UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

ADVANCED AVIATION WEATHER RADAR DATA PROCESSING AND REAL-TIME IMPLEMENTATIONS

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

DOCTOR OF PHILOSOPHY

By

JINGXIAO CAI
Norman, Oklahoma

2017

ADVANCED AVIATION WEATHER RADAR DATA PROCESSING AND REAL-TIME IMPLEMENTATIONS

A DISSERTATION APPROVED FOR THE
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

BY

To my family...

Acknowledgments

Firstly, I would like to sincerely thank my research adviser Prof. Yan (Rockee) Zhang. Without his insightful guidance, valuable opinions and meticulous supports, my research career would be overwhelmed by difficulties faced along the way and my work would have been unachievable.

Secondly, I would like to extend my thanks to all my committee members, Prof. Tian-You Yu, Prof. Boon Leng Cheong, Prof. Ronald Barnes and Prof. Changwook Kim, for their great helps, suggestions and encouragements. Also, special thanks to Dr. Richard Doviak for his brilliant advices, enthusiastic inspirations and tireless reviews for the latest and most important research during my Ph.D career.

I would like to also express my deeply gratefulness to my outstanding colleagues for the invaluable supports and collaborations during my Ph.D career at OU.

Finally, I would like to thank my family: My wife Xi, my son to be born, my parents and my grandparents. They are the reason I get up everyday and work through difficulties and obstacles to pursue my professional career. Words can not express my gratitudes for their unconditional supports and sacrifices.

Table of Contents

List of Tables

List of Figures

Abstract

The objectives of this dissertation work are developing an enhanced intelligent radar signal and data processing framework for aviation hazard detection, classification and monitoring, and real-time implementation on massive parallel platforms. Variety of radar sensor platforms are used to prove the concept including airborne precipitation radar and different ground weather radars.

As a focused example of the proposed approach, this research applies evolutionary machine learning technology to turbulence level classification for civil aviation. An artificial neural network (ANN) machine learning approach based on radar observation is developed for classifying the cubed root of the Eddy Dissipation Rate (EDR), a widely-accepted measure of turbulence intensity [1], [2]. The approach is validated using typhoon weather data collected by Hong Kong Observatory's (HKO) Terminal Doppler Weather Radar (TDWR) located near Hong Kong International Airport (HKIA) and comparing HKO-TDWR EDR$^{1/3}$ detections and predictions with in situ EDR$^{1/3}$ measured by commercial aircrafts. The testing results verified that machine learning approach performs reasonably well for both detecting and predicting tasks.

As the preliminary step to explore the possibility of acceleration by integrating General Purpose Graphic Processing Unit (GPGPU), this research introduces a practical approach to implement real-time processing algorithms for general surveillance radar based on NVIDIA graphical processing units (GPUs). The pulse compression algorithms are implemented using compute unified device architecture (CUDA) libraries such as CUDA basic linear algebra subroutines and CUDA fast Fourier transform library, which are adopted from open source libraries and optimized for the NVIDIA GPUs. For more advanced, adaptive processing algorithms such as adaptive pulse compression, customized kernel optimization is investigated. A statistical optimization approach is developed for this purpose without needing much knowledge of the physical configurations of the kernels. It was found that the kernel optimization approach can significantly improve the performance. Benchmark performance is compared with the CPU performance in terms of processing accelerations. The proposed implementation framework can be used in various radar systems including ground-based phased array radar, airborne sense and avoid radar, and aerospace surveillance radar. After the investigation of the GPGPU on radar signal processing chain, the benchmark of applying machine learning approach on embedded GPU platform was performed. According to the performance, real-time requirement of the machine learning method of turbulence detection developed in this research could be met as well as Size, Weight and Power (SWaP) restrictions on embedded GPGPU platforms.

Chapter 1

Introduction

1.1 Missions

The scope of this dissertation is data processing of weather radars for measuring, detecting and classifying specific kinds of aviation weather hazards. The mission is to solve two problems in the weather radar systems: (1) Using parallel computing to accelerate Doppler weather radar signal processing chain processing with constraint of Space, Weight and Power (SWaP). (2) Using machine intelligence (MI) to build data modeling of the massive amount of weather radar data, and use such data modeling to detect (and classify) the hazards or clutters. The main focus of the targets is aviation-related hazards.

1.1.1 The Data Computing Challenge

As an example we developed an initial parallel computing framework that accelerate the reflectivity data product generation from NASA's high-altitude precipitation radars [3] as well as OU-ARRC's AIR radar processing. NASA-Goddard Space Flight Center, for example, has been operating multiple high altitude airborne radars. Figure 1.1 shows an example reflectivity mapping

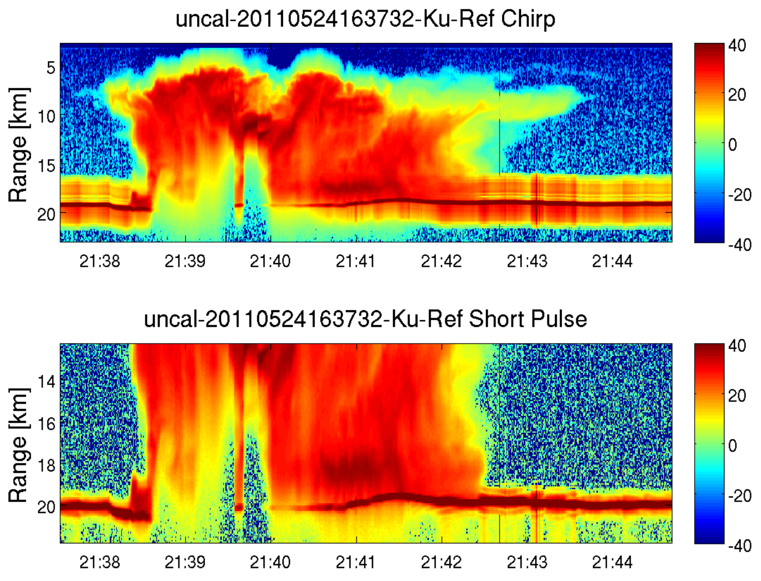


Figure 1.1: Sample image of hurricane reflectivity data from high-altitude radar flight campaign. Courtesy of NASA Goddard Space Flight Center, MD.

of a hurricane from 20 km altitude using one channel from one of the Ku-band (13 GHz) radars. For the radar records like in Figure 1.1, there are a few issues that need to be addressed. First, the radar used in Figure 1.1 was facing downward and was about 20 km above the sea. Second, the attenuation of radar singal at Ku band is substantial as it could be observed by the radar recordings around 21:39 and there should be a strong return around 20km range as it should be the sea level. Therefore, the attenuation of EM wave should be taken into consideration especially for shorter wavelength application, such as Ka-band based radar system, which will be introduced in this research. Currently, the computation of the data product is done offline and non-real-time. As a solid-state pulsed-Doppler radar, the computation procedure involves pulse compression, Doppler filtering (including pulse-pair processing), pulse averaging/smoothing, and reflectivity calculation. More details for this sensor and data are provided in [4]. The estimated raw data rate with current configuration is more than 10 Giga-Byte for each hour of flight data collection. The upgrade plans of these high-altitude precipitation radars involve multiple channels and multiple frequencies, which will dramatically increase (> 10 times) the amount and rate of data. Meanwhile, the need for being able to form the data product/image from onboard platforms is urgent, since it will greatly enhance the observation capability and forecasting capability of severe storms and reduce processing workload on the ground.

For this example, we suggested to develop an initial demonstration of real-time processing of these high-altitude radar data based on General Purpose Graphic Processing Unit (GPGPU). GPU processor has been used in previous airborne radar imaging product generation, such as the Synthetic Aperture Radar (SAR) [5]–[9]. However, it has not been used onboard for high-altitude precipitation radars before. Prior work in the area mainly focused on simple data parallelism scheme (i.e., dividing the radar scan image into GPU blocks). However, there are intrinsic parallelisms deep in each signal processing algorithm along the processing chain that have not been exploited. The optimal radar data redistribution between different stages of processing has not been studied (e.g., corner-turn), and an integrated, end-to-end multi-channel surveillance radar processing chain/function on a GPU platform has not been demonstrated for operating phased array radars. Furthermore, software tools to explore the parallel computing performance (such as CUDA/OpenCL) have been updating in unprecedented speed. New capabilities (such as nested parallelism and Basic Linear Algebra Subprograms (BLAS) libraries) in these tools provide foundations to break-through the current performance limitations and release the power of embedded parallel computing to radar processing, and was not been applied in prior R&D work.

On the other hand, the existing processing algorithms in these airborne radars are not optimized, or may not be carefully designed to be suitable for parallel processors. One of the challenging issues is the antenna and range sidelobes, which contaminates the radar measurement and distorts the measurement of ground truth [10]. In prior projects, our team has already developed a novel algorithm, called 2D-RMMSE (Two-dimensional Recursive Minimal Mean-Square Estimation) [11], which has successfully demonstrated the effectiveness of removing challenging antenna and waveform distortions with both ground radar data and simulated airborne radar data. Naturally, the next step is to apply this algorithm to actual measured airborne radar data from flight campaigns and deploy them to the onboard processors.

The computational load of these newly developed processing algorithms has been a hurdle for their application to real-time meteorological product generation. For example, the computation load for RMMSE algorithm is $O(N^3)$, where N is the length of waveform, compared to the magnitude of $O(N)$ for the traditional matched filtering [12]. Enhanced computing power (without significant increasing the size and weight of the processor system) is necessary to make the data product generation feasible for NASA, NSF or other agencies' missions.

1.1.2 The Intelligent Hazard Detection Challenge

The second challenge of the missions in this dissertation is the development of more advanced algorithms for detection, classification, diagnosis and monitoring different types of aviation hazards, especially for the weather-related hazards and electromagnetic (EM) interferences. For example, turbulence has been a major threat for aviation safety, and detecting turbulence reliably and in real-time using different sensors has been a significant challenge since 1960s [13], [14]. After the front-end signal processing and data quality control, the important step is to detect, classify and retrieve the information in the data. The challenge is balancing physical model complexity, computational load, accuracy, and algorithm performance with limited data quality and various types of hazards. This work uses aviation turbulence detection and wind farm clutter detections as examples, while the similar approach can be applied to other aviation hazards.

1.2 General Background of This Dissertation

In modern civil aviation, weather is one of the biggest causes of flight delays/cancellations and accidents, which is accounted for billions of dollars loss in the economy every year [15]. As for accidents, especially air crashes, the lives of people on board are on the string every single time. Those incidents are typical “Low-Probability, High-Impact” events, especially impact on the public faith on civil aviation industry which is the blood and veins in modern global economy. Preventing those incidents from happening would require, first, a method to detect such hazardous weather and, second, a practical approach to apply those method. Weather radar, as the primary measure to monitor weather, is the equipment to rely on for such task.

Airborne radar and ground radar have different purposes and architectures. Airborne weather radar is typically installed inside of the nose area of an aircraft. It is usually designed to serve the purpose of monitoring the weather of immediate front of the aircraft. Due to the size and power limitations, the range and resolution are limited. A ground weather radar, on the other hand, usually has higher power levels and better resolutions. Generally, the two most critical stages of a flight in aviation safety are taking-off and landing phases. Therefore, monitoring the weather condition near the runway would be most beneficial to the aviation safety. Terminal Doppler Weather Radar (TDWR) was developed and usually deployed near the airport to serve the purpose of weather surveillance near the airport terminal areas. TDWR provides rapid weather data updates, such as reflectivity, radial velocity and spectrum width, of the volume of atmosphere within the vicinity of the runway. Those data combined with coordinate information would be used to generate a detection and/or prediction of intensity of aviation related-weather hazards. .

1.2.1 Aviation Satety and Aviation Hazards

After human beings' first flight in 1903, following the footsteps of Wright brothers [16], aviation becomes one of the most important technological development in recent human history. From propeller biplanes, such as P-51 Mustang [17] to super-sonic fighter jets, such as F-22 Raptor [18], aerospace technologies advance with the time. In civil uses, aviation is one of the catalysts of the modern society. From four-seat Cessna 172 Skyhawk [19] to Jumbo-Jet Boeing

747 [20], civil aviation reshapes the modern life of the human beings in many ways. With modern passenger airplanes, it shrinks the spatial and temporal distance between any two points on the globe. The research and development on aviation, especially on engine and aerodynamic, also inspired the development of many other industries, such as spaceflight and automobile.

Table 1.1: Some of the Hazardous Weather Conditions for Aviation

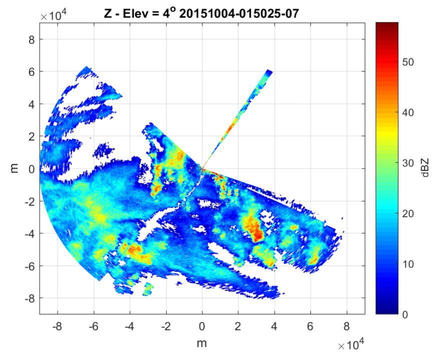
Type	**Notes**	**Incident**	**Detection Strategy**
Turbulence	turbulent movement of air masses	UA826 (1997) [21]	Radar (with precipitation)
Microburst	small down draft	DL191 (1985) [22]	Low Level Windshear Alert System(LLWAS) [23]
Windshear	a difference in wind speed and/or direction over a relatively short distance	USAir1016 (1994) [24]	Airborne Windshear Detection and Avoidance Program (AWDAP) [25]

Table 1.1 lists some of the most impactful hazardous weather conditions for aviation today. As shown in Table 1.1, several tragedies occurred because of those hazardous weather conditions and the lacking of understanding and detecting method of those weather conditions back then. Today, several systematic methods are developed to monitor those hazardous weather conditions, such as LLWAS for microburst and AWDAP for windshear. However, as for the turbulence, because of its chaotic nature (which could come from multiple sources) and its difficulty to be described [26], as it's named as "the chief outstanding difficulty" in hydrodynamics [27], it is difficult to find a systematic way to detect such weather conditions. Although there have been made some progresses in this area [26], none of those method works well enough to provide accurate prediction and detection under all conditions. As we will describe in latter part of this chapter, utilizing artificial intelligence technologies might

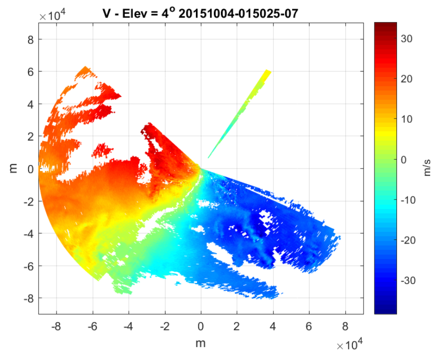
give us a better approach to detect and predict.

1.2.2 Principles of Weather Radar

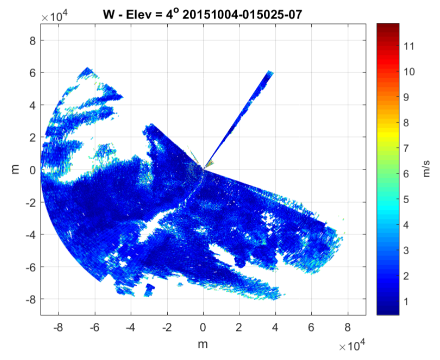
Initially Radar was the acronym of **Ra**dio **D**etection **a**nd **R**anging [28]. The concept of radar was first documented by Nikola Tesla in 1900 [29] after radio waves have been discovered by Heinrich Hertz in 1886 [30] which was predicted by James Maxwell from his famous equations in 1865 [31]. However, the development of radar was rapidly expedited during the war. During the World War II, radar was primarily used for detecting flight objects, such as aircrafts, in the air beyond human's eyesight [32]. This functionality is still used in today's military and civil aviation. During the war, radar operators found that not only the flying object but also weathers could produce echoes on their screen, which makes the radar operation for detecting flying objects more difficult in bad weathers. After the war, the phenomenon of weather-caused echoes was investigated and the first operational weather radar was developed by Davis Atlas from MIT [33]. Since then, radar was regularly used for weather surveillance purposes and weather radar itself was evolved from conventional parabolic dish radar [34] to dual-pol Doppler phase array radar [35], [36] to provide more information in terms of contents and speed. Nowadays, a completed S-band weather radar network WSR-88D (NEXRAD) was established in the US [37] and many TDWRs [38] are developed and deployed near many major airports to specifically serve the purpose of aviation safety [39]. In this research, the measurements from HKO-TDWR, a TDWR located near the Hong Kong International Airport (HKIA), will be used. It serves the same purpose for aviation safety as the US based TDWRs do.



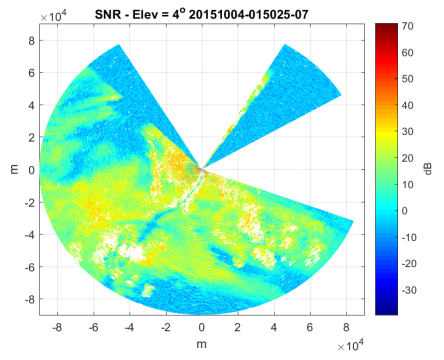
(a) Z



(b) V



(c) W



(d) SNR

Figure 1.2: examples of radar products of HKO-TDWR

Table 1.2: Basic Weather Radar Measurements

Radar Products	**Unit**	**Notes**
Reflectivity (Z)	dBZ	Represents the return power level of transmitted EM wave
Radial Velocity (V)	m/s	Indicates the mean velocity of targets within a radar pixel with respect to the TDWR, based on Doppler effect
Spectrum Width (W)	m/s	Represents the distribution of velocities within a single radar pixel
Signal-to-Noise Ratio (SNR)	dB	Represents the power ratio of the peak returned signal and noise floor

For the example of TDWR radar manufactured by Misushibi, Figure 1.2

shows sample data in Table 1.2 in a Plan Position Indicator (PPI) scan. As we can see, after data quality control processing, radar data products are not available in some areas due to low-SNR, interference or blockages. Table 1.2 lists some of the radar products which will be beneficial to turbulence detections in this research. The clear relation between turbulence and the spectrum width of radar measurements has been derived before [40]. However, as it will be discovered in this research, Eddy Dissipation Rate (EDR), a value to quantify the intensity of turbulence, is solely derived from radar spectrum width measurements and not sufficient to provide reliable detection or prediction. Therefore, finding a better way to exploit radar data is critical to improve the detection and prediction performance.

1.2.3 Artificial Intelligence

Artificial Intelligence (AI) has always been attractive to researchers in many fields ever since the invention of the computers. Although, there was a period of time, that the field of AI had slowed down, but after the discovery of Back Propagation (BP) algorithm [41], the neural network-based AI approach resurged as a promising tool to solve many real-world problems. Especially after the "Deep Learning" approach appeared with the booming of GPU supercomputing in recently years, the field of AI has been pushed into a much higher level, and evolving faster than ever before. More detailed history of AI will be given in Chapter 3.1.

1.2.4 High Performance Computing

The desire to a faster computing device has never been satisfied after the first electronic computer ENIAC [42]. Supercomputers are usually utilized on those tasks which need large amount of calculations, such as weather forecast, climate research, computational chemistry, astrophysics simulation and cryptanalysis. In the early days and most of the cases nowadays, High Performance Computer (HPC) refers to those huge clustered supercomputers, such as CRAY computers [43] and Sunway TaihuLight (The fastest supercomputer in the world) [44]. However, with the development of the heterogeneous computing [45], especially the parallel computing utilizing GPGPU [46], HPC has become more versatile, accessible and affordable for variety of studies. Today, Sunway TaihuLight comes with a whopping 93 PFLOPS of computational capability [44], but not all research needs or could gain access to such facilities. On the other hand, a single GPU card, Titan Xp (Pascal), released by NVIDIA comes with 12 TFLOPS, which is reasonably powerful and affordable ($1200 at release) [47] for many studies.

1.3 Selection of High Performance Computing (HPC) platform

To achieve real-time processing, there are two parts of acceleration need to be done, the acceleration of the AI/MI process itself and the front-end signal processing chain for improving the data quality. There are a few different hardware platforms that may be used for this real-time implementation scenario. Xining Yu's work [48], [49] provided a Digital Signal Processor (DSP) based embedded HPC platform. DSP is a specialized component that would provide HPC performance (160GFLOPS - TI C6678) in a relatively low power consumption. However, with the current stage of this AI/MI turbulence detection application, power consumption is not a critical constraint. In addition, the difficulty of implementing the algorithms in DSP, which will require heavy modifications, makes DSP a less ideal option for this research. In this study, a GPGPU based HPC approach is implemented. As mentioned in early part of this section, GPGPU approach would provide a sufficient high computational capability (12TFLOPS - Titan Xp Pascal) at a relatively affordable cost ($1200 - Titan Xp Pascal). In addition, with the utilization of NVIDIA's CUDA, most of the existing algorithms would not need extensive modifications and still achieve desirable acceleration performance.

Chapter 2

Systems and Requirements

2.1 Aviation Weather Hazards and Radar Sensing

As mentioned in Chapter 1, there are numerous weather hazards which could impact aviation safety, such as icing, microburst, windshear and turbulence. Some of these hazards could be detected by radar sensing method and avoided if an appropriate precaution maneuverer is taken by the pilot. Turbulence is the main hazard that is focused in this work.

As modern aviation advances as of today, turbulence would rarely do the damage on the structure of modern jet-planes. However, turbulence remains a major threat to the crew, passenger and cargo on-board. Moreover, severe turbulence could cause the airplane changing its direction dramatically and losing altitude quickly, which is extremely dangerous especially when the airplane is close to the terrain during takeoff and landing phase of a flight. In addition to the threat to aviation safety caused by turbulence, passenger experience would be certainly worse when turbulence is experienced. Therefore, having a reliable method to detecting and predicting turbulence on and off the flying track would not only improve the aviation safety but also the passenger experience.

Table 2.1: Comparison Between Lidar and Radar

	Lidar	**Radar**
Wavelength	μm range	cm range
Range	30km and more	150km and more
Attenuation	high	low
Environment	Clear Air	With Precipitation
Beam Width	Narrow	Wide

As for turbulence, there are several different kinds of turbulence, such as clear air turbulence and turbulence associated with precipitations. For clear air turbulence, due to limitation of radar, it would be better to utilize lidar, an alternative of radar utilizing light (high frequency EM wave), instead of radar for observations. But for the turbulence with precipitation, it is plausible to use existing radar technology and measurement for detection and prediction. Table 2.1 lists comparisons between lidar and radar.

2.2 Examples of Radar Systems Used in Research

Data from the following radars are used as part of this research. They are HIWRAP, TDWR, a Ka-band radar prototype and AIR. In this section, they will be introduced respectively.

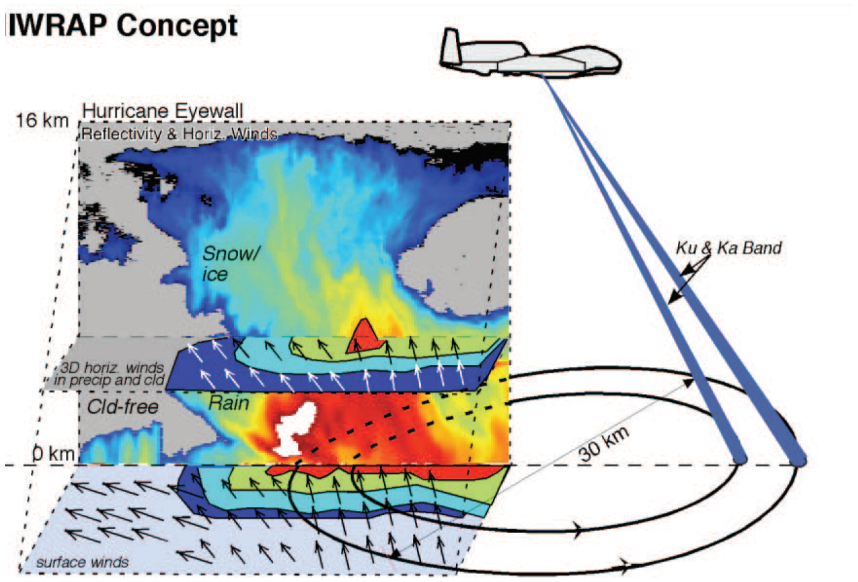


Figure 2.1: HIWRAP Measurement Concept. Courtesy of NASA Goddard Space Flight Center, MD.

