the subsequent. R-CNN is a combination of region proposal extraction network along with convolutional layers. With only a small amount of annotated detection data, R-CNN can help localize objects using a deep network and train a high-capacity model. It achieves high object detection accuracy by classifying object proposals with a deep CNN. R-CNN can scale to thousands of object classes without having to rely on estimated techniques like hashing.

R-CNN has some limitations. R-CNN is a computation extensive model and takes a lot of time in inference. It cannot be used for real time applications. See Figure12.1 for elaboration on R-CNN architecture.

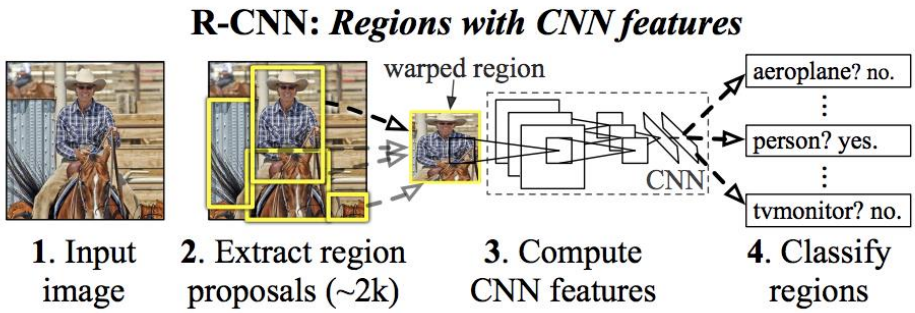

Figure 12.1 R-CNN

Fast R-CNN is a similar approach to R-CNN only a more efficient one. In Fast R-CNN we directly feed input images to CNN instead of region proposals. We define the region of proposals from the convolutional feature map, warp them into squares, and reshape them into a fixed size using a RoI pooling layer so that they can be fed into a fully connected layer and further into SoftMax for prediction.

Both R-CNN and fast R-CNN use selective search which is a time-consuming algorithm for finding region proposal so it's still not efficient enough for real time applications.

12.3 End to End object detection approaches

YOLO [12] is regarded is an end-to-end object detection approach. In YOLO a single CNN is used to predict the bounding boxes around objects and identify their classes. The way YOLO works is that we take a picture and divide it into a SxS grid, with m bounding boxes within each grid. The network generates a class likelihood and bounding box offset values for each bounding box. The bounding boxes with a class probability greater than a threshold value are chosen and used to locate the object in the picture.

12.4 Handling class-imbalanced dataset in object detection

Imbalance problems in computer vision have a significant scope and tend to affect the performance of object detectors and classifiers to a large extent. If there is a substantial disparity in the number of examples relating to various classes, class imbalance occurs [13].

While the foreground-to-background imbalance is the classic example, there is also an imbalance among the foreground (positive) groups. Dataset Imbalance in object detection can be of following types:

- Class Imbalance
- Scale Imbalance
- Spatial Imbalance
- Objective Imbalance

Our dataset AIDER Detect had evident class imbalance and scale imbalance. There are couple of approaches to counter class imbalance and scale imbalance in object detection problems.

Class imbalance in object detection can be countered using weighted average loss, balanced cross entropy and focal loss. Focal loss is the one most commonly used for imbalance problems.

Focal loss is used one stage detectors to counter the problem of extreme class imbalance. The loss function is a dynamically scaled cross-entropy loss, where the scaling factor decays to zero as confidence in the correct class increases.

The focal loss is defined as $FL(p_t)$ where p_t is probability in Equation 8 :

$$\boldsymbol{FL(p_t) = -(1 - p_t)^y log(p_t)} \quad (8)$$

$\boldsymbol{y}$ is referred as *gamma* and or ***modulating factor*** which reduces the loss contribution from easily detected examples.

We have used focal loss and cross entropy loss in our experiments for the finding the optimal solution.

12.5 Self-gated activation functions for object detection

The choice of an activation function is a step of significant importance in training deep neural networks. The most widely used activation function in computer vision is Rectified linear unit – ReLU and its derivative Leaky – ReLU. Recent research shows that a self-gated activation function **swish** which is a derivative of sigmoid tends to work better on certain datasets and on object detection problems [14].