2.2.1 High Altitude Imaging Wind and Rain Airborne Profiler (HIWRAP)

HIWRAP is a dual-frequency (Ka- and Ku-band), dual-beam (300 and 400 degs incidence angles), conical scan, solid-state transmitter-based Doppler radar system [4]. It is designed to operate on the Global Hawk UAV on a high-altitude assignment. Figure 2.1 depicts the concept of HIWRAP [4]. HIWRAP utilizes the solid state transmitter and advance pulse compression schemes, which allows HIWRAP to use significantly less power than typical radar used for cloud and precipitation measurements. In addition to Global Hawk, HIWRAP has been deployed on NASA ER-2. However, when mounted on NASA ER-2, the HIWRAP was in the fixed nadir pointing configuration.

The specifications of HIWRAP is described in Table 2.2 [4]. Because the utilization of advanced pulse compression scheme, it requires more computa-

Table 2.2: Specifications of HIWRAP

Parameters	**Specification**	
	Ku-band	**Ka-band**
RF Frequency (GHz)	Inner Beam: 13.910 Outer Beam: 13.470	Inner Beam: 35.560 Outer Beam: 33.720
Transmitter Peak Power (W)	25	8
3 dB Beam Width (°)	2.9	1.2
Polarization	Vertical (outer beam), Horizontal (inner beam)	
Minimum Detect Reflectivity (dBZe, 60 m resolution, 10 km range and 3 km chirp pulse)	0.0	-5.0
Dynamic Range (dB)	> 65	
Doppler Velocity (m/s)	0-150 (Accuracy < 1.5 m/s for SNR > 10 dB)	
Scanning	Conical Scan, 10-30 rpm	

tion power to process the data than regular pulsed radarsr to achieve similar if not better performance.

2.2.2 Hong Kong Observatory - Terminal Doppler Weather Radar (HKO-TDWR)

TDWR is a radar designed to serve the purpose of weather surveillance within the vicinity of an airport. Therefore, TDWR operates on C-band and has a shorter range and higher resolution than NEXRAD which operates on X-band. The difference in reflectivity output between NEXRAD and TDWR is demonstrated in Figure 2.2. However, due to the resolution and range difference between NEXRAD and TDWR, it is difficult to perform side-by-side comparisons. They complement each other with overlapping coverage, designed to optimally cover different airspace regions.

Table 2.3: Specifications of HKO-TDWR

Parameters	**Values**
Frequency	5.625 GHz
Antenna systems	7.9 m Metal Parabolic Reflector. Linear polarized. Beamwidth $\theta_1 \leq 0.55°$, gain 50.54 dBi. Sidelobes are $\leq$ 27 dB within 5° and $\leq$ 40 dB beyond 5°.
Scanning	Hazardous weather and monitor modes with different elevation coverages; 5 mins per volume scan, azimuth scan speed 24° per second.
Transmitter	250 KW peak power Klystron, 500 watts average, transmitted signal phase is uncorrelated from pulse to pulse.
Waveform	Pulse width 1 μs, Dual PRF 1.104 KHz and 1.38 KHz. $\pm$60 m/s non-ambiguous velocities, occupies 4 MHz bandwidth.

Receiver and processor	Receiver noise power: - 111 dBm at 1 MHz bandwidth;
	Dynamic range is 100 dB;
	3 dB noise figure;
	ground clutter suppression > 50 dB
	and velocity accuracy $\pm$ 0.5 m/s.

Display properties	0 to 90 km, 150m range bin spacing, 600 range bins per radial, 0.70° azimuth spacing.

Specifications for a typical TDWR (in particular, the Mitsubishi TDWR located near Hong Kong International Airport) are listed in Table 2.3 [38]. As we can see in Table 2.3, the TDWR operates with a narrower beamwidth and

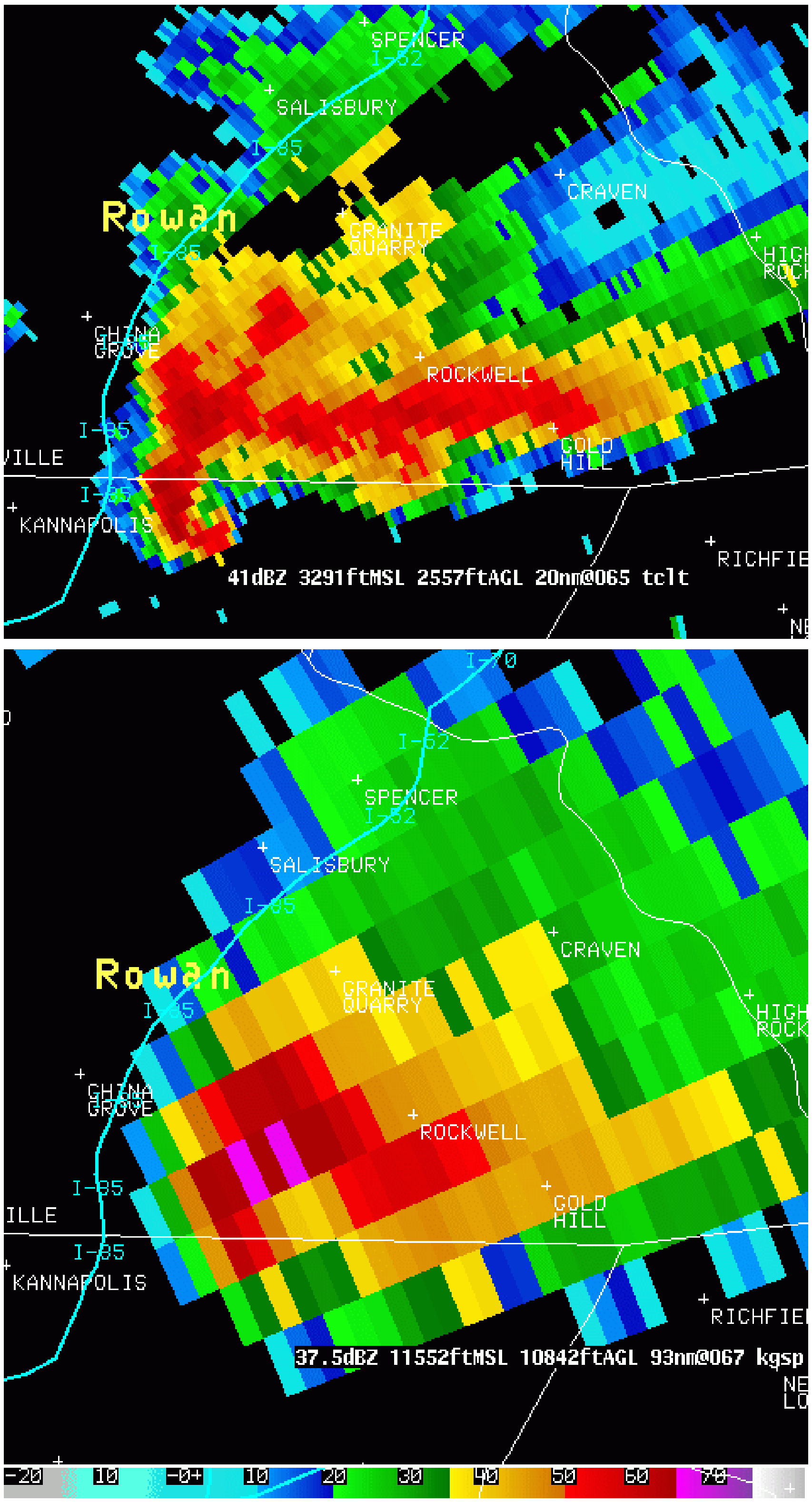


Figure 2.2: An example of the difference in reflectivity output between NEXRAD (bottom) and TDWR (top). Courtesy of NOAA [50].

much higher rotation speed at lower power level comparing with NEXRAD. Because the power level of TDWR is high enough compared with solid-state-based radar system like HIWRAP, pulse compression scheme is not necessary and is not implemented in TDWR.

2.2.3 Ka Band Frequency-Modulated Continuous Wave (FMCW) Radar Prototype for Airborne Remote Sensing

A small, low-cost FMCW radar at Ka band was built at Radar Innovations Laboratory of the University of Oklahoma to verify the hardware performance of FMCW radar and the potential of using such radar for icing hazard detection. A block diagram of this FMCW radar is demonstrated in Figure 2.3 and a photo of this prototype built is shown in Figure 2.4.

As seen in Figure 2.3, the Ka band FMCW prototype radar is based on a 2.4 GHz FMCW radar, and the Intermediate Frequency (IF) signal is generated by a Digital Phase Lock Loop (DPLL). After 1 to 2 stages of multiplier depending on the channel, the stage 2 IF wave will be mixed with 2.4 GHz signal to generate 35.2 GHz Ka band transmitted wave in the transmitting channel and with 35.2 GHz Ka band received wave to generate 2.4 GHz signal in the receiving channel. Therefore, the Ka band signal works as the carrier and the actual FMCW signal process and radar function is operated at the 2.4 GHz baseband. There are two antennas used in the system, one used for transmitting and the other one used for receiving. Both the antennas are to be aligned straight and in the same position so as to reduce the errors.

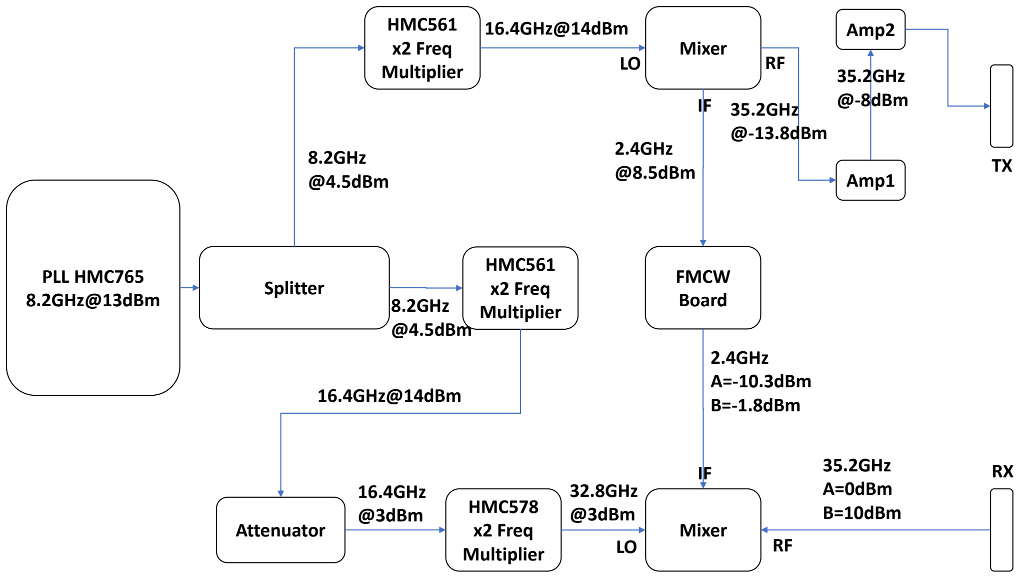


Figure 2.3: Block diagram of Ka band FMCW radar

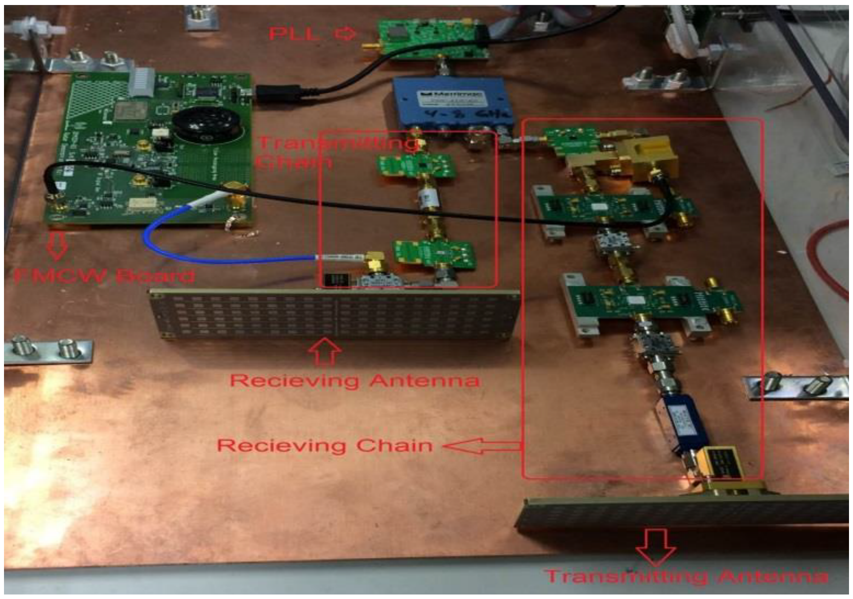


Figure 2.4: Prototype of Ka band FMCW radar

Detailed link budget analysis regarding the different types of targets is provided at the initial stage. It could be used as a guideline for system design and the baseline of performance expectations. As seen in Figure 2.5, it was shown that if the airborne radar antenna has a beamwidth of 4.6/15 degrees (Horizontal/Vertical) and a gain at 30 dB, and the FMCW transceiver operates at 35 GHz with 50 MHz ramping bandwidth, 20 MHz LPF cutoff frequency and 1 ms ramping time, 50 m range resolution is achievable at 1km range, with a minimum detectable reflectivity at 0 dBZ and 10 Watt transmitted power. It is sufficient to detect hazards large particles in cloud (i.e., ice) and hydrometeors which are larger than 0.1 mm.

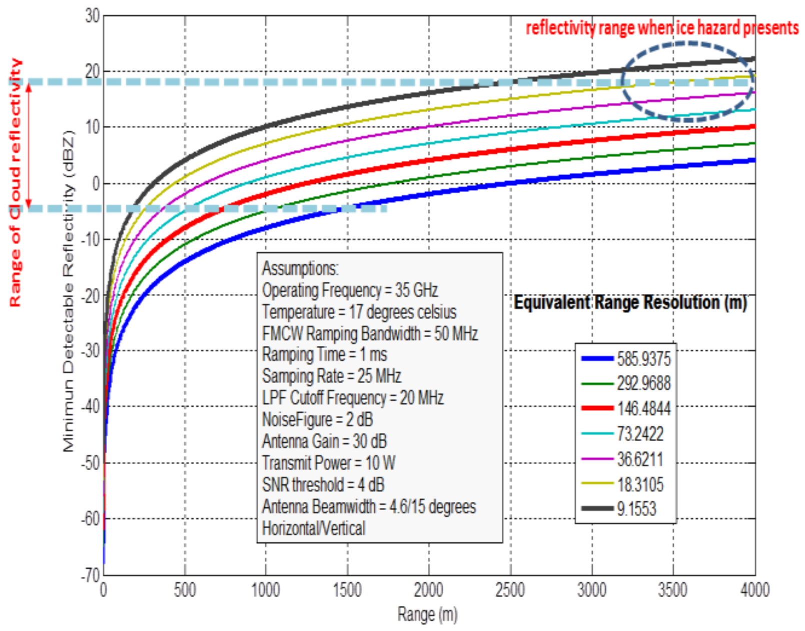


Figure 2.5: Link budget analysis of Ka band FMCW radar

The initial field test of this prototype radar was done indoor with a rectangular aluminum board and aluminum sphere targets in a relatively short range (since transmit power is only 0 dBm). The size of rectangular board is 20 inches by 15 inches and the diameter of the sphere was 39 inches. The

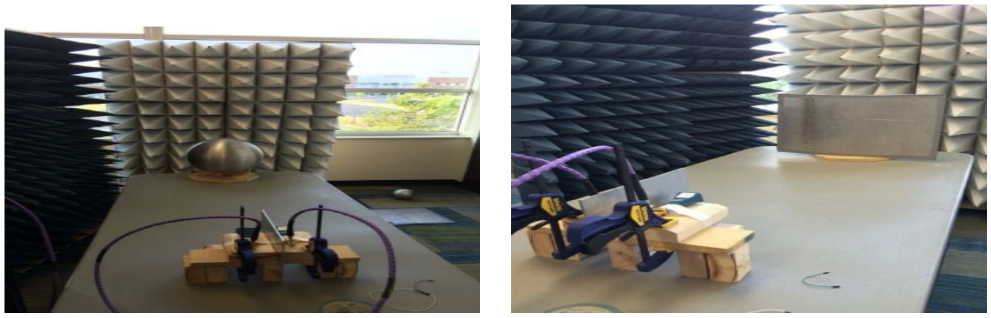

(a) Sphere target (b) Rectangular target

Figure 2.6: In-door test environment

setup of this initial indoor test is shown in Figure 2.6 and the background reflection in Figure 2.7. It should be noted that there are two antennas used in the system, one used for transmitting and the other one used for receiving. Both antennas are to be aligned straight and in the same position to reduce the errors.

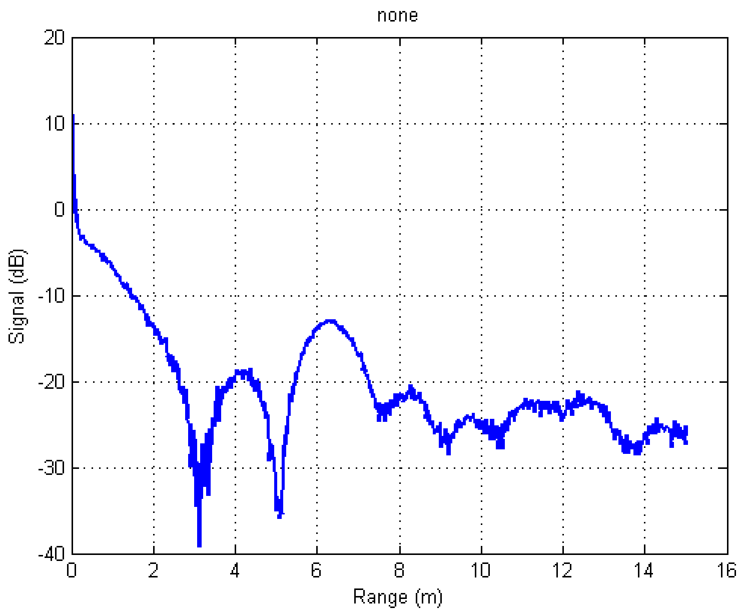


Figure 2.7: Example range profile

2.2.4 Atmospheric Imaging Radar (AIR)

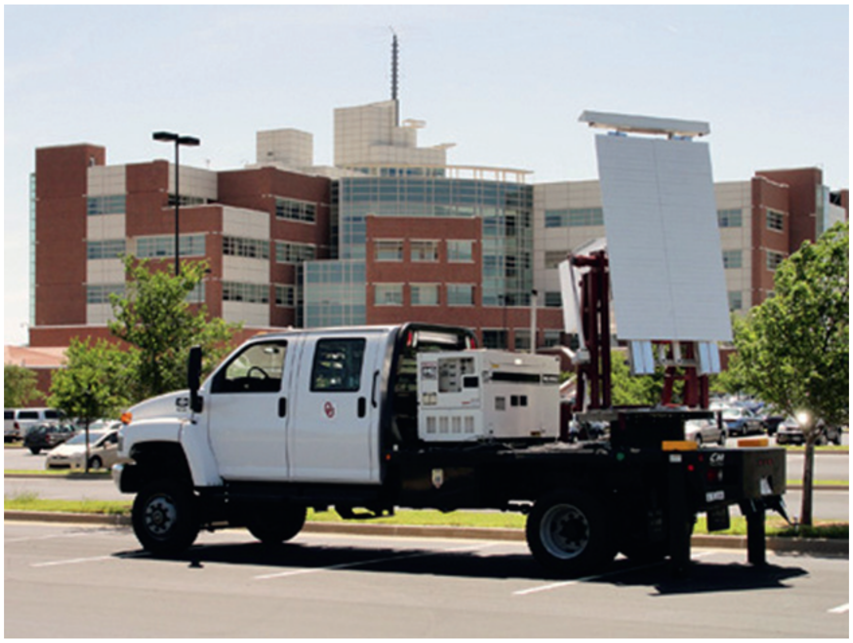


Figure 2.8: Photo of AIR. Courtesy of OU-ARRC.

AIR is one of the systems developed by OU-ARRC [51]. AIR is an X-band imaging radar with beamforming capability in elevation by using a linear arrays with 36 elements and operating mainly in Range-Height-Indicator (RHI) mode. However, with the antenna installed on a mechanical rotation pedestal, AIR also supports in Plan Position Indicator (PPI) mode. As seen in Figure 2.8, the antenna is a planar array with 37 separate units and the transmit element is mounted at the top of 36 different receiving elements. And the whole system is mounted on a truck bed which allows the system to be transported to close to the place of interest, which increases the resolution and reduces the EM attenuation and power requirement. In addition, the capability of using pulse compression techniques improves its resolution and reduces the power requirement.

For the back-end components, each receiving antenna element is connected to a down conversion unit with the capability to down-convert RF signal at 9.55 GHz to IF signal at 50 MHz. Then the IF signal is passed to 1 of 5 eight-channel digital receivers at a sampling rate of 40 MHz. Such sampled signal is then used to produce 14-bit IQ data streams to be stored on hard disk for future analysis [51]. For the transmitter and up-converter part of the back-end system, AIR utilized a 3.5kW traveling wave tube (TWT) amplifier to generate output transmitting signal. An arbitrary waveform generator is utilized to provide waveform flexibility at 50 MHz. Then such a signal is up-converted to RF frequency at 9.55 GHz. The RF signal is amplified by the TWT amplifier and then radiates through transmit antenna, which as a fan-beam shape and 1 deg azimuth, 20 deg elevation beamwidth.

The specifications of AIR are listed in Table 2.4 [51]. As indicated in the specifications, AIR would benefit from incorporating pulse compression algorithm, which will increase the sensitivity and range resolution. As will be discussed later, adaptive beamforming and pulse compression processing are being considered for AIR. These algorithms are computation intense but highly capable to be parallelized. Therefore, applying an acceleration scheme based on GPGPU platform developed in this research would be highly beneficial.

2.3 Real-Time Requirement for Observing Weather Hazards

Intense and transient weather hazards, such as microbursts, turbulence and sometimes tornado, require real-time detection and prediction. It would be

Table 2.4: Specifications of AIR

Parameters	**Values**
Peak Power	3.5 kW
Frequency	9550 MHz
Pulse Length	1-15μs
Pulse Repetition Frequency	1-2000 Hz
Maximum Duty cycle	2%
Maximum Pulse Bandwidth	5 MHz
Fan Beamwidth	Vertical : 20° Horizontal : 1°
Number of Beams	Infinite number of 1° vertical beams
Beamforming Techniques	Fourier, Capon, Robust Capon, etc.
Array Aperture	1.2×1.8 m
Sampling Rate	40 MHz
Rotation Rate	up to 20°s^{-1}
Rotation Angle	-80.0° to +100.0° (180.0° by 20.0° coverage)
Sensitivity	Better than 10 dBZ at 10km Gain with Pulse compression: Approximately 10 dB
Range Resolution	Short Pulse : 150 m with Pulse Compression : 30m
Polarization	Horizontal

critical to know the location and intensity of those weather hazards for air-controller and pilots to take pre-cautious measures to avoid damages. Therefore, a proper acceleration scheme must be taken into consideration. Generally, if the radar data are gained as batches, data processing should be done between two consecutive batches. As for NEXRAD and TDWR, a typical Volume Coverage Pattern (VCP) takes about 5 minutes. Thus the real-time requirement should be easily met. However, the area of interest and resolution varies depending on applications, which would lead to a large number of points of interest needing to be processed. The adjacent points of interest may or may not be treated as independent in different algorithms.

2.3.1 Solutions to Acclerated Computing Using General Purpose Graphic Processing Unit (GPGPU)

There are two ways to accelerate an algorithm. First is to reduce the computational complexity category of the algorithm by optimizing the computation scheme of the algorithm itself. Second is to adopt a parallel computation scheme for the algorithm to be accelerated, and then to implement the modified paralleled algorithm on a parallel computing platform, such as Multi-Core CPU and GPGPU. In this research, a GPGPU approach is utilized. As for machine learning based algorithms, a specified machine learning ASIC would further expedite the training and implementing process, such as Tensor Pressing Unit (TPU). Table 2.5 demonstrates a comparison among specific CPU, GPU and TPU.

In this research, GPGPU was selected as the platform for algorithm imple-

Table 2.5: Comparison CPU, GPU and TPU

	CPU i9-7980X	**GPU TITAN Xp**	**TPU**
Cores	18	3840	4
Speed (TFLOPS)	3.08	12.15	180
Bandwidth (GBps)	85.332	547.7	600
TDP (Watt)	165	250	160
Suitable Tasks	General	More Strict	Application Specific

mentation, because of its moderate easiness to be programmed and relatively good performance compared with CPU counterpart. Also, with the support of cuDNN, GPU could be used as acceleration device for various deep learning frameworks, including TensorFlow, Caffe, Torch, etc., with little or no major modification on existing deep learning algorithms. This will further expand the usage of GPU devices in the field of this research focusing on.

Chapter 3

Machine Intelligence Algorithms

3.1 Introduction to Machine Intelligence

Machine Intelligence (MI), also known as Artificial Intelligence (AI), is the intelligence expressed by the machine built by human beings. Generally, machine intelligence is the intelligence expressed through an ordinary computer. It's also a research field related to the possibility of such intelligence system being reality and how to implement such a system if it is possible. In text books, such research field is defined as "research and design on intelligent agent", and an intelligent agent is a system that can observe the surroundings and act appropriately to achieve a pre-set goal. Machine Intelligence is also defined by John McCarthy [52] as the science and engineering of making such intelligence machine. The research on Machine Intelligence could be divided into several technical problems and is mainly focusing on solving practical problems.

There are two types of definition of machine intelligence, known as "strong AI" and "weak AI", see Table 3.1. "Strong AI" is defined as a machine with the capability of reasoning and solving problems and with self-consciousness. On the other hand, "weak AI" is defined as a machine that only looks "intelligent"

Table 3.1: Types of AI

AI Types	Notes
Strong AI	a machine with capability of reasoning and solving problem and self-conscious
Weak AI	a machine being only look "intelligent" but without self-conscious

but without self-conscious. At current stage, the research and applications on AI are mostly applied on the definition of "weak AI". However, with the emergence of artificial neural network, the AI has become stronger and more capable solving real world problems which are improbable if not impossible to solve before, thank to the growth of computation power that supports such AI applications.

The main focus of this research is on Machine Learning (ML), which is an important branch of AI. Focus of the research on AI begins from reasoning via knowledge to learning, and obviously, machine learning is one viable way to achieve AI by solving the problem using learning approach of machines. During the past 30 years, ML has been developed into an interdisciplinary field including but not limited to probability theory, statistics, approximation theory and computational complexity. The theory of ML is to design and analyze a kind of algorithm which would make computer "learn" automatically. The algorithm of ML is a kind of algorithm that would automatically analyze data and extract principles of such dataset to make predictions on unknown dataset based on the concluded principles from the "learning stage". Because of the involvement of large amount of statistical theories in developing ML algorithms, ML theory is also called statistically learning theory. Nowadays, ML is applied on numerous fields including but not limited to data mining, computer vision, natural language processing, biometrics and search engine.

Table 3.2: Types of Machine Learning Algorithms

Types	**Notes**	**Examples**
Supervised Learning	Training with pre-labeled dataset	Regression
Unsupervised Learning	Training with unlabeled dataset	Clustering
Semi-Supervised Learning	Hybrid of Supervised and Unsupervised Learning	PU learning
Reinforcement Learning	Maximize reward by evaluation	Q-learning

Generally, there are 4 types (as listed in Table 3.2) of machine learning: supervised learning, unsupervised learning, semi-supervised learning, reinforcement learning. Supervised learning approach is utilized in this research to develop the AI for turbulence detection/prediction. Also, there are several different popular ML algorithms with the Artificial Neural Network (ANN) being the most popular one, which is also the approach utilized in this research. Other than traditional Back-Propagation ANN, several variants of ANN also exist, such as Convolution Neural Network (CNN) which is the fundamental model for deep learning. Decision Tree (DT), Support Vector Machine (SVM) and Bayesian Classifier are also ML algorithms, compared with ANN, those algorithms may not perform as well on a large variety of applications as ANN, but those algorithms have the advantage of being easy to understand as a "white box" process compared with "black box" process of ANN.

3.2 Artificial Neural Network (ANN)

In the field of ML and Cognitive Science, ANN is a mathematical and computational model which mimics the structure and functionality of biological neural network. The capability of ANN comes from a large amount of interconnection between neurons. In most cases, ANN could alter its internal structure based on the change of external information. Therefore, it is an adaptive system.

Table 3.3: Components of ANN

Components	**Notes**	**Examples**
Architecture	Architecture defines the variables and the topological relation among them in the network.	Weights of the interconnections between neurons and activities of the neurons.
Activity Rule	Most ANN models have motivation rules of short time period. They defines the way neurons changing their values based on the changes on other neurons.	Activity Rule depends on the weights in the network.
Learning Rule	It's the motivation rule of long time period of ANN. It defines the way of weights changing in the network along with time progressing.	Learning Rule depends on the activities of neuron, target value and current weights.

A typical ANN consists of 3 parts: Architecture, Activity Rule and Learning Rule, which are listed in Table 3.3.

Generally, an ANN consists of multi-layers of neurons. Every layer of neurons has inputs (outputs of neurons from the former layer) and outputs. Layer

i has N_i neurons and every N_i neuron takes the outputs of respective N_{n-1} neurons as inputs. The interconnection between neurons is called Synapse in biology. In mathematical model, every synapse has a value called weight. To calculate the "potential" of a neuron in i-th layer, multiply the weight and output from respective neurons from $i-1$-th layer and calculate the summation of those results. Then the "potential" value passes through an activation function, which is used to control the range of value, to get output of the neuron. Typical activation functions include Sigmoid Function, which is continuous and differentiable that makes Delta Rule processing easier. The output of the neuron is non-linear, but it's acceptable as long as the final output is meaningful.

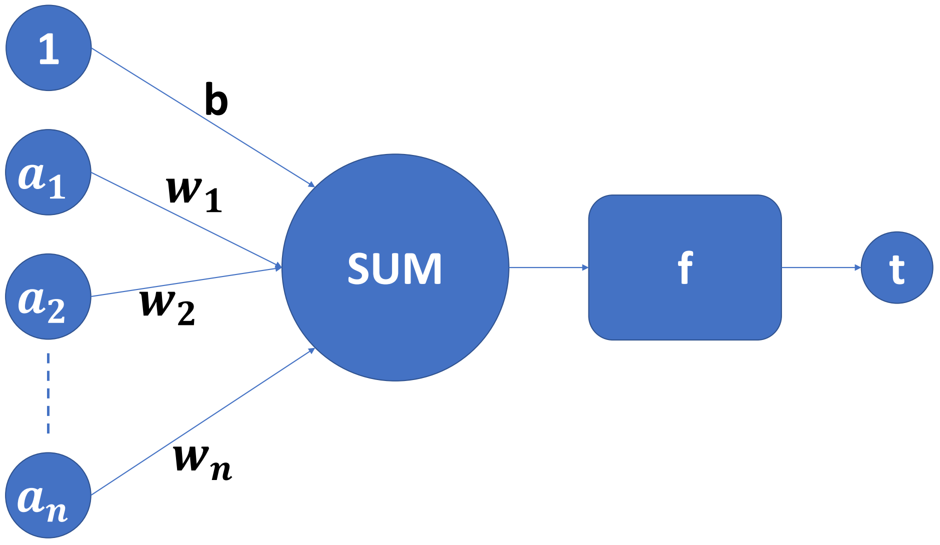


Figure 3.1: Typical structure of a Neuron. a_1 a_n are inputs, b is the offset, w_1 w_n are weights of respective Synapse, f is the activation function and t is the output.

$$t = f(\overrightarrow{W}'\overrightarrow{A} + b) \tag{3.1a}$$

$$\overrightarrow{W}'\overrightarrow{p} + b = 0 \tag{3.1b}$$

Figure 3.1 depicts the structure of a neuron and the function of this neuron is described by Equation 3.1a, where $\overrightarrow{W}$ is the weight vector and $\overrightarrow{W}'$ is the transpose of $\overrightarrow{W}$, $\overrightarrow{A}$ is input vector, b is the offset and f is the activation function. From Equation 3.1a and Figure 3.1, it could be concluded that the function of a neuron is to first calculate the inner product of input vector and weight vector, then get a scaler result through the nonlinear activation function. In formal words, a neuron divides an n-dimensional vector space into two parts by a hyperplane, called decision boundary, and for a given input vector, neuron could determine which side of the hyperplane the vector is in. The equation of this hyperplane is described in Equation 3.1b, where $\overrightarrow{p}$ is the vector on the hyperplane. Figure 3.2 demonstrates a typical structure of a back-propagation artificial neural network (BP-ANN) with 3 layers.

3.3 An Example Application: Wind Farm Clutter Detection

3.3.1 Introduction

In recent years, there has been a proliferation of wind farms across the country. The wind turbines in wind farms appear as a clutter not only in airborne

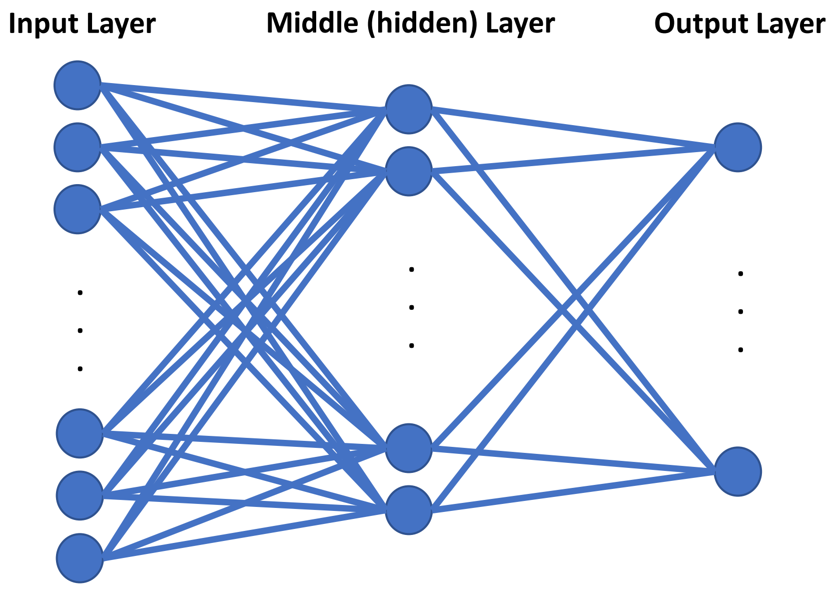


Figure 3.2: Typical structure of a back-propagation artificial neural network (BP-ANN)

radars but also in ground-based radars that scan at lower elevation. The wind turbine clutter obscures the visibility of ground and airborne targets, especially for an airborne platform. For some radars, wind turbines' sidelobes (flashes of energy in neighboring cells) contaminate the data while for others, the turbine itself is present at the range cell of interest. The metallic tower and ceramic blades of wind turbines generate a significant radar return. Even the sidelobes' power is comparable to ground targets. The rotational motion of the blades contaminate the frequency domain of the return. Any target in the vicinity of a wind turbine will be obscured in terms of power and Doppler frequency shift. In this paper we attempt to recognize the range cells that have wind turbine clutter. After recognition, a demodulating signal can be added to the original signal to mitigate the effect of wind turbine returns but that will be outside the scope of this paper. We will focus on the first part, to correctly determine the wind turbine presence in a given range-azimuth cell.

3.3.2 Previous Works

Wind Turbine Clutter (WTC) is a relatively new problem in the field of remote sensing. This problem especially affects an airborne platform. The attempt to using machine learning algorithms to solve this problem is a novel idea. There are papers on other methods of wind turbine detection, e.g., using a transponder on the wind turbine itself [53]. Another work [54] used signal processing techniques and proper radar parameters setup to detect wind turbines. In both cases, the wind turbines are viewed as targets, not as clutter. By treating wind turbines as clutters, this paper assumes that the return signals are sporadic, partial, and uncorrelated.

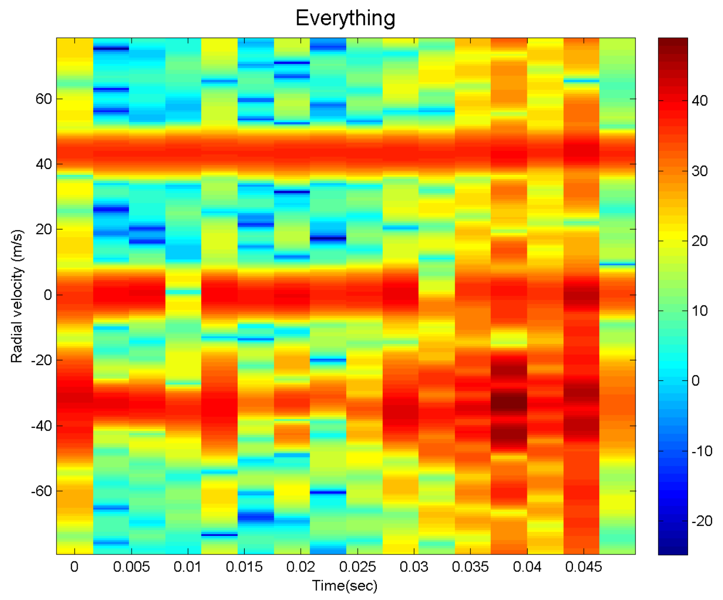


Figure 3.3: Time-Doppler spectrum of a range cell which contains ground clutter, wind turbine and a target

3.3.3 Supervised Learning Approach

All of the modern radars transmit multiple pulses, and with the return from each pulse, a time-Doppler signature of a range-azimuth patch can be created. The radial speed of a target is modeled as constant because the looking angle[1] to a target remains constant for each CPI[2]. This produces a straight line in the time-doppler plot[3]. Doppler signatures from a wind turbine is more complicated. The metallic tower produces a return with doppler-shift equaling to that of the radar platform velocity. The blades are constantly rotating and thus produce a periodic pattern that has high and low Doppler frequency shift which is seen as “flashes” in the time-Doppler plot. The radial speed of the blades fluctuates periodically because the looking angle to each blade changes during the rotation. Wind turbines are operated to face the incoming wind, so the direction of a wind turbine may be unpredictable. The Doppler spectrum changes with a periodicity. This periodicity produces a pattern, which can be recognized using MI algorithm such as ANN.

The sample data is generated from an airborne radar simulation software [55] (with simulated parameter listed in Table 3.4) and is organized into a matrix of size $600\times128\times60$, where there are 600 different range cells, 60 different azimuth cells, and 128 pulses in each azimuth (here 1 CPI has 128 pulses). For each range-azimuth cell (there are $600\times60 = 36000$ cells), the 128 pulses are taken and used to generate its time-Doppler spectra. This time-Doppler spectra is the input to the ANN. Supervised learning method is used. The

[1]azimuth and elevation

[2]Coherent Pulse Interval, the time for which a train of coherent pulses are transmitted, usually tens of milliseconds

[3]The doppler-shift value is created by the combined motion of radar platform and the target

Table 3.4: Simulation Parameters for Airborne Radar

Parameters	**Notes**
transmit power	40W
frequency	9.486 GHz
PRF	10 kHz
bandwidth	4.6875 MHz
waveform	phase coded
sampling frequency	12 MHz
antenna	64×64 planar AESA array antenna in Y-Z plane, about 2° beamwidth in AZ/EL, -60° - 60° (0° is X-axis) 2° step azimuth coverage
number of pulses	64 in each azimuth

training examples are created from the radar return itself but with the range-azimuth cell containing ground truth. This can be done by using other sources of information and is needs to be done only once to create a set of training data. Once the network is trained, the network weights can be applied to the new radar returns for classifications.

Structure of ANN

The neural network structure is 1024-80-3 (the number of nodes in each layer) as shown in Figure 3.6, which uses Sigmoid function as the activation function. The use of 80 hidden nodes is based on empirical experiment results. Other number of hidden nodes were also used but having 80 hidden nodes, leads to optimal balance between performance and speed. The use of 1024 input nodes is dictated by the data itself. The data-size is kept constant because reducing input data will also reduce the number of features. The output data is a 3 element vector with each element value ranging between 0 and 1. The element position where the value is closest to 1 serves as the identification of the entity.

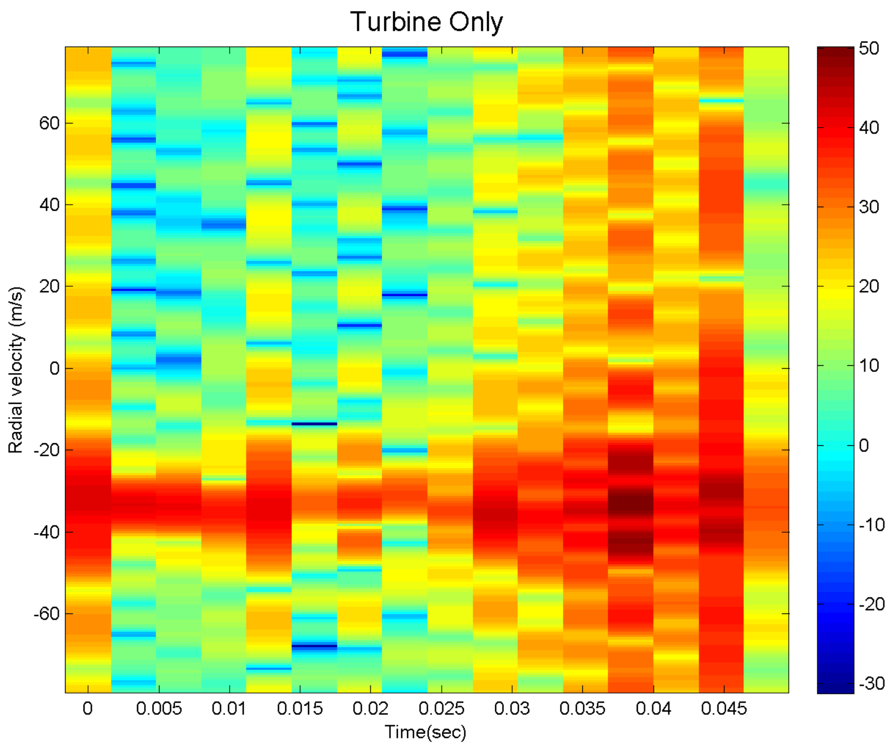


Figure 3.4: Time-Doppler spectrum of a range cell which contains a wind turbine.

Numbers 1, 2, and 3 are assigned to ”Noise”, ”Ground Clutter”, and ”Wind Turbine” respectively[4].

Learning and Cross Validation

After extracting the returns from the correct range-azimuth cell and identifying them to be Noise, Ground Clutter, or Wind Turbine, those data was fed into the ANN and the training process iterated 5000 times. A scheme of cross validation was applied where, for each iteration, 30 validation and 40 testing examples were extracted randomly from the original training data set.

[4]For example, output of [0.012, 0.986, 0.005] meant the pattern is recognized to be entity 2 which is ”Ground Clutter”

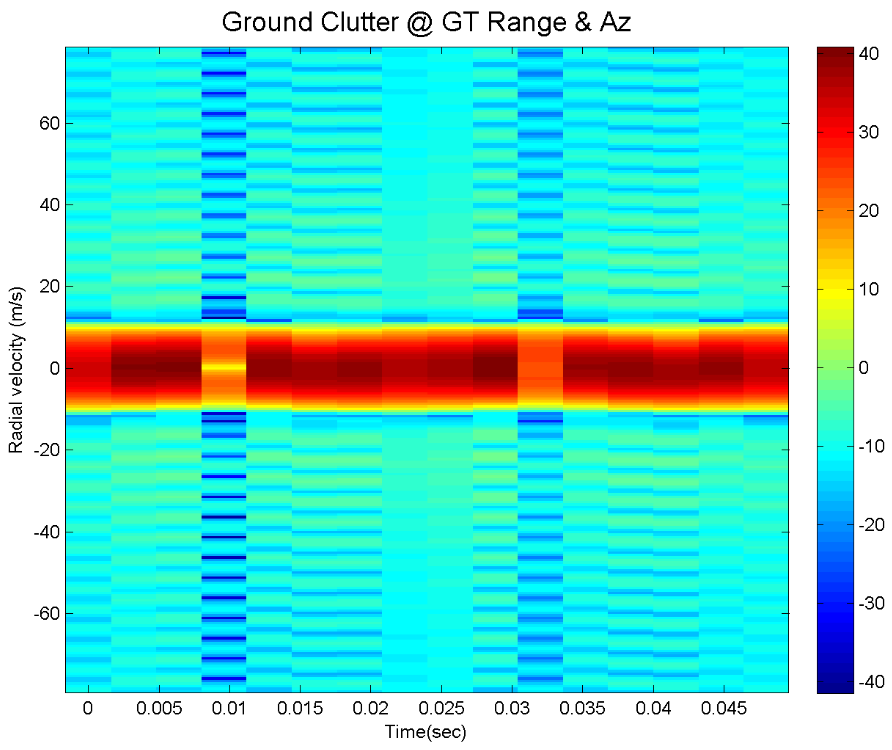


Figure 3.5: Time-Doppler spectrum of a range cell which contains ground clutter, target spectrum will also look the same but different radial velocity

Training was done in the remainder of the original training data set. The validation and testing examples could, in theory, have some overlapping samples because of the random extraction. This particular scheme was applied because it allows for arbitrary number of iterations to be performed. The cross-validation results are in Figure 3.7 and it can be seen that the ANN converges after 2500 iterations. Here, the Mean Square Error (MSE) is calculated as $\sum_i^I \sum_j^J (output_{i,j} - truth_{i,j})^2/IJ$, where I is the total number of pixels in a PPI scan and J is the total number of nodes in the output layer of the neural network.