Swish is defined as:

$$\boldsymbol{f(x) = x.sigmoid(x)} \tag{9}$$

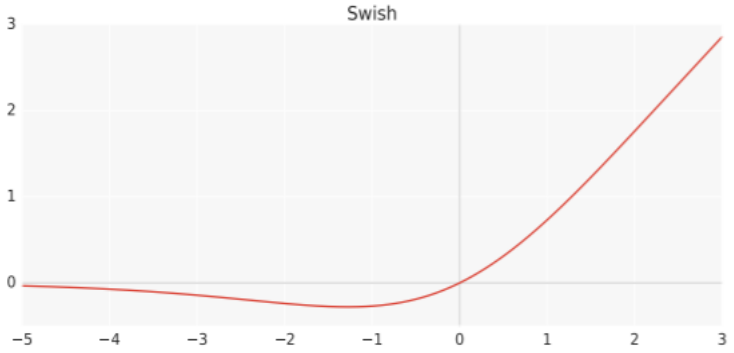

Figure 10.3 Swish activation function

Swish is unbounded on both ends i.e., above and below (See Figure10.3)

Swish congruously outperforms ReLU and its variants in many deep learning models. Unbounded property of swish prevents saturation when training is slow due to near-zero gradients.

Chapter 13

Victim Localization Dataset

We have created a new dataset for object detection on aerial disaster imagery – named as **AIDER Detect.** It contains 2 Classes – **person** and **vehicle** and about 1428 images containing 6900 vehicles and 2100 humans in total.

Annotations files are Pascal VOC format based.

Figure13.1 shows sample image annotations from the AIDER Detect dataset:

Figure 13.1 AIDER Detect sample Images

Chapter 14

Architecture Development

While taking into account the dataset specifications and information gathered on the basis of literature review, we decided to approach the problem of object detection in AIDER-Detect with two design choices for architecture development.

14.1 Region Proposal Network (With Faster R-CNN)

Faster R-CNN tends to produce accurate results with aider detect but not feasible for real time object detection as processing 5 to 8 frames per second. We decided to not proceed with this approach further.

14.2 You Only Look Once (YOLO)

YOLO [12] is a state-of-the-art object detection algorithm. YOLO uses an end-to-end approach where we use one CNN for localization and classification and hence feasible for real-time applications.

YOLO consisted of Yolo layers and a deep CNN backbone which captures all the spatial information. We have experimented with following variants of yolo v3 and yolo v4:

- YOLOv3
- YOLOv3-SPP (Spatial Pyramid Pooling)
- YOLOv3-Tiny
- YOLOv4-Tiny

14.3 YOLO Backbone

We expanded our design space for design choices of YOLO backbones. We used the custom ACFF module mentioned in 5.1 Atrous Convolution Feature Fusion (ACFF) Module.

ACFF module tends to capture spatial information more efficiently for our dataset in image classification. But introducing ACFF Module in YOLO backbone tends to increase our computational cost. Eventually we proceeded with vanilla YOLO Backbone with down sampling blocks and simple convolution blocks proceeded by YOLO layers.

14.4 Data Pre-Processing

Before training, our images are pre-processed for better results. We perform shear, rotation, scale and translation transforms on images and perform augmentations with hue, saturation and value parameters.

Pre-processing hyper parameters are listed below in Table 9.

Parameter	Value
Hue -aug	0.0138

Saturation -aug	0.678
Value -aug	0.36
Scale factor	0.05
Shear factor	0.641
Rotation factor	1.98 degrees
Translation factor	0.05

Table 9 YOLO pre-processing hyperparameters

14.5 Training

We use pre-trained weights from darknet trained on ImageNet **darknet.conv.53** for YOLOv3 and YOLOv3-SPP which has weights for first 53 convolutional layers and **darknet.conv.15** for YOLOv3-Tiny and YOLOv4-Tiny which has first 15 convolutional layer weights. We train these networks for almost 3000 iterations which took 12 to 36 hours depending upon the model and computational platform available. We trained our model on NVIDIA Titan X (12GB).

Training hyperparameters are listed in Table 10:

Parameter	Value
Loss function	Focal loss
Focal Loss Gamma	2
Optimizer	Adam
Learning rate	0.001
Momentum	0.93
Weight decay	0.000484
giou	1.0
IoU threshold	0.5

Table 10 YOLO training hyper-parameters

Chapter 15

Quantization

15.1 Reduced Precision Networks

We can achieve improved inference efficiency on edge devices with quantized networks. Aggressive quantization faces difficulty in complex networks for object detection so we stick to a minimal quantization of up to int 8. Low bit width arithmetic in quantized networks can accelerate the inference on hardware. We subjected our all trained YOLO models to three quantization schemes:

- **FP32 (32-bit Floating Point or "Full" Precision)**

- **FP16 (16-bit Floating Point or “Half” Precision)**
- **INT8 (8-bit Integer Quantization)**

Quantization and network optimization was performed in NVIDIA TensorRT Deep Learning SDK which has been explained in section 3.6 Quantized Neural Networks. We quantize the network backbone and add YOLO layer support to the network for quantization separately.