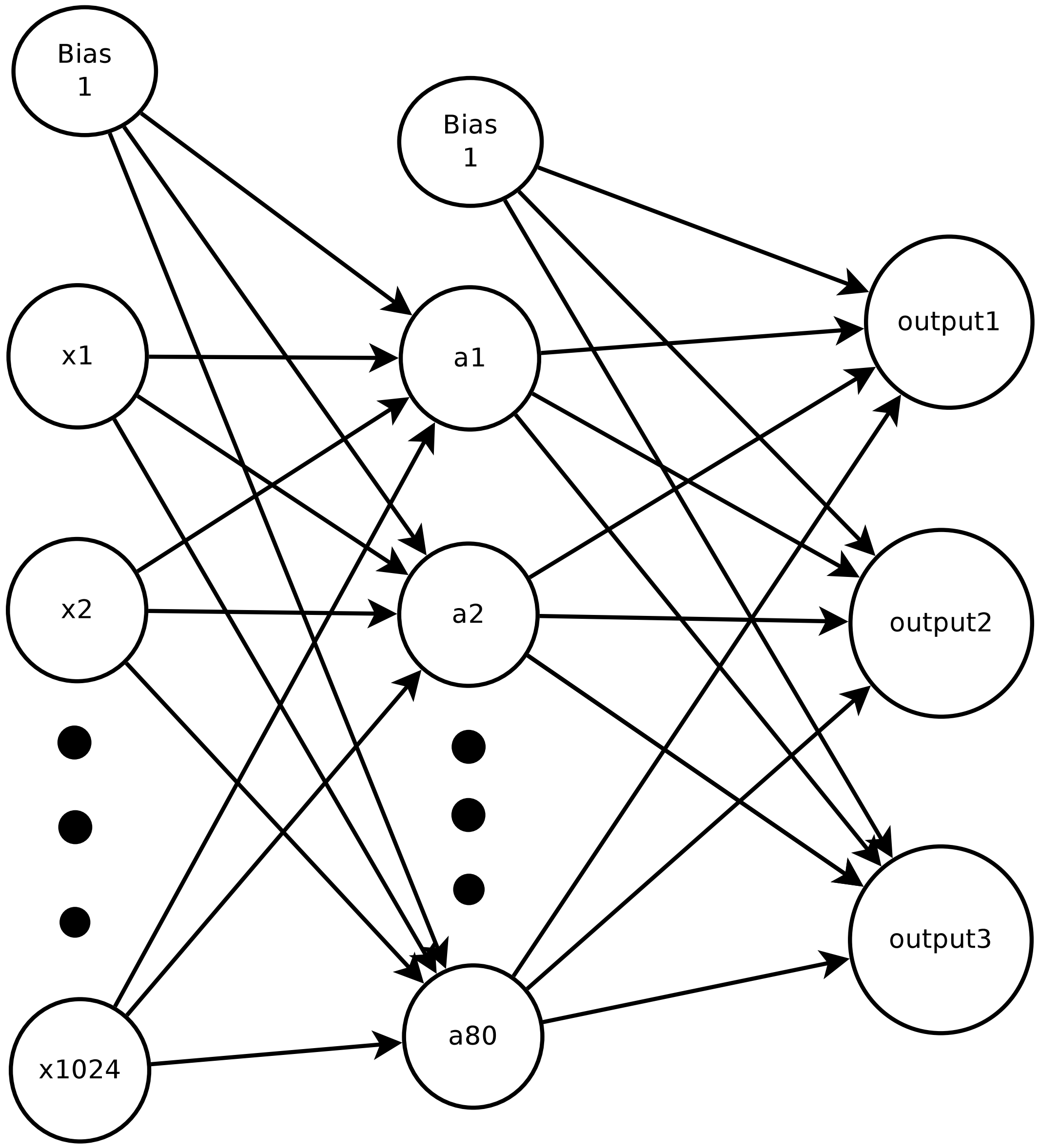


Figure 3.6: Structure of ANN used for WTC identification example

Results and Discussion

Figure 3.8 shows the original radar return which is the input to the ANN. In Figures 3.8 and 3.9, both X and Y axises are in unit of meters. The red spots are the wind turbine returns, the yellow smear around the positions of the wind turbines are the range sidelobes whose power is comparable to the

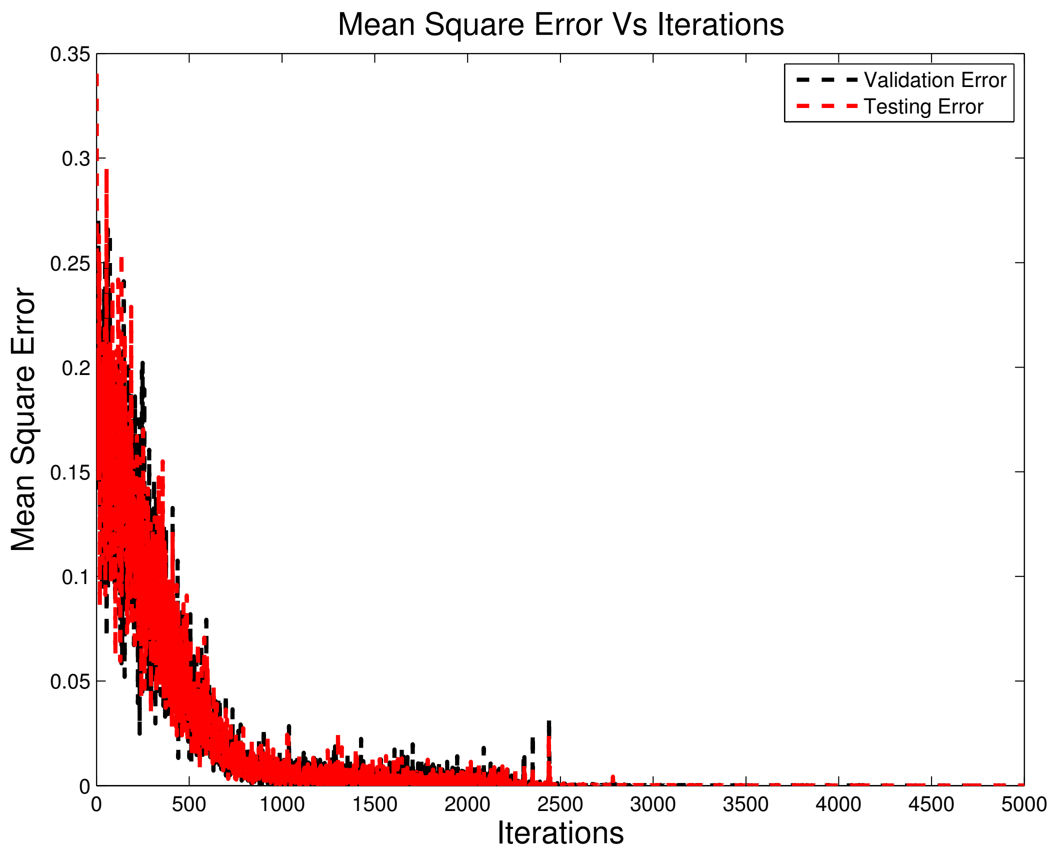


Figure 3.7: Cross-validation error against Iterations

Ground Clutter which is the yellow band around the middle of the figure[5]. The light blue area is just noise and the dark blue area represents the area where the noise is even lower.

Figure 3.9 is the output from the ANN. As can be seen in Figure 3.8 and 3.9, the network correctly identifies most of the Ground Clutter and the Wind Turbines. There are a lot of false alarms as well. In many instances, the side-lobe return from wind turbine is recognized as Ground Clutter.

[5]The red "smear" exactly in the middle is also Ground Clutter return, Ground Clutter return is highest in 0 degree Azimuth and is present mostly where the main beam (of antenna) illuminates the ground

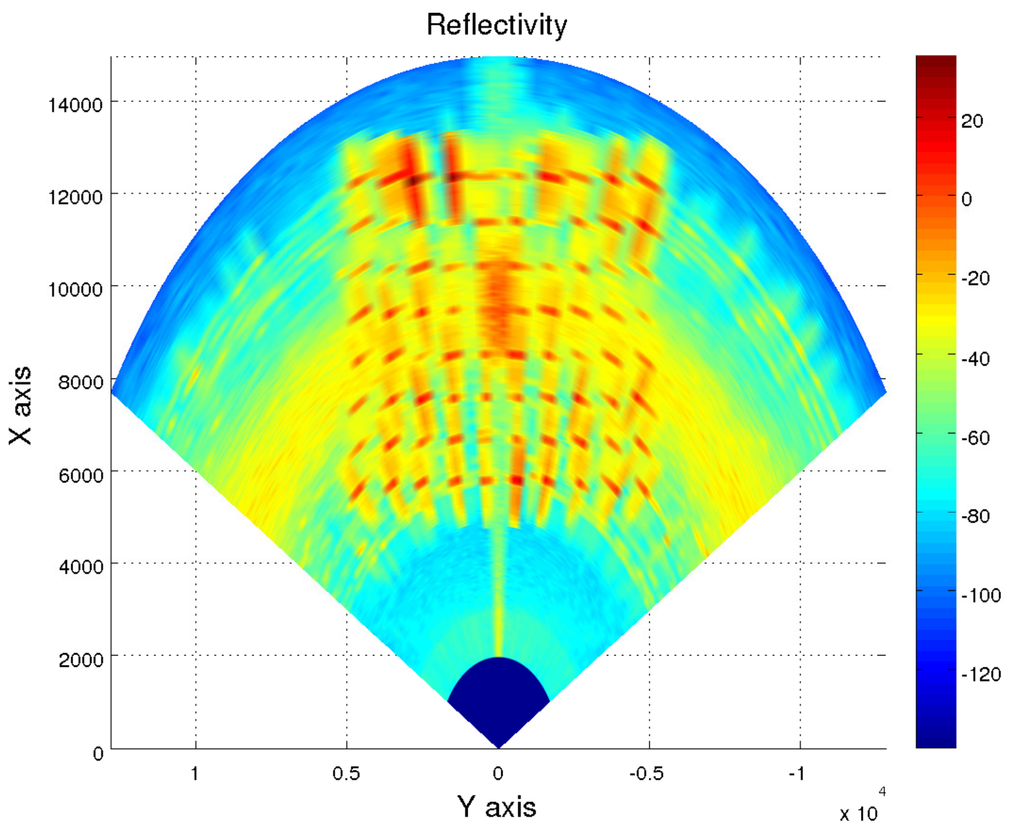


Figure 3.8: Simulated airborne radar PPI scan of a wind farm - power return [55]

At this point, the ANN can not recognize Noise, Ground Clutter and Wind Turbine with a very good precision. There are many false alarms in the recognition of WTC, which is mainly due to WTC sidelobes (they appear like WTC as well). Sidelobe reduction processing, such as the adaptive pulse compression algorithm (APC) discussed in a later chapter, could potentially yield a reduction of those false alarms. The main advantage of this ANN agent is its speed. The training, which is done beforehand, takes about 155 seconds ($\sim$2.5 minutes). There are 2500 iterations of 196 training examples with cross-validation scheme applied. The testing takes around 0.5 seconds for 36000 different testing examples. This ANN can act as an initial detector/recognizer that outputs the possible range-azimuth cell where wind turbine is present. It is the authors' thought that more sophisticated algorithms can be used only after initially using an ANN.

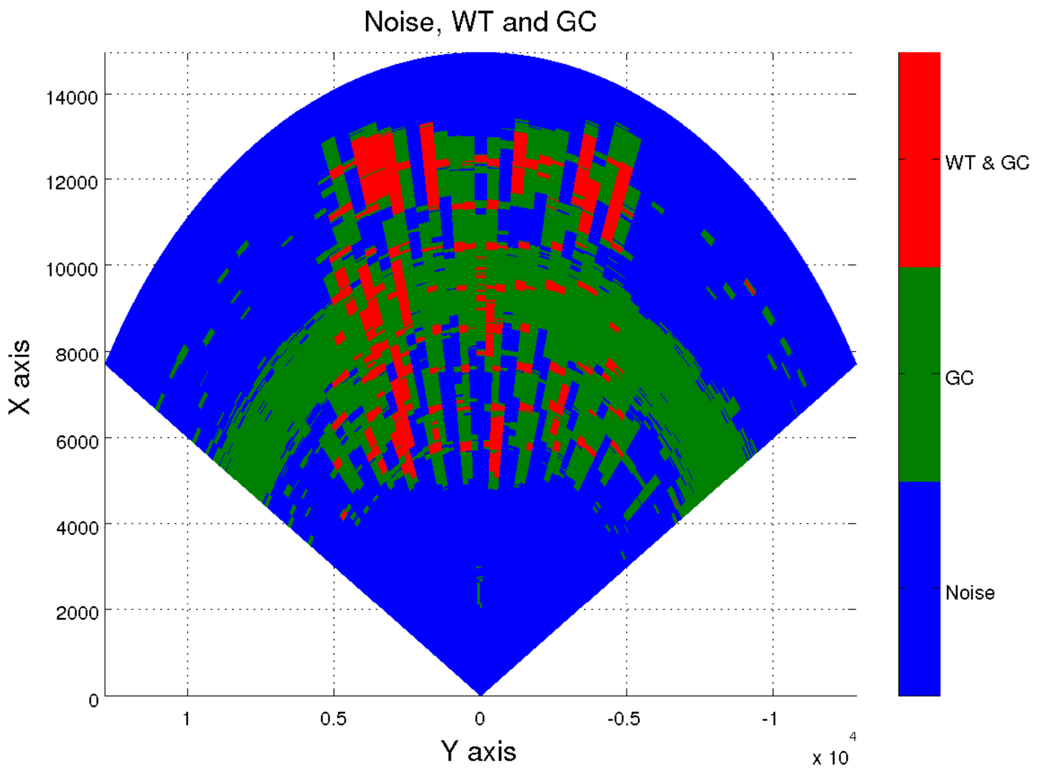


Figure 3.9: Simulated airborne radar WTC recognition output from the ANN

3.3.4 Unsupervised Learning Approach

The supervised learning approach used in Section 3.3.3 required priori knowledge of truths at each patch in the PPI scan as can be seen in Figure 3.8, which is difficult to obtain in some situations. In this section, we develop a new unsupervised learning approach based on ANN autoencoder, EM algorithm and Bayes classifier.

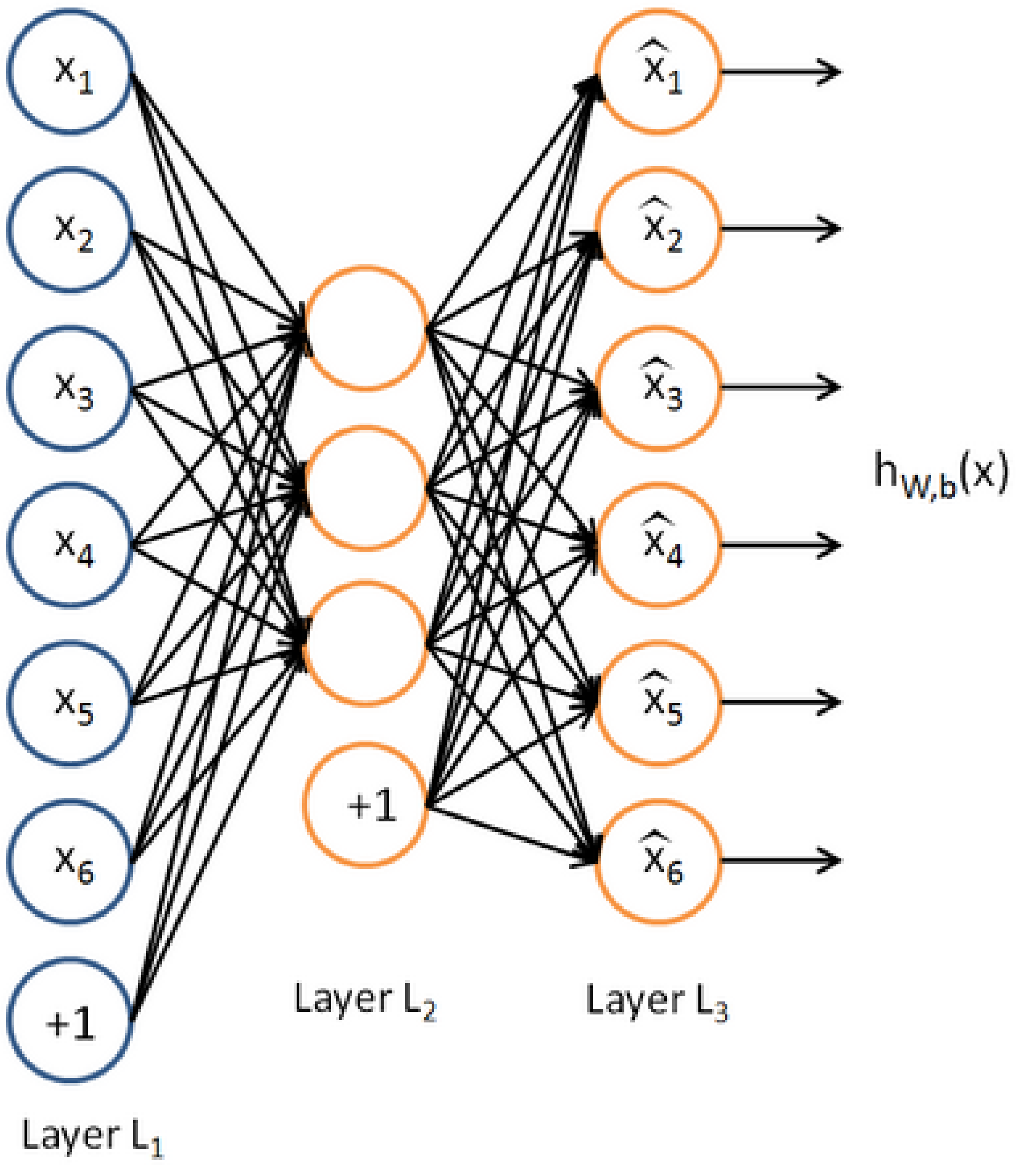


Figure 3.10: The structure of autoencoder

Autoencoder

Autoencoder, also called autoassociator or Diabolo network, is a method proposed in deep learning [56]. It trains the ANN by using the input as the output reference, applies the network to the data set and gets the values of neurons in one of hidden layers as output code. The structure of antoencoder is shown in Figure 3.10 [57]. In Figure 3.10, there is only one layer and the structure is symmetric. But in practice, the network may contain more than one hidden layer to tackle more complex data set and the whole structure does not have to be symmetric. In this chapter, we use five-layer symmetric ANN as the autoencoder.

EM Algorithm

EM algorithm was proposed in [58] as an iterative method for maximum likelihood estimates of parameters as used in statistical models. In this work, we use this algorithm to estimate the distribution of codes from autoencoder by assuming the codes follow independent multivarious Gaussian distributions.

Local maximum Problem There is a well-known problem associated with EM algorithm, i.e., the algorithm could fall into a local maximum, which prevents itself from finding the global maximum [59], [60]. Figure 3.11 is an example of EM algorithm trapped into a local maximum[6]. There are four 2D-Gaussian distributions centered at $(10,10),(10,-10),(-10,-10),(-10,10)$, where the standard deviations is 1. It is noticeable that, after applying EM algorithm, there is one distribution located near $(-10,10)$, two distributions located near $(-10,-10)$, and one distribution located near $(10,0)$ with a big standard derivation. It is not the set of distributions we used to generate those data points.

[6]Black circles represent data points generated by four 2D-Gaussian distributions. Stars represent the estimated μ of each distributions from EM. Dash lines and dash-dot lines represent the estimated σ of each distributions along with X and Y axle from EM

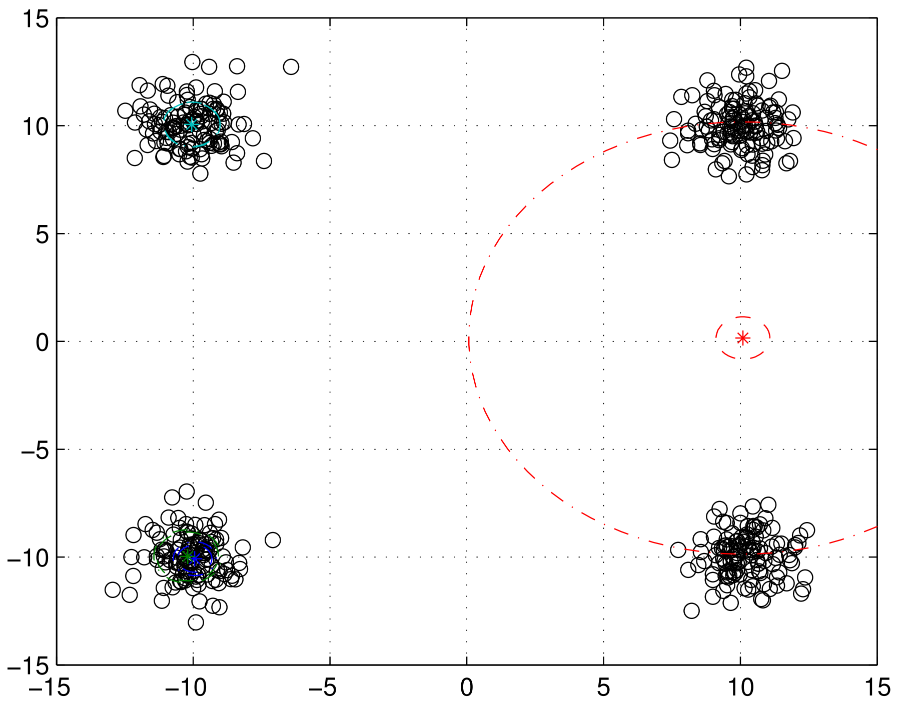


Figure 3.11: An example of EM algorithm outputs ”trapped” into a local maximum. x and y axles are first and second dimension of the simulated data points.

Solution There is a simple way to mitigate the local trapping effect by running EM many times with random starting points, then taking the highest likelihood one among them as the global maximum. But it is computationally expensive, especially when the number of starting points is large while the EM converges slowly [60]. Some researchers proposed a method called *multiple restart* strategy to reduce the computation load while taking multiple starts [61].

In this chapter, we propose a simple method to avoid the local maximum trap by training EM with more distributions than there really are and choos-

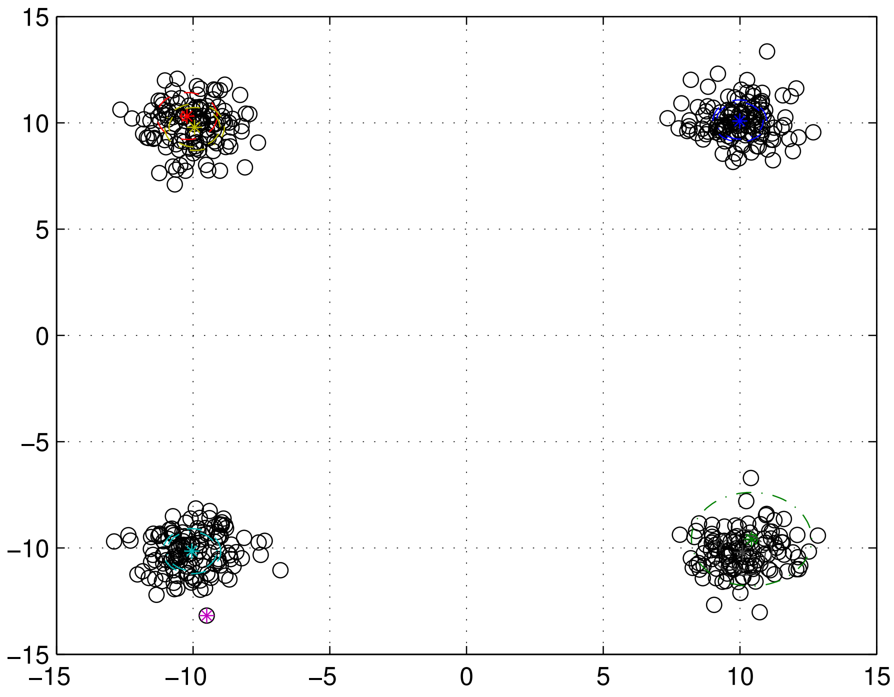


Figure 3.12: An example of EM algorithm with more distributions assumed than the truth.

ing some of them to form the global maximum result based on the rank of utilizing factor among them. Figure 3.12 illustrates how this method works. Notice that there are more than one distributions centered in some of clusters, but there is at least one distribution centered in one cluster. In fact, the distributions with highest four utilizing factors are centered in different clusters.

Bayes Classifier

Bayes classifier is a Bayesian approach of classifier to minimize the probability of misclassification [62]. In this chapter, we use this approach to classify the code from autoencoder by the maximum posterior probability using the distribution derived from EM algorithm.

Applying Process

Get Autoencoder We used five-layer symmetric ANN network as our autoencoder. The input data we used are the vector of magnitude values of discrete Fourier transform of radar signal return time series at a specific range and azimuth angle[7]. The whole PPI scan of data used is shown in Figure 3.8. The size of input at each range cell is 128 and the numbers of nodes inside ANN from input layer to output layer are $128, 24, 4, 24 and 128$.

Get Distributions We assumed the distributions of codes from autoencoder along with each dimensions are multi-dimensional independent joint Gaussian distributions. We used seven distributions to train the EM algorithm, and the algorithm stops when the difference of distribution utilization factor between two consecutive iterations is less than 10^{-6} or the times of iteration reaches 500.

Apply Bayes Classifier Through the EM algorithm in Section 3.3.4, we got seven groups of μ and σ values where each group consists of four $\mu - \sigma$ pairs along with four code dimensions. Then, we pick some of distributions with highest utilization factor as reference. Finally, we apply equation 3.2 to get a MAP[8] classification of each point in data set.

[7]radar resolution cell
[8]Maximum a Posteriori Estimation

$$\begin{aligned} C^{\text{Bayes}}(x) &= \underset{r\in\{1,2,\ldots,K\}}{\arg\max}\ \mathrm{P}(Y=r|X=x) \\ &\textit{where } C\colon \mathbb{R}^d \to \{1,2,\ldots,K\} \end{aligned} \tag{3.2}$$

Results

The classification results when 4 and 7 distributions are used are shown in Figures 3.13 and 3.14. In Figure 3.13, the agent successfully classified wind turbines, colored in cyan, ground clutters, colored in green, and nothing, colored in blue. And the red ones should be the sidelobes of ground clutters. In this configuration, it is impossible for the agent to distinguish wind turbines from their sidelobes. In Figure 3.14, the agent successfully classified the categories mentioned above and gave us more information. Black, cyan and yellow green, green and magenta ones represent wind turbines, sidelobes of wind turbines, ground clutters, respectively. In this configuration, the agent could classify more precisely and even distinguish some of the wind turbines from their sidelobes successfully. But, the results are much more ”noisy”[9] with more distributions used.

Discussion

Using unsupervised learning approach, we avoided the difficult part of machine learning approach of classification in some cases, which is obtaining the priori knowledge of training data. In addition, this approach could even provide us better performance and more information than what we expected. But, as a drawback, the agent could only classify instances from data set into different

[9]More ”false alarms” or misclassifications

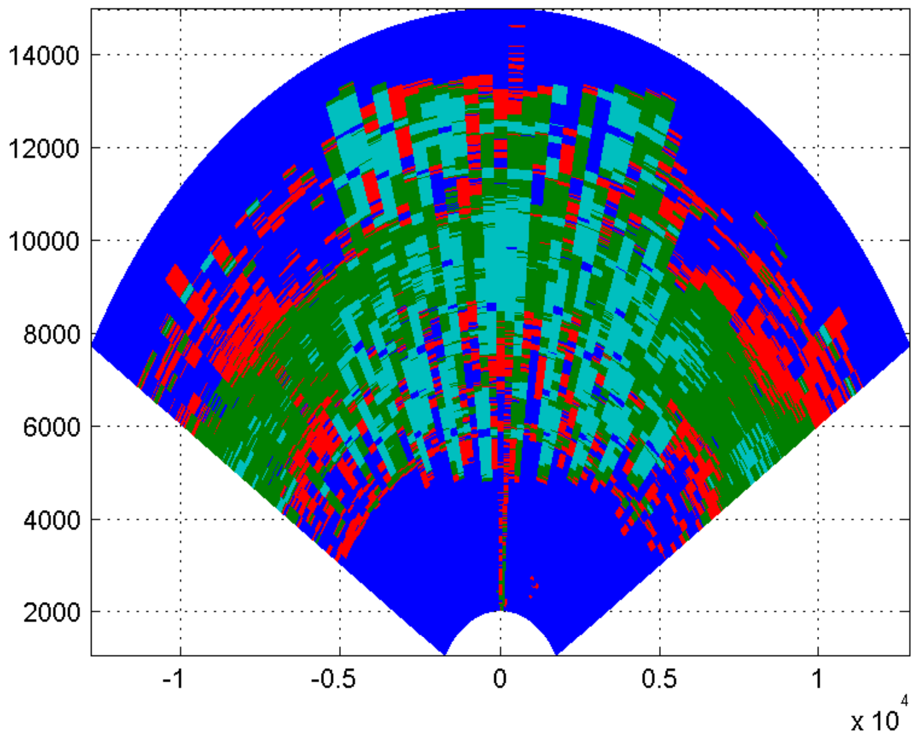


Figure 3.13: Classification result using 4 distributions. Each color represents a category obtained from EM algorithm.

groups without giving any information about the labels of each group, which is understandable since it is an unsupervised learning approach and we have never trained the agent with labeled information.

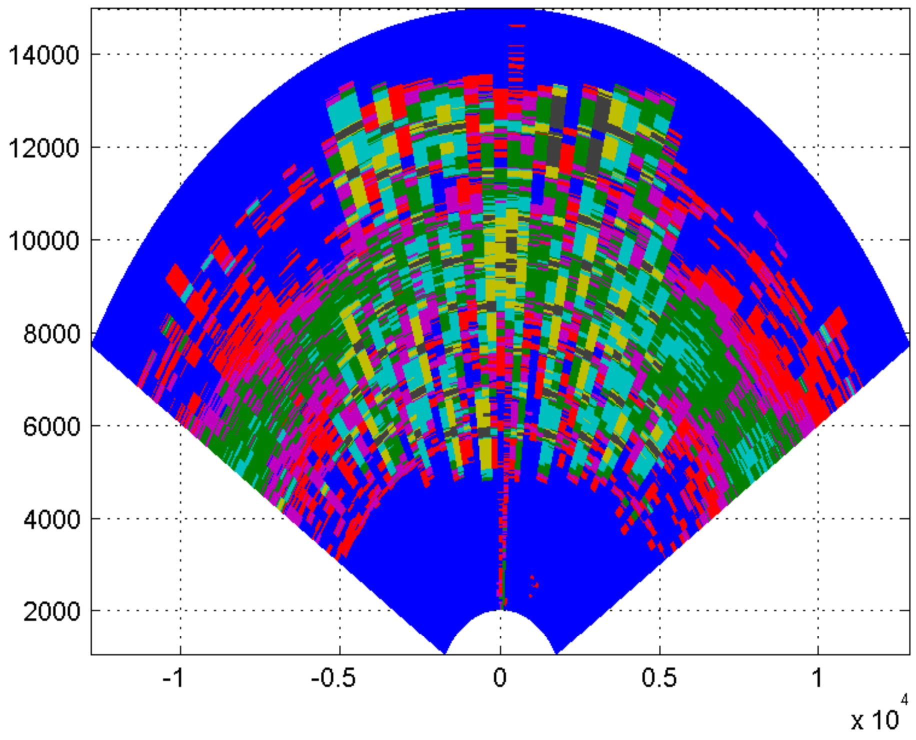


Figure 3.14: Classification result using 7 distributions. Each color represents a category obtained from EM algorithm.

Chapter 4

Turbulence Detection and Classification using HKO-TDWR with AI Algorithm

4.1 Introduction

Turbulence has been a major threat to aviation safety, and detecting turbulence reliably and in real time using different sensors has been a significant challenge since 1960s [13], [14]. There have been many developments in measuring turbulence affecting safety of flight with the support of onboard in-situ sensors, remote turbulence sensors such as LIDAR, airborne and terminal Doppler radar, and a large body of theories based on physical knowledge to develop algorithms that relate radar measurements to the strength of turbulence [63]–[66]. Compared to Doppler LIDAR, radar is less sensitive to the adverse weather conditions such as snow, fog and rain [67]–[69]. Compared to airborne radar, ground-based radar such as Terminal Doppler Weather Radar (TDWR) is less subject to ground clutter, able to provide fast coverage of larger regions remotely, and provides data sharing among networked users. The challenges of using ground radar, however, have been largely limited by the data quality of the Doppler spectrum width (SW), which is well-documented in prior studies [40], [70]–[72]. For example, SW data is often affected by noise as it requires an order of magnitude higher SNR for acceptable accuracies compared to Doppler velocities [70]. Moreover, windshear of steady wind biases turbulence measurements as well as multiple-trip echo does [70], [71], [73]. Even through the TDWR has used some mitigation algorithms [72], it is still very difficult to physically model and correct all the error factors. Error modeling needs precise correlation between the radar data and on-board in-situ measurement, which is usually difficult to achieve. As an additional challenge, both high-altitude EDR and low-altitude EDR estimation and turbulence detections are important for aviation. Turbulence near-ground is more frequent and complicated due to the impacts from terrain and surrounding ground features that can cause severe turbulence [74]. It is desirable for radar to be capable of accurately monitoring turbulence under precipitation conditions, at different altitudes, and possibly incorporating other radar measured variables, such as reflectivity and radial velocity, besides spectrum width.

The US National Weather Service's WSR-88D had used a turbulence detection algorithm (TDA), and now it uses the NCAR Turbulence Detection Algorithm (NTDA) [72]. The data quality of the turbulence monitoring products from these algorithms has been limited by the challenges mentioned above. Furthermore, these algorithms were mainly developed for en route turbulence detection, whereas the 5-cm wavelength TDWR provides higher angular resolution and low-altitude rapid-scan coverage of the airspace in the approach and departure lanes into and out of the airport, as well as better resolution in the complete air space surrounding the airport.

Because of data quality issues and limitations of physical models, EDR estimations based on radar measurements have not been reliable. Thus application of Machine Intelligence (MI) or Artificial Intelligence (AI) approaches for radar turbulence detection has received increasing attention recently [75], [76]. In NTDA for example, Fuzzy Logic algorithm is used for SW data quality control by adding “confidence” in measured data. Simple Bayesian detection method was used in [77]. More exploration of using random forest algorithm by combining multiple sensor data was discussed in [75]. These pioneering works show some promise. However, they are limited by either the specific training and test cases, or availability of truth data. Similar to other MI/AI algorithm applications, there are many ways for parameterization, training and testing. As one of the motivations of this work, the stability, reliability and risk for “over-training” of these algorithms needs to be further understood. Also, there is the challenge of selecting appropriate “feature vectors” for MI/AI algorithms, and there are many different ways to combine them; there has been no systematic method for optimizing them. One of the aims of the current research is to address these challenges by using a carefully designed training and verification approach based on Artificial Neural Network (ANN).

The MI algorithm is the core of the turbulence estimation process. The innovation of this work is combining the existing physical models of the aviation turbulence and the “data modeling” approach in machine intelligence algorithms. Massive amount of measurement data is provided by HK Observatory (HKO), which contains both TDWR measurements and in-situ measurements provided by airlines which serves as the basis for training database. In spite of the amount of data available, challenges of data quality control and lack of “balanced” training data sets (e.g., the training data representing each specific turbulence class) for MI/AI algorithms remains. The “radar feature vectors” are selected based on their physical meaning, mutual correlation, and resultant classification performance. A training algorithm is also developed based on pre-processed and pre-selected sample groups. The training data samples are not only the raw-measurement, but also “processed” samples based on physical models. With this unique approach, the algorithms are trained strategically using resampled data sets, and then used to classify or predict EDR levels.

There are significant benefits to this approach. First, all the uncertainties of data quality are “built in” training data [78]. Second, physical knowledge (related to turbulence process and radar observation) provides a way to “tune” the training data that emphasize the fundamental common physics for training and testing cases, and reduces the differences of variables between training and testing. This approach is essentially a novel way of improving data quality which allows for full-incorporation of uncertainties in the data. Third, this approach is more robust and less subject to the mismatch between radar measurement and in-situ measurements, and finally the algorithm training process is usually able to discover the underlying pattern of the data automatically. The trained algorithm is then verified using the HKO data sets, and a series of promising results are obtained.

This chapter is organized as follows. Section 4.2 provides an overview of the turbulence detection system. Section 4.3 discusses the radar data quality control and pre-processing of radar data. The development of feature vectors used in MI algorithms is discussed in Section 4.4. The details of training and testing algorithms for turbulence detection are discussed in Section 4.5. Test results from MI algorithms are presented in Section 4.6, and the conclusion remarks are given in Section 4.7.

4.2 Concept of Operations

Figure 4.1 depicts the overall concept of operation for aviation hazard detection system based on radar observations. This is a general framework that can be extended to different radar platforms, and the application to HK-TDWR radar is an example. For the envisioned field operations, recorded data are used mainly for training, and the real-time measurements are used for testing. In lieu of real-time data we use other archived data (i.e., archived radar data II)

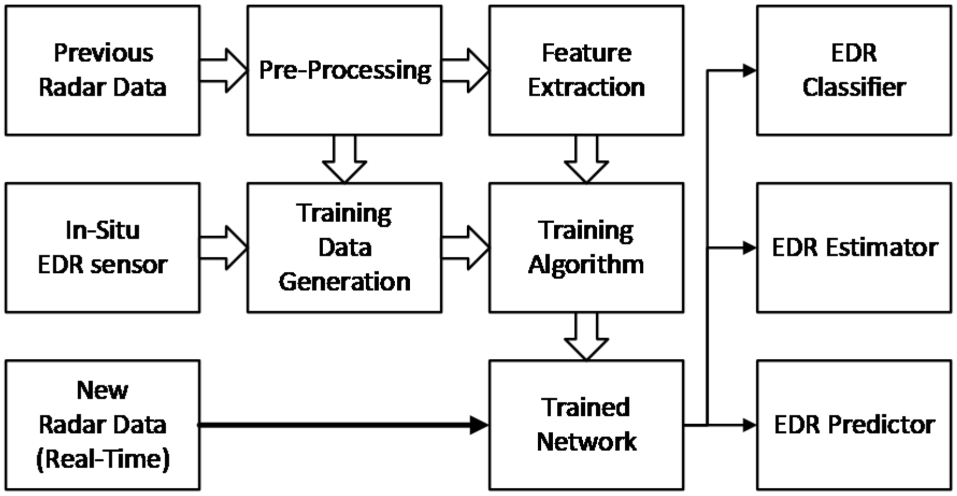


Figure 4.1: Overall concept of operation for aviation hazard detection.

Processing is an important step for both training and testing. The first step of processing is data editing, which monitors the data quality metrics (such as SNR) before the next step of processing. Because all out-of-trip echoes are incoherent, the SNR in this article is an effective SNR in which noise power is the sum of out-of-trip echo powers in each resolution volume (V_6s [40]), and receiver noise power. The second step of Processing is computing the turbulence hazard index (EDR$^{1/3}$) using physical models. The third step is correlating in-situ measurements of EDR$^{1/3}$ with those deduced from radar. In this step, the detailed computing and selecting V_6s for turbulence classifications are performed.

Processed radar measurements are combined with in-situ EDR$^{1/3}$ sensor data for generation of training data for MI algorithms. Another step is extracting feature vectors for the MI algorithms. Feature extraction is challenging because we need the feature vector to be sufficient while avoiding over-training. Supervised training [79] is used for this study, in which the training data contains both radar measurements and in-situ "truth" data.

The key for the training algorithm is how to optimally combine and utilize training data and feature vectors. For the Artificial Neural Network (ANN) used in this work, training outcome is a set of networks with different weights. The trained ANNs are ready for receiving real-time incoming radar data for these possible functions: (1) EDR$^{1/3}$ classifier for turbulence hazard detection, (2) EDR$^{1/3}$ estimator, which can be used to "track" the turbulence level changes, and (3) EDR$^{1/3}$ predictor, which uses the current radar measurements to "forecast" the turbulence status 5-10 minutes later. The current work focuses mainly on the EDR$^{1/3}$ classification and prediction functions.

4.3 Pre-Processing of TDWR Data

4.3.1 Background Information

The data source of this study is the TDWR radar about 10 km northeast of the Hong Kong International Airport [80]. A photographic image of the airport and the location of TDWR radar is shown in Figure 4.2. Note there are two main runways (07R/25R and 07R/25L) on the airport, which are marked as red lines. The period of data collection is from Oct 04 2015 to Oct 05 2015 during the Typhoon Mujigae event. During processing, we interpolate GPS recordings (at 1Hz rate) to the same time that EDR$^{1/3}$ data are recorded (at 4Hz rate). More details on how the onboard EDR$^{1/3}$ is computed is elaborated in [2].

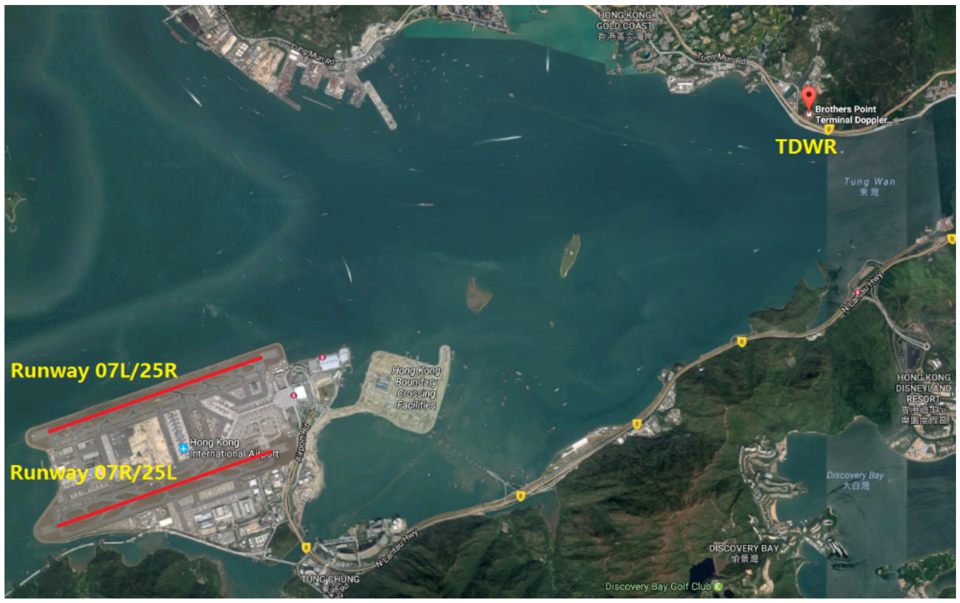

Brothers Point Terminal Doppler...
TDWR
Tung Wan
Runway 07L/25R
Runway 07R/25L
Hong Kong International Airport
Hong Kong Boundary Crossing Facilities
HONG KONG GOLD COAST
HONG KONG DISNEYLAND RESORT
Discovery Bay
DISCOVERY BAY
Discovery Bay Golf Club
TUNG CHUNG

Figure 4.2: Photographic image of the HK airport and the location of the TDWR radar.

The key parameters of the TDWR radar are listed in Table 4.1 [38].

Table 4.1: HKO Brothers Point TDWR Radar Specs

Parameters	**Values**
Location	22°21'31"N, 114°1'16"E
Frequency	5.625 GHz
Antenna systems	7.9 m Metal Parabolic Reflector. Linear polarized. Beamwidth $\theta_1 \leq 0.55°$, gain 50.54 dBi. Sidelobes are $\leq$ 27 dB within 5° and $\leq$ 40 dB beyond 5°.
Scanning	Hazardous weather and monitor modes with different elevation coverages; 5 mins per volume scan, azimuth scan speed 24° per second.
Transmitter	250 KW peak power Klystron, 500 watts average, transmitted signal phase is uncorrelated from pulse to pulse.
Waveform	Pulse width 1 μs, Dual PRF 1.104 KHz and 1.38 KHz. $\pm$60 m/s non-ambiguous velocities, occupies 4 MHz bandwidth.
Receiver and processor	Receiver noise power: - 111 dBm at 1 MHz bandwidth; Dynamic range is 100 dB; 3 dB noise figure; ground clutter suppression > 50 dB and velocity accuracy $\pm$ 0.5 m/s
Display properties	0 to 90 km, 150m range bin spacing, 600 range bins per radial, 0.70° azimuth spacing

The HKO-TDWR was operated under a hazardous weather mode, whose Volume Coverage Pattern (VCP) contains 15 elevation angles. Figure 4.3 shows an example of the reconstructed elevation scan profile of VCPs. The red text in Figure 4.3 is discussed in Section 4.3.5. It can be observed that the

VCP takes about 4 minutes and 50 seconds to complete and there are multiple measurements at the lowest elevation, which is about 0.5 degree.

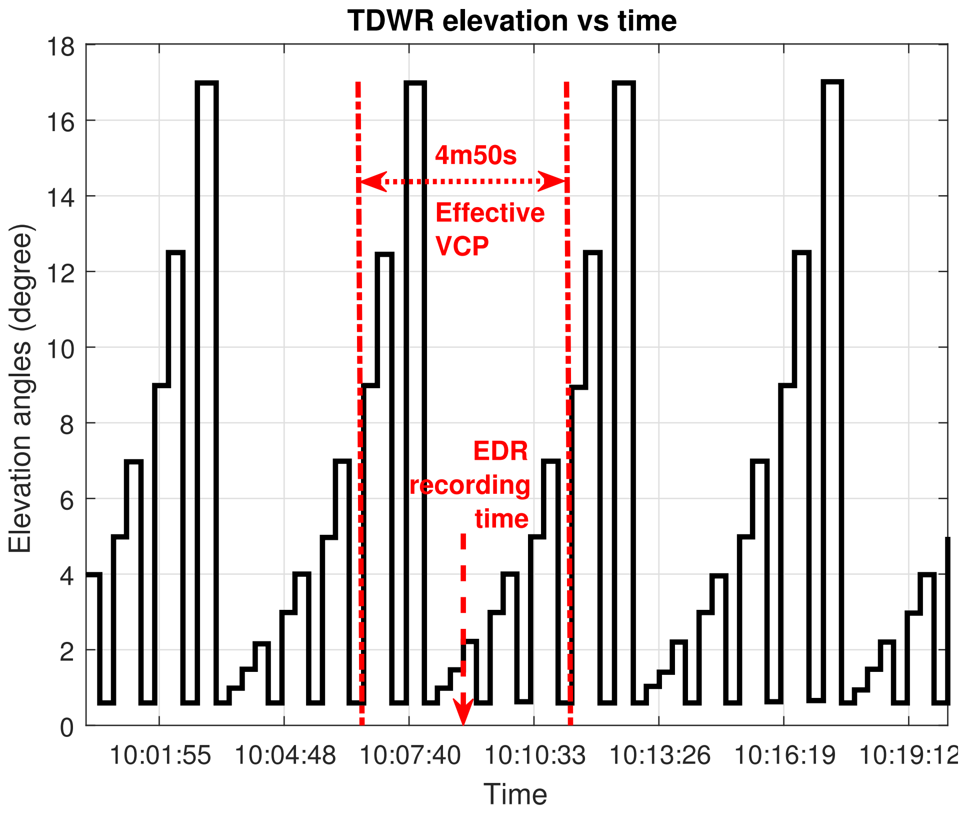
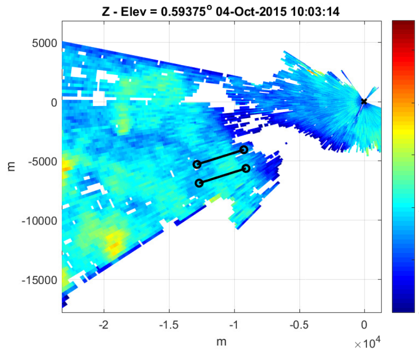


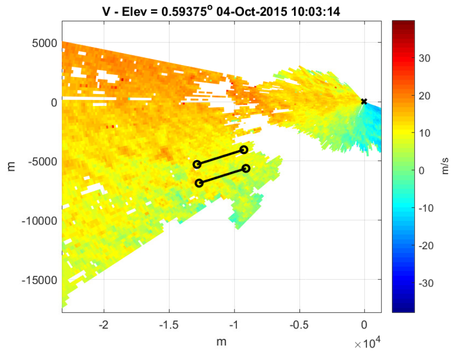
Figure 4.3: The elevation angles of the conical scans vs time, and a span of time embraced by red vertical lines within which radar data was used to construct Vertical Profiles of Radar Variables (VPRVs) shown in Figure 4.6.

4.3.2 Initial Radar Data Visualization

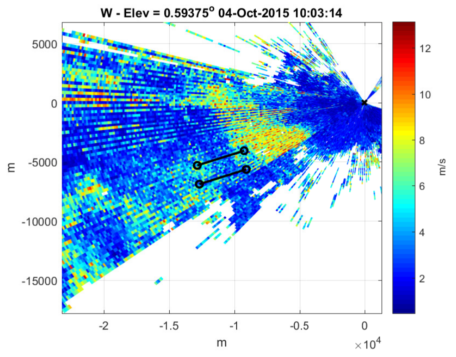
Figure 4.4 shows the initial “readout” of the TDWR scans and overlap plotting with runway and radar locations. In these figures, black cross represents the location of TDWR and black solid lines represent north and south runways of HKIA, respectively. The four key data parameters are: reflectivity (Z), radial velocity (V), spectrum width (W) and signal-to-noise ratio (SNR). As the first step of data quality control, some of the data points have been removed (the white blank spots) mostly due to out-of-trip echoes that bias Z and increase the Standard Deviation (SD) of V and W. In spite of missing data, most likely due to overlaid out-of-trip echoes, we found that even when data voids were over the airport, the trained neural network, because of its robustness for classification [76], [77], is still able to produce outputs.



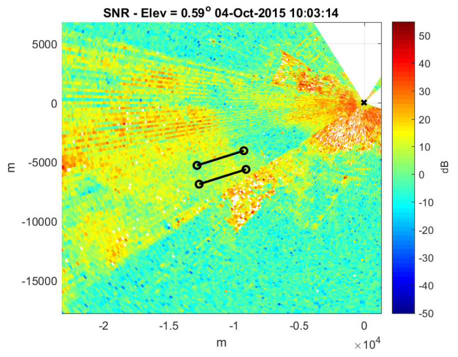
(a) Z



(b) V



(c) W



(d) SNR

Figure 4.4: Example measurement data product from a PPI scan of the TDWR, Reflectivity (Z), Radial velocity (V), Spectrum width (W) and Signal-to-Noise Radar (SNR).

4.3.3 General Statistics of Turbulence Scenarios

The first result of data analysis is that the actual occurrence of turbulence is a small fraction of the measurement cases, which is consistent with actual operational conditions and previous discussions in Section 4.5. Even in an event of typhoon, such as the cases studied in this experiment, negligible turbulence would be expected in the majority of the flying time. Table 4.2 summarizes the overall classification results (using the classifier developed later in this Chapter) for all the data sets available. Obviously, the majority of the detection outputs from the classifier are for low-altitude and negligible turbulence cases, which reflects the nature of turbulence distribution according to different altitude levels. Also, since the low-altitude turbulence may have more catastrophic impacts, it is reasonable to have accurate detection results for the low-altitude scenarios.

4.3.4 Signal-to-Noise Ratio (SNR) and Spectrum Width Measurements

The spectrum width σ_v (or W) estimates are computed using the HKO-TDWR's signal processor (manufactured by Misushibi) and an equation equivalent to Eq.(6.27) of [40], which is shown in Equation 4.1,

$$\hat{\sigma}_v = \frac{\lambda}{2\pi PRT\sqrt{2}} | \ln \frac{\hat{s}}{|\hat{R}_1|} |^{1/2} \tag{4.1}$$

Table 4.2: Number of Detections by Classifier on Altitude vs Turbulence Category

Category / Alt(m)	**Negligible**	**Slight**	**Moderate & Above**
0-500	241532	133446	1292
500-1000	380768	114962	1328
1000-1500	257396	85321	1152
1500-2000	281940	55909	856
2000-2500	278126	35058	612
2500-3000	250450	20533	360
3000-3500	270882	14260	232
3500-4000	262581	11856	164
4000-4500	259528	5556	124
4500-5000	279996	13064	92
5000-5500	246448	2380	72
5500-6000	254661	3028	68
6000-6500	258045	6524	68
6500-7000	224046	648	36
7000-7500	220897	1340	20
7500-8000	202460	2016	16
8000-8500	174776	1580	8
8500-9000	180244	2800	0
9000-9500	140712	704	0
9500-10000	138172	588	0
>10000	1888952	44	0

However, because noise power in the Doppler spectrum of HKO's TDWR is an unknown variable related to unknown out-of-trip echo power plus receiver noise power, the signal power estimate is computed by the TDWR processor using lag zero estimates ($\hat{R}_0$) of the autocorrelation of echo samples multiplied by SNR(SNR+1)$^{-1}$ [40]. The SNR is obtained from estimates of the autocorrelation magnitudes at lags zero ($\hat{R}_0$), one ($\hat{R}_1$), and two ($\hat{R}_2$) as suggested by [81].