Quantization of YOLO has significantly reduced the computational burden and energy consumption for inference. The accuracy is also stable and FPS has increased after TensorRT optimizations.

Chapter 16

Evaluation & Performance Metrics II

16.1 Object Detection Metrics

16.1.1 Intersection over Union (IoU)

The IoU evaluates the overlap between the ground truth label bounding box (mask) with the predicted mask – and it is calculated as the ratio of area of intersection between ground truth and prediction to the area of union of the two as:

$$IoU = \frac{Area\,(\,GT\ \cap Pred\,)}{Area\,(\,GT\ \cup Pred\,)} \tag{10}$$

It can also be illustrated as:

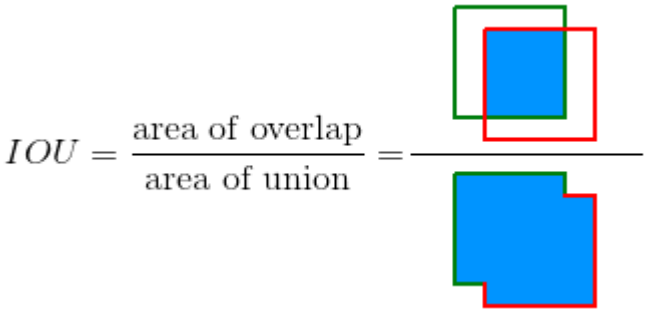

Figure 16.1 IoU definition

Note: *We have recorded our results in terms of the* ***Generalized Intersection over Union (GIoU)*** *which is a new loss and a metric for bounding box regression that improves upon the classical IoU.*

16.1.2 Average Precision (AP)

Average Precision is a standard numerical metric that can be used for comparing the performance of object detectors. It is the value of precision averaged across all (unique) recall levels.

AP is calculated as the area under the (interpolated) precision-recall curve as:

$$AP = \sum_{i=1}^{n-1} (r_{i+1} - r_i) p_{interp}(r_{i+1}) \tag{11}$$

Where p_{interp} is the interpolated precision at a certain recall r_i.

16.1.3 Mean Average Precision (mAP)

The Average Precision (AP) is calculated individually for each class. Thus, the **mean Average Precision (mAP)** is the averaged AP across all classes in an object detection problem.

The mAP across K classes is given as:

$$mAP = \frac{\sum_{i=1}^{K} AP_i}{K} \tag{12}$$

Chapter 17

Victim Localization Results

The following results have been evaluated on the platforms described in **Chapter 9.**

MODEL	***PRECISION***	***RECALL***	***F1 SCORE***	***mAP SCORE***	***FPS @ GTX 1660 Ti***	***FPS @ Nvidia Jetson TX2***
YOLOv3 (Swish Activation + Focal Loss)	0.22	0.538	0.313	0.334 (all) 0.174 (person) 0.493 (vehicle)	10.2	1.4

YOLOv3-Tiny (LeakyRelu + Focal Loss)	0.377	0.267	0.309	0.196 (all) 0.0876 (person) 0.303 (vehicle)	94	12.25
YOLOv4-Tiny (Adam + Focal Loss+ 2yolo layers)	0.221	0.666	0.328	0.441 (all) 0.317 (person) 0.564 (vehicle)	79.7	14.3
YOLOV4 – Tiny (SGD+ Focal loss + 3 Yolo layers)	0.223	0.629	0.334	0.412 (all) 0.278 (person) 0.574 (vehicle)	77.8	15.9

Table 11 Object detection results of AIDER Detect

Chapter 18 Conclusion and Future Work

18.1 Conclusion

Pakistan is one of the most disaster-prone countries in the world. Generally divided into natural and man-made, all disasters are managed by a systematic process of disaster management that aims at minimising the damage and restoration of people to their normal state. Pakistan is well familiar with disasters which have caused a heavy toll in terms of men and material.

However Quick identification and measurement of extent of disaster can help us minimize the loss caused by it. To cater the in efficient disaster response system at Pakistan, an automated emergency response system able to assist rescue teams in identifying disaster struck area and victims has been developed. We have employed state-of-the-art deep learning approaches for assessing the impact of disaster using aerial imagery.

After an analysis of design choices, we have implemented an efficient deep learning system capable of accurately classifying aerial imagery as disaster labelled categories at an

accelerated rate. The proposed solution provides an adequate trade-off between accuracy, inference speed, and complexity. Furthermore, we have introduced the feature of victim localization in our deep learning system which employs You Only Look Once [12] object detection approach which attain a significant mAP score. We have also explored different approaches for attaining maximum accuracy on the highly imbalance emergency response dataset.