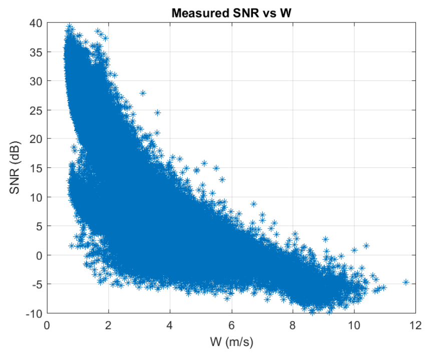
The fundamental physical model for describing the relationship between radar-measured spectrum width $\sigma_v \equiv W(m \cdot s^{-1})$ and $\varepsilon(m^2 s^{-3})$, i.e., EDR, is given by Equation 4.2 (Section 10.3 of [40]).

$$\varepsilon \approx \begin{cases} \dfrac{0.72\sigma_v^3}{r\sigma_\theta A^{\frac{3}{2}}}, when & \begin{matrix} \sigma_r \leqslant r\sigma_\theta \\ r > 17188m \end{matrix} \\ (\dfrac{\sigma_v^3}{\sigma_r(1.35A)^{\frac{3}{2}}})(\dfrac{11}{15} + \dfrac{\frac{4}{5}r^2\sigma_\theta^2}{\sigma_r^2})^{-\frac{3}{2}}, when & \begin{matrix} r\sigma_\theta \leqslant \sigma_r \\ r \leqslant 17188m \end{matrix} \end{cases} \tag{4.2}$$

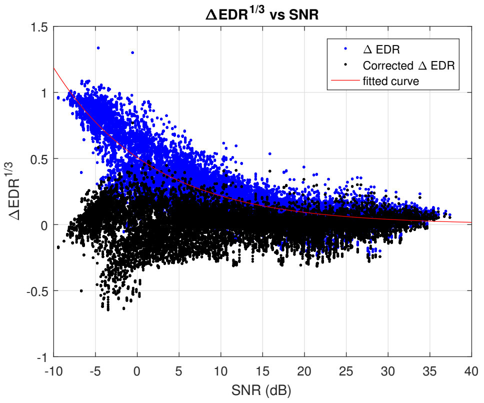
In Equation 4.2, σ_r is range resolution in meters, σ_θ is angular resolution in rad (values of σ_r and σ_θ are from Table 4.1), r is the range of a particular radar resolution volume in meters, A is a universal dimensionless constant between 1.53 and 1.68, and 1.53 is used in this study. Equation 4.2 only provides an estimate of EDR$^{1/3}$ since biases due to steady wind shear etc. are not included. However, it can be further "calibrated" using in-situ measured EDR$^{1/3}$ using the approach described below, which is then sufficient for the MI/AI algorithms in this study.

Two of the four available radar measurements from HKO's TDWR are SNR and spectrum width (W), and they are not independent of each other. As shown in Figure 4.5a, when the SNR is low, W tends to be high, and vice versa. The possible reason for this is that the spectrum width computation assumes Gaussian shaped spectra and unbiased SNR estimates. It can be seen that if SNR is over-estimated at low SNR, W measurements at location of low SNR will be higher. In addition, when SNR is low (below about 15 dB), W estimates have larger variance as also seen in Figure 4.5a.

According to the Equation 4.2 which assumes V_6 is uniformly filled with isotropic turbulence, W is the single parameter (other than the range r) that determines the EDR value. Therefore, to get most benefit of W values, a correction process must be applied to the calculated EDR based on Equation 4.2, a correction that depends on the corresponding SNR measurement. This correction is based upon comparisons of EDR$^{1/3}$ calculated from Equation 4.2 and EDR$^{1/3}$ obtained from in situ measurements.



(a) Estimated SNR vs estimated W.



(b) Calculated $\Delta EDR^{1/3}$ (blue dots) and the corrected $\Delta EDR^{1/3}$ vs estimated SNR (black dots).

Figure 4.5: Scatterplots of all 560,000 collected data.

Figure 4.5b shows the EDR estimation errors as $\Delta EDR^{1/3} = EDR^{1/3}_{Calculated} - EDR^{1/3}_{insitu}$ before (blue dots) and after (black dots) bias correction. $EDR^{1/3}_{Calculated}$ is obtained from W estimates and Equation 4.2. From the plot before correction, a clear trend of bias (solid red line) of $\Delta EDR^{1/3}$ at various SNRs is observed. In this work, we choose an exponential fitting to estimate bias caused by different estimated SNRs, and then offset the bias in the correction procedure.

4.3.5 Association of In-situ Measurements with Radar Measurements

Because of potential mismatch of the aircraft's onboard GPS location recording and TDRW radar's resolution cell location, as well as the possible location errors due to radar resolution limitation, it is important to appropriately associate the onboard in-situ measurements from the aircraft and remote radar measurements from the TDWR before further processing. The goal is to minimize the temporal and spatial difference between the two sources of measurements. We illustrate the algorithm used here as an example and part of training algorithm. First, using GPS data from the onboard flight recording, time and location information of the aircraft for every EDR measurement can be derived. In addition, using range, azimuth, elevation and time information from the radar measurement and the GPS coordinates of the TDWR, the correspondence between EDR and the location and time of radar measurements can be established. However, the time and altitude difference between EDR and radar measurements is inevitable. To mitigate the impact of those differences on the performance of the classifier under training, and to obtain predictive estimates of turbulence, along about 100 km aircraft path approaching and departing HKIA, the radar measurements within the whole Volume Coverage Pattern (VCP) are used.

Because each of the VCPs used in this study took about 4 minutes and 50 seconds to complete and consisted of 15 different scanning cones, all radar measurements, that fall into $\pm$ 2 minutes 25 seconds span of the time during which the in situ EDR data was obtained, are used to construct vertical profiles of averaged TDWR measurements above and below the path of the aircraft. The data that are averaged are those 9 within plus minus one resolution volume in range and azimuth closest to the vertical at the flight location. We label these profiles as VPRVs (Vertical Profiles of Radar Variables). The constructed VPRVs are shown in Figure 4.6. Therefore, the maximum temporal offset between the time of measured EDRs and the time of averaged radar measurements is 2 minutes 25 seconds. Because the aircraft is in the VPRV, effects of descending high Z regions associated with strong downdrafts that create wind shear and turbulence that affect safety of flight can also be part of the detection and prediction of turbulence used by the AI algorithm.

4.4 Development of Feature Vectors

Large amount of tests were performed in order to determine optimal feature vectors for turbulence detection or classification. As a common guideline, the feature selection needs to minimize redundancy information while containing useful characteristics. The tests start with using all possible features to train and then test gradually down-select them to the combination that provide the best end-results for turbulence detections. The vertical profiles of averaged radar data were found to contain most useful feature information for turbulence. In Figure 4.6 we display one of the examples (CX134-1 case) showing vertical profiles of radar variables at different times. The radar variables are overlaid with recorded aircraft altitude along the flight path, the aircraft's distance to the radar, and the in-situ measured EDR$^{1/3}$. It is interesting to observe that high EDR$^{1/3}$ values correlate with certain patterns of the vertical profiles of radar data.

Radar measurement data from the entire vertical profile can be used as part of feature vectors. Also as observed in Figure 4.6, it is useful to show radar measurements not only from the VCP around the flight time but also earlier VCPs, because they could provide information about the evolution of weather features such as intense reflectivity cores descending, which often are associated with microbursts and downdrafts that can generate dangerous low altitude wind shear and turbulence [82].

Table 4.3: Structure of Training Vector of Features

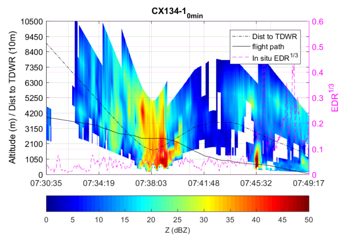
Feature Names	**Feature Numbers**	**Notes**
Calculated EDR value	1	After correction
Altitude of the aircraft	2	From GPS recording
Altitudes of radar scans	3-4	The lowest and highest points of a given vertical profile
Vertical profiles of reflectivity	5-49	15 measurements in a profile, 3 profiles for 3 different times.
Vertical profiles of radial velocity	50-94	15 measurements in a profile, 3 profiles for 3 different times.
Vertical profiles of spectrum width	95-139	15 measurements in a profile, 3 profiles for 3 different times.
Vertical profiles of SNR	140-184	15 measurements in a profile, 3 profiles for 3 different times.

Another observation from Figure 4.6 is that the altitudes and azimuths of the aircraft need to be used to specify the portion of data from the VCPs to be used for training and testing. Since the spacing of altitudes is linearly determined in each VCP, only the lowest and highest altitudes are needed to construct the vertical profiles of radar data. Based on these observations and after extensive experiments, 184 features are extracted for training and testing. For every VPVR, the features include four types of radar products (Z, V, W, and SNR) at 0, 5 and 10 mins before the flight time at the aircraft GPS location. Thus, there are 180 features directly from radar measurements. In addition, the four "derived" features including calculated EDR$^{1/3}$ from radar measured spectrum width (with SNR-dependent bias correction), flight altitude from flight data record and the lowest and highest altitudes for radar measurements. (Again, these altitudes are included as features to specify the airspace being monitored, and only these two are needed since the scanning altitudes linearly span from lowest to highest.) Details of the 184 feature vector element are summarized in Table 4.3.

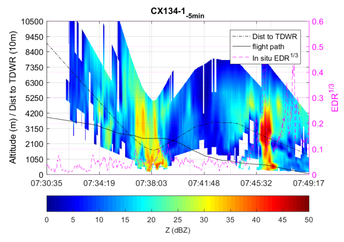
4.5 Training and Testing Algorithms

4.5.1 Objectives of Training and Testing Algorithms

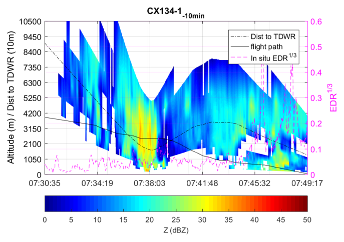
The goal of training and testing in this work is to classify the level of severeness of turbulence based on radar measurements. The turbulence categories used in this study are based on associated EDR$^{1/3}$ value shown in Table 4.4. The EDR$^{1/3}$-Turbulence Intensity Category relation is based on a medium-sized transport aircraft under typical en-route conditions [1], [83]. The supervised training process uses part of the measured radar data and truth data (onboard turbulence sensor) to generate the classifier. Because of its robustness to data quality degradation and popularity in MI community, ANNs are used as a classifier in this study. The testing process simply compares the classifier output



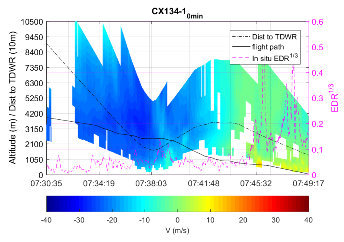
(a) Z, 0 min



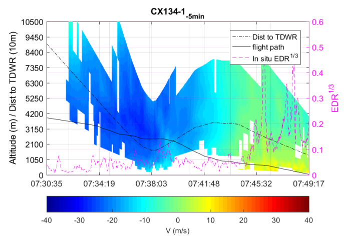
(b) Z, -5 min



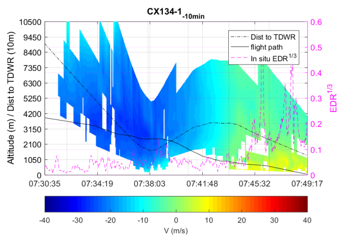
(c) Z, -10 min



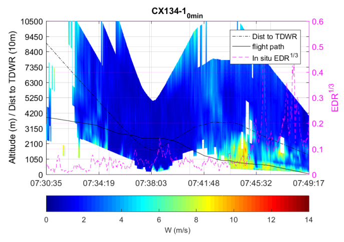
(d) V, 0 min



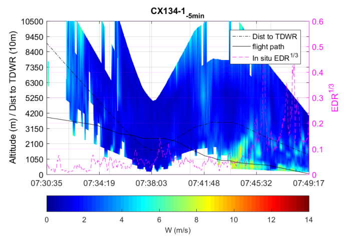
(e) V, -5 min



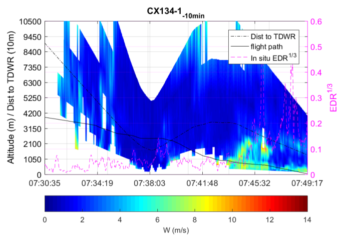
(f) V, -10 min



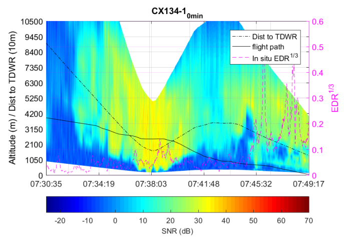
(g) W, 0 min



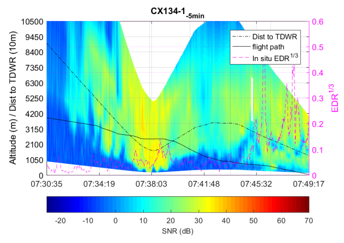
(h) W, -5 min



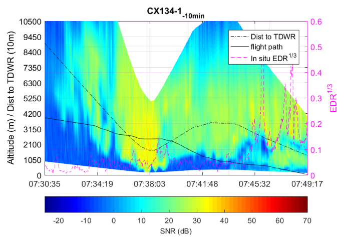
(i) W, -10 min



(j) SNR, 0 min



(k) SNR, -5 min



(l) SNR, -10 min

Figure 4.6: Vertical profiles of radar variables along with the flight path of CX 134 at 0, -5 and -10 minutes before the flight time.

according to Table 4.4 with the onboard EDR$^{1/3}$ measurements to determine the confusion matrices.

Table 4.4: Relation between EDR$^{1/3}$ and Turbulence Categories

EDR$^{1/3}$	<0.1	0.1-0.4	0.4-0.7	>0.7
Turbulence Levels	Negligible	Slight	Moderate	Severe
Turbulence Categories for Classification	1	2	3	3

4.5.2 Training Data Generation

After extracting radar variables for each GPS recording (after interpolation), the final size of entire data set is 567, 225 data entries. Each data entry (data point) in the dataset contains 184 radar-measured features. As shown in Table 4.5, the majority of data points are collected when the aircraft experiences little to no turbulence. Moreover, when severe turbulence is experienced, the altitude of the aircraft is usually low. These unbalanced data distributions will introduce a bias to the ANN toward low EDR$^{1/3}$ classification, except for those low altitude cases if these data are used directly for training. Furthermore, it is undesirable to let our classifier to heavily rely on the altitude to make the detection decision, because that will affect the capability to detect turbulence at higher altitude which is also important. Therefore, the data recordings are resampled during the training process based on the value of EDR$^{1/3}$ and the altitude of the airplane to make the training data set more evenly distributed in terms of EDR$^{1/3}$ value and aircraft altitude. As an example, Table 4.6 shows the number of data points within specific EDR$^{1/3}$ and altitude ranges of one of the training data sets generated (which contains about 10,000 data samples in total, 5,000 of them are used for actual training).

Table 4.5: Number of Measurements vs Altitude and in situ EDR$^{1/3}$

Alt(m) \ EDR$^{1/3}$	**0-0.1**	**0.1-0.2**	**0.2-0.3**	**0.3-0.4**	**0.4-0.5**	**0.5-0.6**
0-1000	45208	58534	26718	5371	526	20
1000-2000	94548	6659	1073	80	0	0
2000-3000	104784	2457	575	232	0	0
3000-4000	81555	407	154	8	0	0
4000-5000	51943	538	83	97	39	0
5000-6000	39258	259	67	28	0	0
6000-7000	23851	228	13	0	0	0
7000-8000	14584	4	0	0	0	0
8000-9000	4277	63	0	0	0	0
9000-10000	2148	0	0	0	0	0
10000-11000	660	0	0	0	0	0

Table 4.6: Number of Resampled Measurement vs Altitude and EDR$^{1/3}$ for A Training Data Set After Resampling of the Measurement Datasets

Alt(m) \ EDR$^{1/3}$	**0-0.1**	**0.1-0.2**	**0.2-0.3**	**0.3-0.4**	**0.4-0.5**	**0.5-0.6**
0-1000	303	208	208	370	370	370
1000-2000	303	208	208	370	0	0
2000-3000	303	208	208	370	0	0
3000-4000	303	208	208	370	0	0
4000-5000	303	208	208	370	39	0
5000-6000	303	208	208	370	0	0
6000-7000	303	208	208	0	0	0
7000-8000	303	208	0	0	0	0
8000-9000	303	208	0	0	0	0
9000-10000	303	0	0	0	0	0
10000-11000	303	0	0	0	0	0

4.5.3 Training of Artificial Neural Networks (ANN)

ANN is one of the most widely used MI/AI algorithms [84]. In this study, training data is randomly selected from the total amount of radar measurements. In this study, a simple three-layer ANN is used. 10,000 of the 560,000 measured data points (which contains more than two hundred flight cases in total, where each flight has 3,000 to 5,000 data points) are used for training dataset. This training dataset is divided as follows: 50% is used for actual training, 25% as internal validation set and 25% as internal testing set. The actual training set is the data set used for training the ANN classifiers. The validation set is the set used to test the performance of ANN at each training iteration to avoid over-training issue. The testing set is used for performance evaluation after the training process been terminated. These three data sets are randomly divided and independent with each other with no overlapping among each other. The reason for choosing the division of 50-25-25 is that the over training (over fitting) issue could be effectively mitigated, although the accuracy will be inevitably sacrificed. However, the reduction in accuracy is compensated by applying the method described in Section 4.5.4. Ten ANNs are trained during the training process, each of them has different initial weighting values in the network nodes and each is trained with different subsets from the training data set. Although these trained ANNs have slightly different performance, they can be combined during classification process in the next step to reduce the false alarm rate for Category 3 classification.

4.5.4 Classification and Decision Logic

The classification algorithm processes the radar measurement data (testing data set) and estimates turbulence severeness levels according to Table 4.4. For turbulence detection (Category 3 and above), the trained classifier requires all the available data be used, including the features which represent the radar measurements at the current time. For turbulence prediction, the data of current radar measurements should not be used to determine the turbulence category. To implement the prediction operation, we set the features which represent the radar measurements of the current VCP (i.e., the radar measurements with 0-minute time shift) as unavailable, which is equivalent to setting the current radar measurements to zero. As such, the classifiers are solely based on radar measurements of previous time without changing the structure of the trained neural networks, which simplifies the implementations.

Figure 4.7 depicts the key elements of the classification algorithm used in this study. First, feature vectors are extracted from test measurement data for a given time and location. Second, the features are sent to all the trained ANNs in the cluster. The outputs from each ANN contains confidence values (i.e., if a category has the largest confidence value, the ANN will classify the turbulence level to that category). The third step is voting among all the 10 ANN outputs to obtain "global" classification results. A vote means that one ANN in the cluster classifies the radar measurement features at a specific time and location as belonging to one specific EDR$^{1/3}$ category.

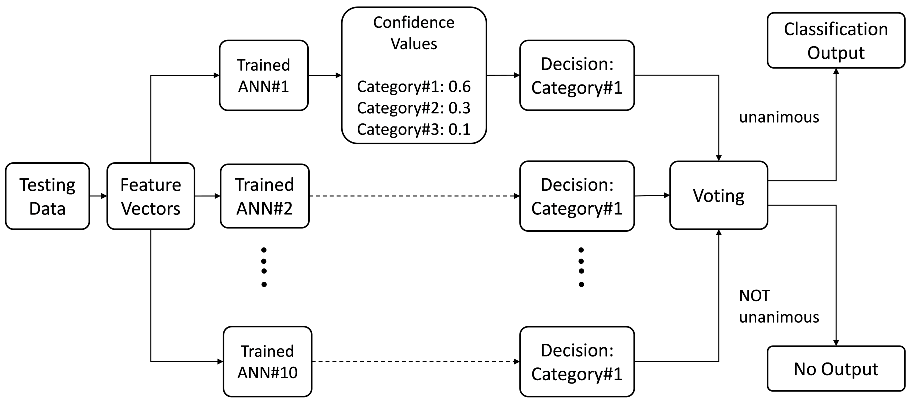


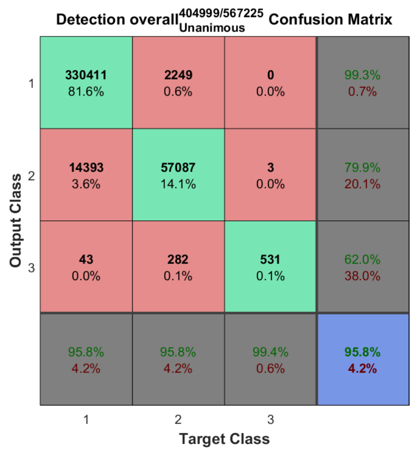
Figure 4.7: Decision and classification algorithm for turbulence detection.

The performance of the classification algorithm is measured by comparing its output to the onboard truth data measurements and the resultant confusion matrix [85], [86]. The performance is mainly manifested by the false alarm rate of turbulence Category 3 (moderate & above), since this category is most important regarding to aviation safety and passenger experience, and it is more prone to get false alarms since the training data sets only have limited cases for this category, as mentioned in Section 4.5.2. In Section 4.6, the detection and prediction results and performance analysis for two cases are presented, they are the flight CX134 (containing 4,488 data points) and flight CX383 (containing 3,860 data points). These two cases represent more than 200 flight cases contained in the entire radar measurement data collection.

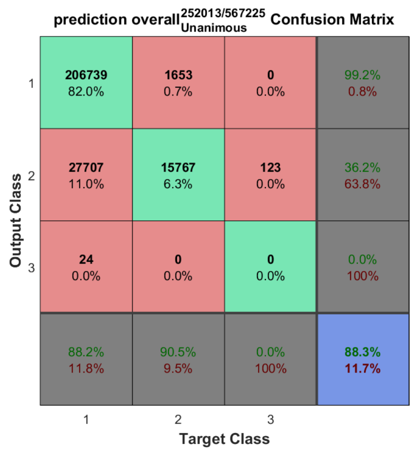
4.6 Summary of Test Results

4.6.1 Summary of All Flight Cases

The overall classification results for all the 200 flights (or 567,225 data sample cases) are collected. Majority of the testing data in the collected datasets produces unanimous voting results (i.e., all ten ANNs in the cluster agree on classifying the data to the same EDR$^{1/3}$ category). Among all the cases, there are 29% of cases in detection and 44% of the cases in prediction that do not have unanimous voting results (i.e., different ANNs in clusters vote for different turbulence categories). A simple majority voting rule was tested first, and the average results show the distinct difference between the cases having unanimous votes and the cases not having unanimous votes. For cases that do not have unanimous votes, the results are more complicated. The overall correct detection (or classification rate) is close to 70% and the overall false positive rate is near 50%. The main reason for these results is the false classification of the Category 1 cases into Categories 2 and 3, which is a result of the training data distribution. More sophisticated algorithms beyond simple majority vote are needed to improve the performance of these cases, such as using ANNs with more layers, and better "weighting" of the confidence levels based on actual vote numbers and prior knowledge. For most of the cases that have unanimous votes, however, the detection performance is good, and the results for these cases are summarized as confusion matrices in Figure 4.8. The fraction number (e.g., 404,999/567,225) in the title of each matrix of Figure 4.8 indicates the number of cases which receive unanimous votes versus the total number of test cases. Since our current goal is using the simple ANN and minimizing computational load, we only record the classification results for the cases that have unanimous votes in this study.



(a) Detection



(b) Prediction

Figure 4.8: Classification performance results for all the collected flight data cases with unanimous classification votes.

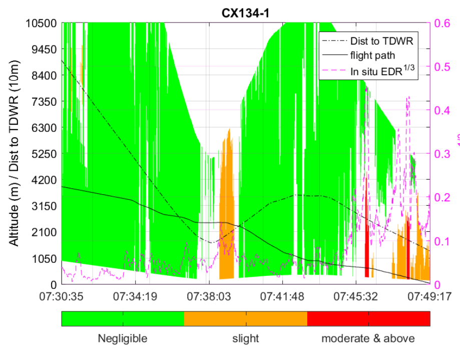
It is more useful to focus on the individual flight data and confusion matrices associated with particular flights, as it is a better way to connect the turbulence classification results with the flight trajectories. Again, we only plot the classification results with unanimous votes, and do not provide classification outputs for cases with less than unanimous voting results. As shown in the flight examples in the following sections, those cases with less than unanimous votes are simply shown as "no output" in the plots.

4.6.2 Case Analysis for Flight CX 134

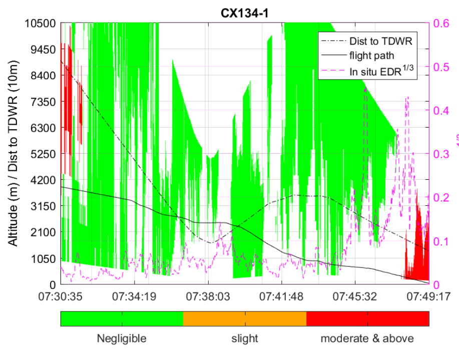
For the test results for the case of CX 134, whose radar and in-flight measurements data are presented in Figure 4.6, the turbulence detections and predictions of vertical profiles along with the flight path are presented in Figure 4.9. In Figure 4.9, the black dash-dot line represents the distances between the CX 134 aircraft and the TDWR, the black solid line represents the altitudes of the CX 134 aircraft and the magenta dash line represents the onboard measurements of EDR$^{1/3}$ values. The green, orange and red color filled spaces are detections/predictions made by the trained classifier of negligible, slight and moderate & above turbulence respectively (i.e., categories 1, 2, and 3; Table 4.4), plotted above and below the flight path of the CX 134 aircraft. Note there are blank (white areas) representing no radar data available because if the aircraft is close to TDWR, available data is limited to lower altitudes for given elevation scan angle. From the detection result of Figure 4.9a, it is observed that the ANN classifier successfully detects turbulences with good accuracy after the time mark 07:45:32, which was at the final stage of landing.

Because the aircraft was so close to the ground for this case, getting accurate moderate and severe turbulence detections is helpful for safe landing.

Although there are apparently some false alarms near time mark 07:38:03 and 07:30:35, the aircraft was at high altitudes for these cases, therefore the accuracy of detection for these cases is not as critical as it is closer to the ground. As for the prediction result in Figure 4.9b, the performance is not as good as the detection results, since less information (due to lacking current radar measurements) was used by the classifier. However, the prediction still indicates that turbulence will occur at the final stage of the landing phase between time mark 07:45:32 and 07:49:17, although these predictions over-estimate the intensity of turbulence.



(a) Detection

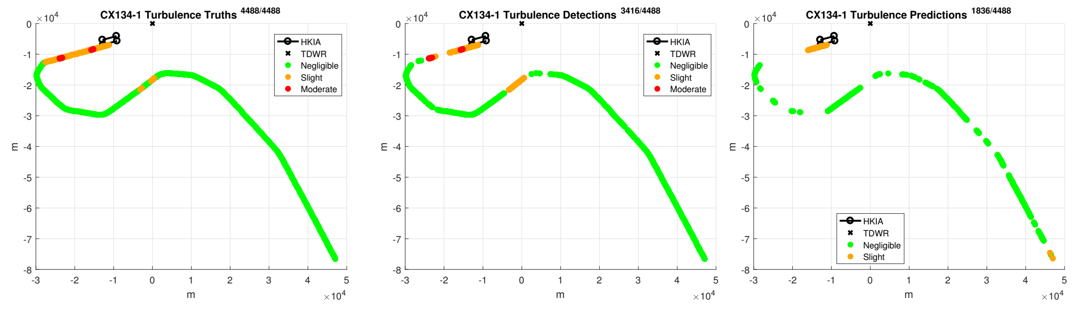


(b) Prediction

Figure 4.9: Detection with 0min data and prediction without 0min data for CX 134.

To have better visualization of the turbulence detection performance along flight tracks, the turbulence measurements, detections and predictions for flight CX 134 along its path are presented in Figure 4.10. The on-board measured turbulence categories are converted from onboard EDR measurements following the rule described in Table 4.4. In Figure 4.10, the black cross represents the location of TDWR, black circles and solid lines represent the north and south runways of HKIA, and green, orange and red dots represent the negligible, slight and moderate & above turbulences (i.e., Categories 1,2 3) measured, detected or predicted, respectively. The detection and prediction results are the outputs from the classifier trained in this study. From the flight track shown in Figure 4.10a, it can be observed that the flight CX 134 was at the final stage of landing when it encountered intense turbulence. The detection results in Figure 4.10b agree reasonably well with the truths shown in Figure 4.10a despite that there are only 3,416 out of 4,488 detection results receiving unanimous votes from the neural network cluster trained in the classifier. Furthermore, there are small amount of the missing data points, which are the detection results that do not receive unanimous votes, and they are associated with the landing stage of the flight. For those points, the category of the turbulence could be inferred from the detections made earlier or later, so they do not have significant effect on the detection performance.

For the prediction results demonstrated in Figure 4.10c, available prediction outputs are sparse compared with detections shown in Figure 4.10b. Only 1,836 out of 4,488 prediction results received unanimous votes. The reason for it is that there is less information (i.e., missing current radar measurements) provided to the classifier for the predictor mode. Therefore, it is more difficult for the classifier to give a result with unanimous votes especially in the severe weather situation as the case tested on, which is during the event of Typhoon Mujigae. However, the prediction results still provide useful information during majority of the flight as well as the level of turbulence being expected near the runways, which is useful because degradation of aircraft performance due to turbulence is more important during landing and taking off.



(a) Truths (b) Detection (c) Prediction

Figure 4.10: Turbulence along with the path of CX 134.

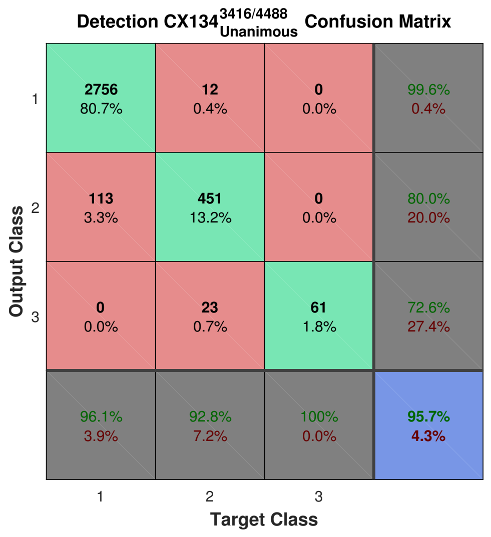
The confusion matrix of detection and prediction testing results for this case are shown in Figure 4.11 using *"plotconfusion"* function in MATLAB. The green and red colored boxes indicate that the cases being categorized correctly or falsely. The green and red percentage numbers in the last row or column of Figure 4.11 represent the percentage of correctly and falsely categorized cases of all cases in respective output or truth categories. The "overall accuracy" at bottom-right corner is computed by taking the ratio of all correctly classified cases (sum the numbers in diagonal boxes) over the total number of test cases. For example, the 95.7% (green) and 4.3% (red) indicate that among the categorized cases the accuracy of detection is 95.7% while the false rate is 4.3%.

Again, Categories 1, 2 and 3 correspond to negligible, slight and moderate & stronger turbulence, respectively. In these matrices, the number entries in every row are the number of cases that have been categorized by the trained MI classifier and every column represents the number of cases belonging to "true" categories as indicated by the onboard recorded EDR$^{1/3}$ measurements. For example, the number 12 in the first row and second column in the matrix means that there are 12 cases being falsely categorized as Category 1, but should belong to Category 2 according to the truth data. The percentage under the number, 0.4% in this example, represents the percentage of all cases (i.e., 3,416 for Category 2), which are falsely categorized to Category 1 by the MI classifier.

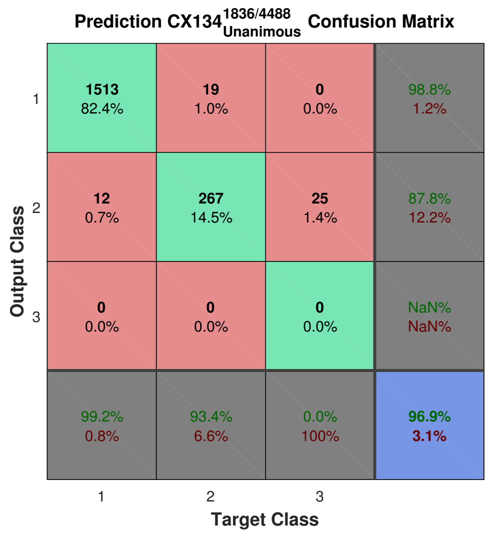
For the error statics associated with each category, the 72.6% (green) and 27.4% (red) in the 3rd row and 4th column of Figure 4.11a indicate that 72.6% of cases being categorized as Category 3 are indeed Category 3 as determined by the recorded onboard EDR value, and 27.4% of the cases being categorized as Category 3 cases are falsely categorized as Category 3. Therefore, the red percentage numbers in gray boxes of the 4th column represent the false alarm rates or false positive rates of respective categories. As another example, the 92.8% (green) and 7.2% (red) in the 4th row and 2nd column of Figure 4.11a indicate that there are 92.8% of Category 2 cases being successfully categorized as Category 2 and 7.2% of Category 2 cases being falsely categorized as other categories. Therefore, the red percentage numbers in gray boxes of 4th row represent the false negative rates for this category.

It can be observed from Figure 4.11a that for the detection task regarding the case of CX 134, the classifier performed well. The overall accuracy is 95.7%. The false negative rate of Category 3 in this test is 0%, which means all the cases which should be categorized as Category 3 are not missed by the classifier. And the false positive rate or false alarm rate of Category 3 in this test is only 27.4%, which means only about 1 in 4 cases which categorized as Category 3 is not actual Category 3. For the aviation safety, the accuracy of detecting high intensity turbulence (i.e., Category 3) is more crucial than that of detecting low intensity turbulence, and reasonable levels of false-alarm are usually acceptable. Moreover, as demonstrated in Figure 4.11a, all the cases that are falsely categorized as Category 3 belong to Category 2 and none of them belongs to Category 1. Since Categories 2 and 3 are generally not quite distinguishable from each other, the false positive of Category 2 being 3 is more acceptable than that of Category 1 being 3. As for prediction task evaluation demonstrated in Figure 10 4.11b, worse performance is expected compared to the detection task. The reason for this is that the current radar measurements are not available for the prediction task. Therefore, less information is utilized by the classifier, which leads to potentially less accurate prediction results. However, turbulence associated with Category 3 is the most important hazard for aviation safety, and although none of the MI classified cases belonging to Category 3 is predicted correctly, all those cases are predicted as Category 2 (instead of Category 1). This might not be unacceptable because a certain level of turbulence is expected for the Category 2 to generate some alerts. Furthermore, the overall accuracy of prediction is surprisingly better than the detection task, with 96.9% cases being predicted correctly, even though fewer cases received unanimous votes (i.e., 1,836 out of 4,488), compared to the detection task, which has 3,416 unanimous votes out of 4,488 total cases.

In Figure 4.11, the reason there is a 27% false positive rate is because there are very small number of Cat 3 cases in the overall dataset, there are only 61 cases for Cat 3, but there are 486 Cat 2 cases according to onboard EDR$^{1/3}$ measurement. Therefore, even only 0.7% (23) of the Cat 2 cases were misclassified to Cat 3, it appears to be a big portion in overall Cat 3. Therefore, this number is “not very fair” for Cat 3 classification performance, and should not be interpreted as traditional “false alarm” rate.



(a) Detection



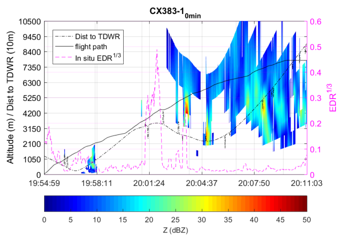
(b) Prediction

Figure 4.11: Confusion matrix of unanimous output of CX 134.

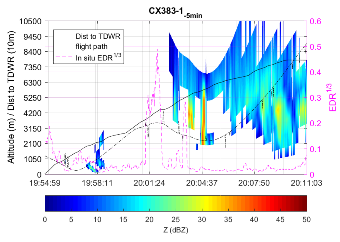
4.6.3 Case Analysis for Flight CX 383

Another representative case selected as an example is the flight CX 383, whose related radar product measurements are presented in Figure 4.12. It is evident that there are a lot of areas in which radar measurements are not available. The reason for this is that there is a mountain (not shown in the figures) between the aircraft path (Figure 4.14) and the TDWR. Therefore, radar measurements at some low altitudes are not available. However, as shown in the following test results, thanks to the robustness of the MI algorithms, the trained classifier still could make reasonable detections/predictions along flight paths without a full suite of radar measurements.

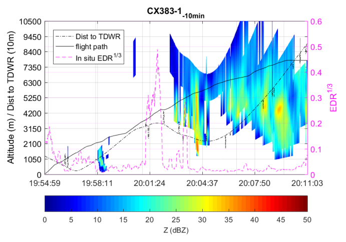
Figure 4.13 shows the detection and prediction results of the case of flight CX 383. The representation of colors and line types are the same as in Figure



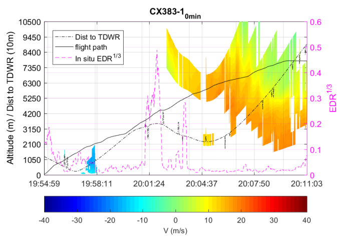
(a) Z, 0 min



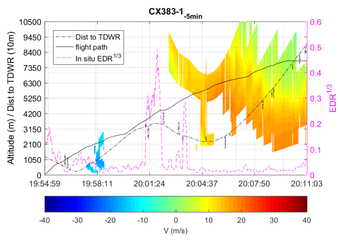
(b) Z, -5 min



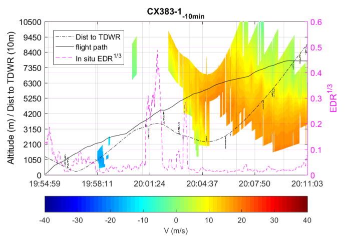
(c) Z, -10 min



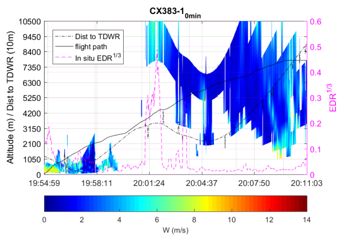
(d) V, 0 min



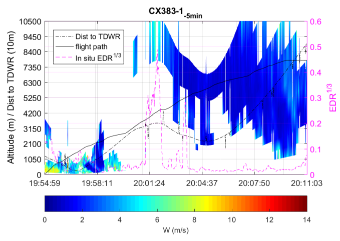
(e) V, -5 min



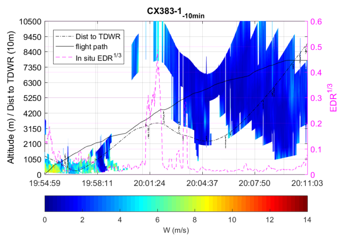
(f) V, -10 min



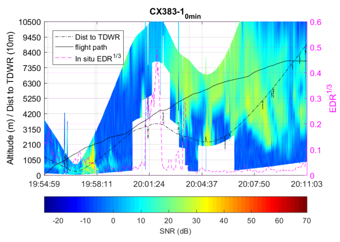
(g) W, 0 min



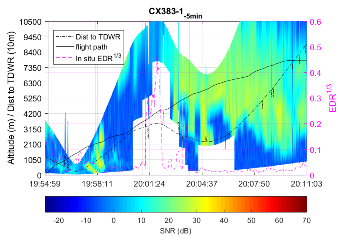
(h) W, -5 min



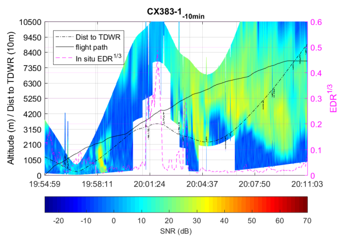
(i) W, -10 min



(j) SNR, 0 min



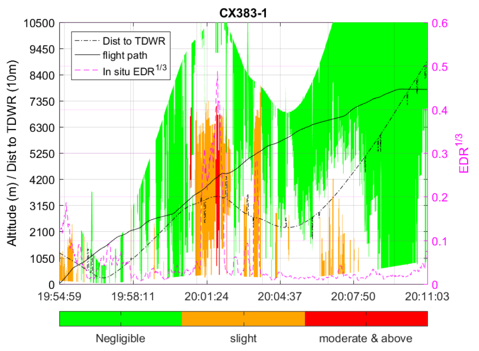
(k) SNR, -5 min



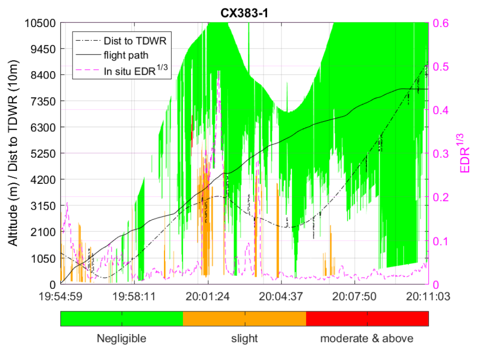
(l) SNR, -10 min

Figure 4.12: Vertical profiles of radar variables along with the flight path of CX 383 at 0, -5 and -10 minutes before the flight time.

4.9. From the detection results shown in Figure 4.13a, it is observed that the detections match the EDR measurements well, especially around time 20:01:24, 19:54:59 and 20:04:37. For the turbulence detected around 20:01:24, in Figure 4.13a, it is observed there are very few radar measurements available



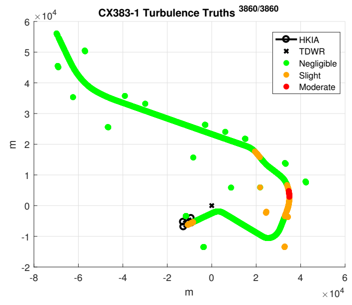
(a) Detection



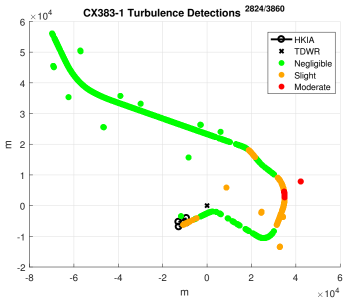
(b) Prediction

Figure 4.13: Detection with 0min data and prediction without 0min data for CX 383.

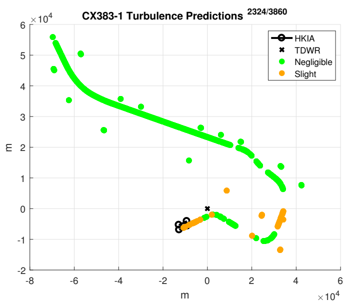
around the aircraft at that time and altitude. However, the trained classifier still manages to give the accurate result even under those conditions. Also, similar performances are achieved at 19:54:59, the taking off stage, and between 20:01:24 and 20:04:37 at mid-altitudes.



(a) Truths



(b) Detection

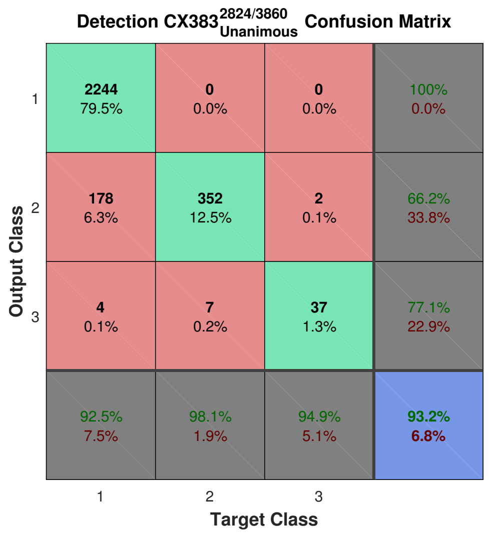


(c) Prediction

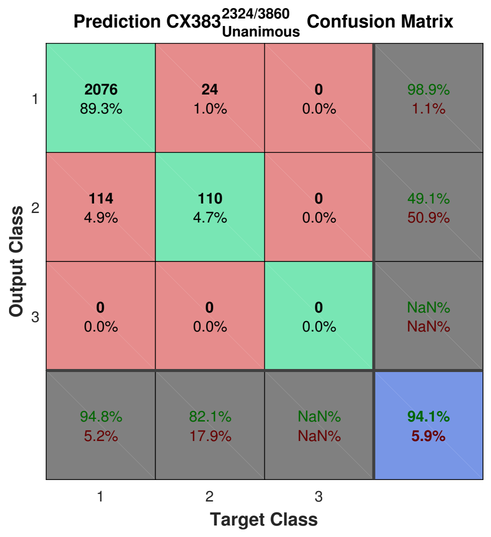
Figure 4.14: Turbulence along with the path of CX 383.

Figure 4.14 demonstrates the truth, detections and predictions of turbulence categories along with the path of flight CX 383. The latter two sets of categories (detections and predictions) are generated by the trained classifier. As shown in Figure 4.14, the aircraft was taking off from the south runway of HKIA towards northeast. After taking off, the aircraft made a right turn near the TDWR and flew over the Victoria Harbor and the city center of Hong Kong and then made a left turn to fly around mountains. Therefore, when the aircraft was flying around the mountains, the mountains were between the aircraft and the TDWR. Hence, the radar signal was heavily blocked at around 20:01:24 by the ground. Again, although very few radar measurements were available around those areas, turbulence was reasonablywell detected along the flight path while the aircraft was experiencing some slight and moderate turbulence at that time. Interestingly, there are also good amount turbulence detections "around" the flight path in this case. Comparing the detection results in Figure 4.14b with the truths in Figure 4.14a, it can be again observed that the classifier performed well along most portion of the flight paths, although it still has a tendency to overestimate the EDR$^{1/3}$ levels. This tendency may be observed near the runway at early stage of takeoff and around coordinate (40000, 0) in the middle altitude in Figure 4.14b. In both areas, the classifier tends to falsely categorize the category from 1-negligible to 2-slight. However, when the turbulence was strong enough to be categorized as 3-moderate & above, the classifier correctly made the detections.

As for the prediction results demonstrated in Figure 4.14c, it performed well around the runway when the aircraft was at early stage of takeoff, but it fails to make a prediction when the turbulence should be categorized as 3-moderate & above when the aircraft was in the middle altitude levels. Also, compared with the detection results shown in Figure 4.14b, fewer outputs are given by the classifier since the trained networks could give unanimous votes for fewer inputs cases



(a) Detection



(b) Prediction

Figure 4.15: Confusion matrix of unanimous output of CX 383.

The confusion matrices of detection and prediction for flight CX 383 are presented in Figure 4.15. The overall accuracies for detection and prediction for the flight CX 383 are 93.2% and 94.1%, respectively. The accuracy of prediction looks higher than that of detection, but the classifier makes less overall outputs in prediction task than it is in detection task, especially when the true turbulence category is 3. From the detection results demonstrated in Figure 4.15a, it could be observed that the true positive and false negative rates for Category 3-moderate & above are 77.1% and 5.1%. As discussed in the previous flight case (CX 134), these two rates are most critical for safety and passenger experience. Also, there is no false categorization from Category 3 to Category 1. Therefore, the classifier correctly gives an alert for every moderate & above turbulence situation, although this alert might not come with correct strength values of the expected turbulence. As for the prediction task results shown in Figure 4.15b, although the overall accuracy of prediction seems adequate, the classifier cannot make any prediction about Category 3.

This is because the neural networks of the classifier cannot classify any case to Category 3 with unanimous votes. Therefore, the 3rd row and 3rd column of Figure 4.15b are all zeros and only 2,324 out of 3,860 data points were given a classification output compared to the 2,824 of 3,860 in the detection task.

4.7 Summary and Conclusions

An machine intelligence (MI)-based algorithm is developed for diagnosing turbulence using Doppler weather radar and providing aviation safety support. The key elements include preprocessing, data association/correlation, radar feature extraction, training data generation, training algorithms, and evaluation methods. ANN is used as the specific MI algorithm and TDWR radar at HKO is used as the specific radar data source, but the techniques may be extended in the future to other weather radars and other MI algorithms. Two flight cases are used for testing the classification performance to detect and predict turbulence.