The experimental study validates that reducing the inference precision to floating point 16 provides an adequate trade-off between latency and precision metrics.

18.2 Future Work

The results in this thesis lay the groundwork for accelerated classification and object detection for disaster management and real-time response. There are some natural extensions to this work that would help expand and strengthen the results:

i. Custom object detection models for disaster response

Our work presented two custom architectures for classification of disasters, that employ the ACFF (atrous convolution feature fusion) block. Custom architectures for object detection may be developed utilizing the ACFF module for improved detection score as well as real-time performance on the AIDER-Detect dataset.

ii. Testing other SOTA detection models

We have reported our results achieved using the fastest real-time object detection algorithm that is the current industry standard – YOLOv3 and YOLOv4. Other state-of-the-art object detection models such as Single-Shot Detector (SSD), Mobile Net family of architectures, MTCNN etc. may be tested and their performance evaluated on the AIDER Detect dataset to generate a performance comparison and to determine the network topology providing the best accuracy-performance tradeoff.

iii. Development of FPGA-based hardware accelerators for classification and detection

Recent studies on convolutional neural networks [6] suggest that a significant redundancy exists in the precision for weights and activations while using full 64-bit or 32-bit registers. Recently, it has been shown that neural networks can classify accurately using one- or two-bit quantization for weights and activations. Such a combination of low-precision arithmetic and small memory footprint presents a unique opportunity for fast and energy-efficient image classification as well as object detection using Field Programmable Grid Arrays (FPGAs). FPGAs have much higher theoretical peak performance for binary operations compared to floating point, while the small memory footprint

removes the off-chip memory bottleneck by keeping parameters on-chip, even for large networks. FPGAs have been recently adopted for accelerating the implementation of DNNs due to their ability to maximize parallelism as well as due to their energy efficiency. Custom FPGA-based hardware accelerators may be developed for the proposed novel classification architectures to achieve a power-efficient model with real-time performance deployed on embedded SoC FPGAs.

iv. Refining the detection dataset

The dataset can be further expanded and enhanced with additional images and classes in order to achieve a better fit and improve on existing models and techniques. Currently the vehicle class of the dataset outnumbers the 'person' class, which leads to class imbalance problems during training. This can be mitigated by added more images with the person class and reducing background noise in the rest of the images.

Chapter 19

References

[1] A. Z. Karen Simonyan, "Very Deep Convolutional Networks for Large-Scale Image Recognition.," *ICLR,* 2015.
[2] X. Z. Kaiming He, Shaoqing Ren, Jian Sun, "Deep Residual Learning for Image Recognition," 2015.
[3] M. Z. Andrew G. Howard, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," 2017.
[4] M. Rashmi Dubey, 1,2 Jiayu Zhou, BS,1,2 Yalin Wang, PhD,1 Paul M. Thompson, PhD,3 and Jieping Ye, PhD, "Analysis of sampling techniques for imbalanced data: An n = 648 ADNI study," *NeuroImage,* vol. 87, 2013.
[5] V. K. Fisher Yu "Multi-Scale Context Aggregation by Dilated Convolutions”," *ICLR,* 2016.
[6] M. C. Itay Hubara, Daniel Soudry, Ran El-Yaniv, Yoshua Bengio, "Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations," *arXiv e-prints,* 2016.
[7] C. K. a. T. Theocharides, "EmergencyNet: Efficient Aerial Image Classification for Drone-Based Emergency Monitoring Using Atrous Convolutional Feature Fusion," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing,* vol. 13, 2020.
[8] M. A. Amirabadi, "Novel Suboptimal approaches for Hyperparameter Tuning of Deep Neural Network [under the shelf of Optical Communication]," 2019.
[9] J. D. Ross Girshick, Trevor Darrell, Jitendra Malik,” "Rich feature hierarchies for accurate object detection and semantic segmentation," 2014.
[10] R. Girshick, "FAST R-CNN," 2015.
[11] K. H. Shaoqing Ren, Ross Girshick, Jian Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," 2015.

[12] S. D. Joseph Redmon, Ross Girshick, Ali Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2015.
[13] K. Oksuz, B. C. Cam, S. Kalkan, and E. Akbas, "Imbalance Problems in Object Detection: A Review," *IEEE Trans Pattern Anal Mach Intell,* vol. PP, Mar 19 2020, doi: 10.1109/TPAMI.2020.2981890.
[14] B. Z. Prajit Ramachandran, Quoc V. Le, "Swish: a Self-Gated Activation Function," *arXiv e-prints,* 2017.