Performance for turbulence detection, classification and prediction are evaluated for the current available measurement data. Limited performance is expected for the turbulence prediction function, because it is difficult to depend only on "data modeling" to describe the evolution of turbulence and the radar signatures, thus more in-depth physical modeling is required for this capability. Nevertheless, the test results of all the available data show promises. Also, as the amount of data increases, more accurate "data models" will be available for the training algorithms, and the training algorithm can be further improved by better handling of the "non-unanimous" cases.

Since many of the system configuration and parameter selections in this work are based on empirical “trial and error” approach, the future work would naturally require a more systematic optimization approach, which may eventually reduce the time and effort to train the classifiers. The classification algorithm can also extend beyond the unanimous voting requirements and use “confidence levels” to further improve accuracy and robustness. The “confidence levels” can be calculated by averaging the values of classification at the output nodes (confidence values) among all the networks (unanimous scenario) or among the networks with consensus outputs (consensus scenario).

Chapter 5

GPGPU-Based Implementation of Radar Processing Chain

5.1 Basic Signal Processing Chain for Surveillance Radar

In this section, the GPU implementations of the advanced algorithms, such as adaptive pulse compression and Doppler processing, and basic MI algorithms are discussed. The performance benchmarks are obtained for illustration. Although the final goal is to achieve a real-time intelligent processing for weather hazard detections, this chapter only intends to provide basic examples and results for initial efforts.

5.1.1 Data Cube-Based Processing Chain

The generic signal processing chain for a pulsed-Doppler radar surveillance mode includes the following modules in sequence order: pre-processing, digital beam-forming, pulse compression, Doppler spectrum estimate and post-processing, as shown in Figure 5.1. Each stage involves intensive matrix manipulations, which are highly capable of being parallelized. Take, for instance, the beam-forming, which has been widely applied in many operational radar systems, is basically multiplication of the input data with a beam-forming table pre-stored in the memory. Because the input data set is multi-dimensional, where three-dimensional is most common, it is usually referred to as a "data cube". The computation along each dimension is usually independent, and thus can be implemented in parallel as well.

The modern radar system tends to include more advanced signal processing algorithms than the basic processing chain shown in Figure 5.1. For example, adaptive pulse compression (APC) [12], [87]–[91] is an important new algorithm to reduce the sidelobe level from traditional matched filtering. Adaptive beamforming and calibration [92]–[94] is another example of such kinds of algorithms. The existing tools and libraries in CUDA may not be sufficient for implementing these sophisticated algorithms. For such cases, customized kernel development and optimization may be necessary. The following sections discuss some of the solutions developed in this study.

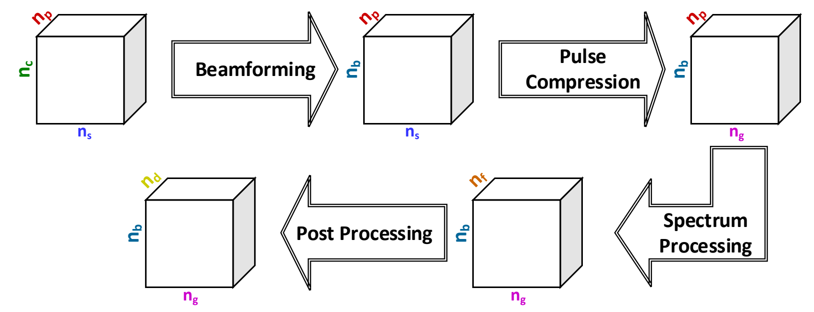


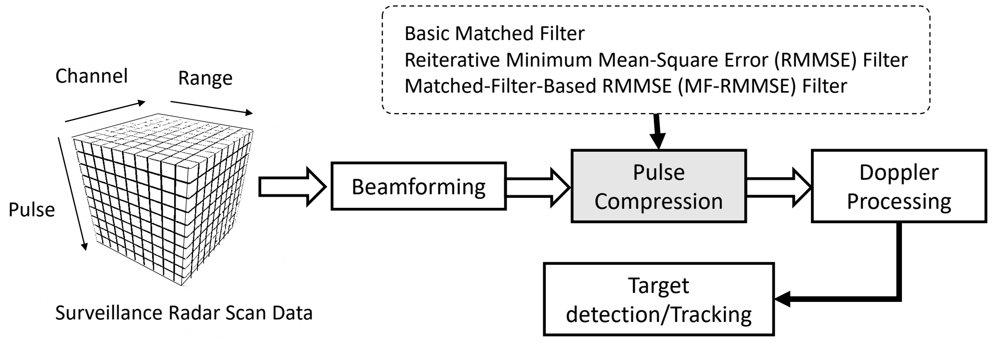
Figure 5.1: The basic "data cube" processing chain of pulsed-Doppler radar. In this figure, n_p is number of pulses per Coherent Processing Interval (CPI), n_c is number of digital channels, and n_g is number of range gates.

5.1.2 CUDA Implementation of The Signal Processing Chain: Basic Concept

For the basic processing chain as shown in Figure 5.1, although we can customize our own "Kernels" for such matrix implementation on GPUs, it is recommended to use a high-level CUDA library such as cuBLAS, which is an accelerated Basic Linear Algebra Subprograms (BLAS) for CUDA-enabled GPUs, as much as we can. Not only does it reduce development time, but also, more importantly, such libraries are highly optimized by professionals for maximum efficiency. Similarly, the cuFFT library can be used for pulse compression. It is worth noting that the size of a data cube at each stage is usually far larger than the number of cores available in GPUs, which indicates that each node of the radar signal processing chain, which is shown in Figure 5.1, may fully utilize all of the computational power of a GPU, potentially maximizing the computational efficiency on each GPU. Therefore, we can pipeline multiple GPUs, where only one node in the chain is implemented on each single GPU, to further reduce processing time.

For general APC, there are scenarios where a large number of ranges and Doppler cells are handled especially in 2D scanning, which have potentials for parallelization. Traditionally, the APC algorithms are specific combinations of various matrix manipulations and are implemented in CPU-based processor architectures [12], [87]–[91]. For large scale APC, the nature of CPUs makes the computation less efficient. In contrast, the original graphic rendering tasks of GPUs require highly parallel architecture, which leads to better computing capability of large data. It is possible and desirable to modify existing APC algorithms to enhance parallelization, to implement such algorithms into embedded GPU platforms and to achieve the acceleration given various types of radar applications.

5.2 Algorithms



Basic Matched Filter
Reiterative Minimum Mean-Square Error (RMMSE) Filter
Matched-Filter-Based RMMSE (MF-RMMSE) Filter
Channel
Range
Pulse
Surveillance Radar Scan Data
Beamforming
Pulse Compression
Doppler Processing
Target detection/Tracking

Figure 5.2: The basic processing chain of pulsed-Doppler radar.

This study focuses on three algorithms in the “pulse compression” category shown in Figure 5.2. They are basic matched filter, reiterative minimum mean-square error (RMMSE) filter and Matched-Filer-Based RMMSE (MF-RMMSE). These algorithms are key to the adaptive pulse compression (APC) algorithm family and important to the resolutions and sidelobe levels of radar range processing.

5.2.1 Matched Filter

The benchmark test procedure for basic matched filtering is as follows: first, simulated ground truth, waveform, and returned signal are generated. Here, LKP3 phase-coded waveform is used as the simulated surveillance radar waveform. Second, zero padding is applied to the waveform as well as the returned signal, to reach the nearest length $l = 2^a 3^b 5^c 7^d$, where a, b, c and d are integers, in order to achieve the optimal FFT performance [95]. Third, a matched filter is applied to the signals as described in equations (5.1) - (5.5), with different implementations of FFT/inverse FFT (iFFT) and multiplication. Since the first two steps are common to all the implementations of the matched filter, time latencies are only measured from the third step.

$$w_r(n) = w(-n) \tag{5.1}$$

$$W(N) = FFT[w_r(n)] \tag{5.2}$$

$$S(N) = FFT[s(n)] \tag{5.3}$$

$$M(N) = W(N) \times S(N) \tag{5.4}$$

$$m(n) = iFFT[M(N)] \tag{5.5}$$

Where w, s and m represent waveform, returned signal and matched filter output, respectively. Subscript r means reversed version of respective data. Lower case letters represent data in time domain (before Fourier transform) while upper case letters represent data in frequency domain (after Fourier transform). $FFT[\cdot]$ and $iFFT[\cdot]$ represent FFT and iFFT computation. The pseudo-code of matched filter is provided in Algorithm 1.

Algorithm 1: Matched Filter

```
1  function MF ($s, wf$)
   Input  : Two vectors of complex numbers $s$ and $wf$.
   // $s$ and $wf$ represent the returned signal and waveform
      respectively
   Output: One vector of complex numbers $mf$.
   // $mf$ represents the match filter output
2  if length(s) != length(wf) then
3      zero padding the shorter vector to make length(x) == length(wf)
4  end
5  $wf$ = reverse($wf$.conj());              // reverse and conjugate $wf$
6  $s$ = circRightShift($s$,1);          // do a circular right shift on $s$
7  $\bar{wf}$ = fft($wf$)
8  $\bar{s}$ = fft($s$)
9  $\bar{mf}$ =$\bar{s} \times \bar{wf}$;          // the multiplication is element-wise
10 $mf$ = ifft($\bar{mf}$)
11 return $mf$
   // the red part of algorithm could be implemented with
      EIGEN(CPU) or CUDA(GPU)
```

5.2.2 Reiterative Minimum Mean-Square Error (RMMSE)

RMMSE is one of the APC algorithms based on the Minimum Mean-Square Error (MMSE) approach [88]. It performs significantly well in term of recovering the truth data from the measurement. However, this outstanding performance comes with a price, namely high computation load. The extremely high computation load limits the application of this algorithm, making it improbable to be applied on the massive size of data from various high definition and/or rapid updating observation tasks. The pseudo-code of RMMSE is provided in Algorithm 2.

```
Algorithm 2: RMMSE
1  function RMMSE ($s, wf, num_{iter}$)
   Input  : Two vectors of complex numbers $s$ and $wf$.
   // $s$ and $wf$ represent the returned signal and waveform
      respectively
   // $num_{iter}$ represents the number of iterations in RMMSE
   Output: One vector of complex numbers.
2  Do zero-padding at the head of $s$;                // $\hat{s}$ = [0 ...0 $s$]
3  Initialize noise matrix $R$;    // $R$ is a diagonal matrix with the
     values of diagonal elements equal to noise power
4  initialize vector $\rho$;                          // $\rho$ is a vector of ones
5  for $idx_{iter}$ = 1 to $num_{iter}$ do
6      $num_{cell} = (2 \times num_{iter} - idx_{iter} - 1) \times (\text{length}(wf) - 1) + \text{length}(s)$
7      for $idx_{cell}$ = 1 to $num_{cell}$ do
8          Initialize matrix $C$
9          for $idx_{in\_cell}$ = 1 to $num_{in\_cell}$ do
10             Update matrix $C$ with zero padded shifted $wf$
11         end
12         $CR = C + R$
13         $\hat{w} = CR \backslash wf$;              // solving linear equations
14         $\hat{w} = \hat{w} \times \rho[idx_{cell} + \text{length}(wf) - 1]$
15         $output_{temp} = \hat{w}^{\intercal} \hat{s}[subvector]$
           // sub vector starts at
              $idx_{cell} + (idx_{iter} - 1) * (\text{length}(wf) - 1)$ with length of
              $wf$
16     end
17     $\rho = |output_{temp}|^2$;                    // element-wise operation
18 end
19 return $output_{temp}$
   // the red part of algorithm could be implemented with
      EIGEN(CPU) or CUDA(GPU)
```

5.2.3 Matched Filter Based RMMSE (MF-RMMSE)

MF-RMMSE is a modified version of RMMSE based on MF output, which is proposed and derived in [12]. It successfully reduces the computational load by processing MF outputs. The computational complexities of matched filter, RMMSE and MF-RMMSE are listed in Table 5.1. The pseudo-code of MF-

Table 5.1: Computational Complexity per Range Cell for Different Algorithms

Algorithms	**Computation Cost**	**Comments**
Matched filter	$O(N)$	where N is the length of waveform
RMMSE	$O(N^3)$	
MF-RMMSE	$O(KN + K^3)$	where K is the length of filter used, and $K << N$

RMMSE is provided in Algorithm 3.

5.3 Graphic Processing Unit (GPU) Testbed

In this section, the hardware and software environments of the GPU Testbed are described.

5.3.1 Hardware

The GPUs used in this study are TITAN Z and TITAN Xp, whose specifications are described in Table 5.2. In this study, only one of the two GK110 GPU cores of TITAN Z is used. Therefore, the TITAN Z used in this study will have the effective computation resource equivalent to TITAN BLACK, whose specification is also listed in Table 5.2. As a comparison, AMD FX 8150 is used as the reference CPU for this study and the CPU implementations utilize FFTW [96] and EIGEN [97] as the counterpart libraries of CUDA.

Algorithm 3: MF-RMMSE

```
1  function MF_RMMSE ($s, mf, num_{iter}$)
   Input : Two vectors of complex numbers $s$ and $mf$.
   // $s$ and $wf$ represent the returned signal and matched
      filter output respectively
   // $num_{iter}$ represents the number of iterations in MF-RMMSE
   Output: One vector of complex numbers.
2  $\rho$ = getMatrixRho();
3  $U$ = getMatrixU($s$);
4  $G$ = getMatrixG($s$);
5  $Sn$ = getMatrixSn($s$);
6  for $idx_{iter}$ = 1 to $num_{iter}$ do
7      for $idx_l$ = 1 to $length(mf)$ do
8          $G_2$ = updateMatrixG($S$,$G$,$lim_{window}$)
           // $lim_{window}$ is the size of processing window
9          $D$ = getMatrixD($\rho, Sn, U$)
10         $\hat{w} = \rho \cdot D \backslash G_2$;            // solving linear equation
11         $output_{temp}[idx_l] = \hat{w}^\intercal mf$;     // vector multiplication
12     end
13     $\rho = |output_{temp}|^2$;                    // element-wise operation
14 end
15 return $output_{temp}$
   // the red part of algorithm could be implemented with
      EIGEN(CPU) or CUDA(GPU)
   // definitions about functions of $getMatrixs$ are referred
      to the original article about MF-RMMSE[12]
```

5.3.2 Software

For the software environment configuration, Windows 10 is used as the operating system, NVIDIA driver version 382.05 and CUDA toolkit 8.0 (including cuFFT and cuBLAS) are used as GPU computing support, MSVC++ 14.0 (Visual Studio 2015) and NVCC 8.0 are used for compiling and linking C++ and CUDA C code, Eigen 3.3.3 is used as fundamental linear algebra library for CPU-based counterpart, and FFTW 3.3.5 is used as CPU-based FFT back-end libraries for comparison.

Table 5.2: The GPUs Used in the Current Studies

Model	**TITAN Z**	**TITAN BLACK**	**TITAN Xp**
Code Name	2×GK110	GK110	GP102
Die Size(mm^2)	2×561	561	471
Fab(nm)	28	28	16
Compute Capability	3.5	3.5	6.1
Microarchitecture	Kepler	Kepler	Pascal
CUDA Core Config[1]	2×2880:240:48	2880:240:48	3840:240:96
Memory Size (MB)	2×6144	6144	12288
Memory Bandwidth[2]	2×336	336	547.7
GFLOPS[3]	8121.6	5120.6	12150
Release Date	03-2014	02-2014	04-2017
TDP(Watts)	375	250	250

[1]Unified Shader Processors : Texture Mapping Units : Render Output Units
[2]Gigabyte per second
[3]Single precision

5.4 Implementation and Optimization

Because both radar PPI scan and CUDA architecture have the hierarchy of three levels, it is reasonable to directly mapping those three layers as a guideline to distribute the computation resources. The initial mapping scheme is described in Figure 5.3. However, after further investigations in particular algorithms, more realistic and efficient computation resource distribution schemes are discovered. In this section, the implementation and optimization schemes for particular algorithms will be discussed.

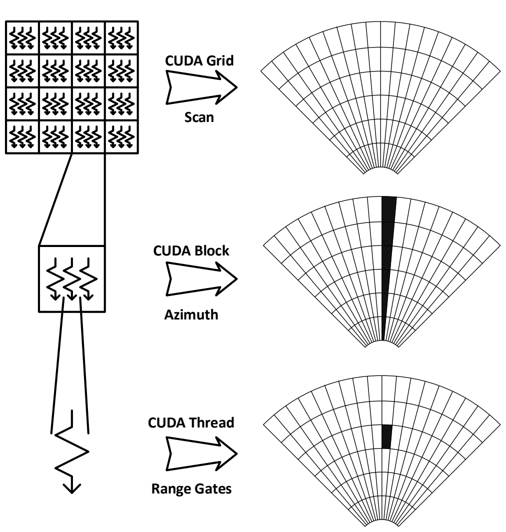


Figure 5.3: The hierarchy of CUDA abstract architecture and related Radar PPI Scan.

5.4.1 Matched filter

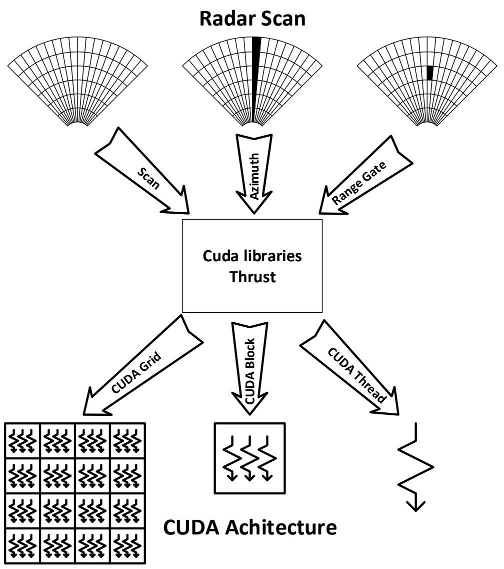


Figure 5.4: The configuration of processing power for matched filter.

The computation of Matched Filter only involves FFT and element-wise multiplication, thus it is feasible to take advantage of CUDA FFT library and

THRUST to perform the computation without building customized CUDA kernels. The GPU version of matched filter algorithm follows the structure shown in Figure 5.4. In this way, the impact of different kernel configurations on performance is less significant. As it will be mentioned later in Section 5.5, different configurations of the kernel, i.e., THREADS PER BLOCK and BLOCKS PER GRID, will have a significant impact on the performance of the GPU version of other APC algorithms.

5.4.2 RMMSE and MF-RMMSE

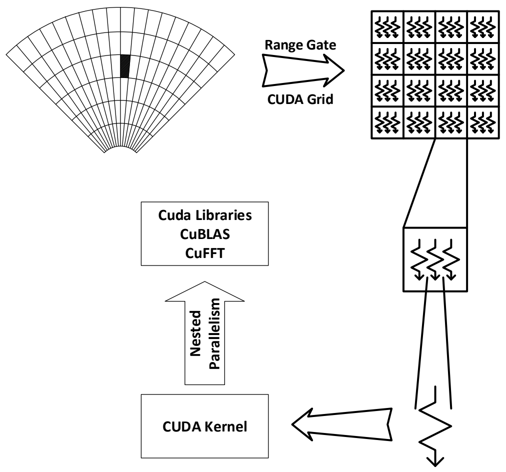


Figure 5.5: The configuration of processing power for RMMSE/MF-RMMSE.

RMMSE and MF-RMMSE are adaptive, iterative and more complicated algorithms, and it would be better to utilize a divide-and-conquer technique, deploy the ability of nested parallelism of CUDA, and explore intrinsic parallelism of such algorithms, while it is applied on a small amount of data for each individual process. In addition, the high level computation scheme of RMMSE and MF-RMMSE is depicted in Figure 5.5. In this scheme, it is inevitable to manually configure the computing resource on the GPU. So the key to reaching the best acceleration performance is to adjust the processing parameters optimally with respect to specific data size. This will be further discussed in Section 5.5.3.

5.4.3 Data Transfer

One major issue of various GPGPU implementations is the bottleneck of data transfer between CPU (or host memory) and GPU (or device memory) [98]. And it would not be a fair performance comparison between CPU and GPU unless the cost of the data transfer has been treated appropriately. Because optimizing data transfer is beyond the scope of this work, the cost of data transfer is excluded from the GPU performance measurements in this dissertation, unless it states otherwise. A discussion on the overhead of data transfer will be given with a real-world application in Section 5.6.

5.4.4 Memory Space Usage

Shared memory is a type of memory in GPU which is faster but also smaller than global memory [95]. Utilizing shared memory on GPU would be beneficial in terms of processing speed in some applications [99], [100]. However, because of the size restriction of shared memory, it is not feasible to utilize such memory in APC algorithms implementations. As described in Sections 5.2.2 and 5.2.3, RMMSE and MF-RMMSE consist several computations between fairly large size of matrices and the smallest object of such algorithms is large matrix which could not be completely restored in the size-restricted shared memory. Therefore, it would not benefit from utilizing shared memory without heavily altering the definitions and structures of these algorithms.

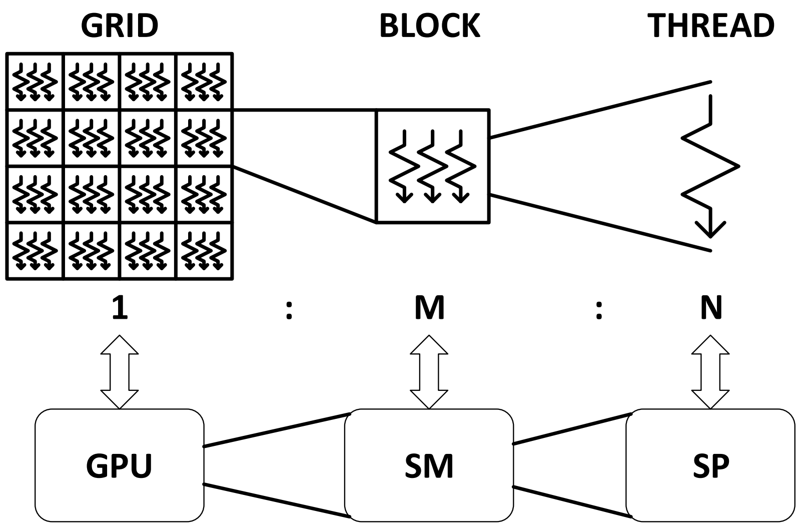


Figure 5.6: The hierarchy of CUDA abstract architecture and related GPU physical architecture.

5.4.5 Statistical Kernel Optimization

One key factor, which has a critical impact on performance, is the configuration of CUDA kernel parameters. As shown in Figure 5.6, there are both three layers of CUDA abstract architecture and GPU physical architecture [98]. The parameters **BLOCKS PER GRID** and **THREADS PER BLOCK** are user definable but the ratio between the GPU, SMs (streaming multiprocessors) and SPs (streaming processors) is fixed for a specific GPU device. In addition, the choice of **BLOCKS PER GRID** and **THREADS PER BLOCK** to get optimal performance depends on the type of algorithm to be implemented and the type of GPU to be used. There is another "layer" between thread and grid existing for CUDA architecture called warp. A warp consists of 32 fixed threads and it is the basic operation group for SM to execute [101]. However, since the warp is not configurable by users, the layer of warp is omitted for seeking the optimal configuration in this research.

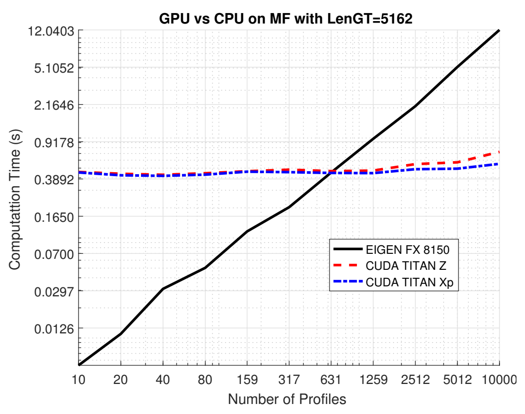
For our research, a statistical sampling and searching method is firstly used to determine the optimal configuration of CUDA kernel parameters for each algorithm implementation scenario (with variations in the size of data), based on testing several combinations of parameters in a given range for each scenario. Time latencies, aka the performance, for each set of parameters chosen on every individual scenario were recorded. Next, we ran the CPU counterpart of the algorithm on respective scenarios and recorded time latencies. Last, we computed the GPU/CPU time latency ratios for each set of parameters on every individual scenario. By running statistically large samples of the parameter combinations, we will have a "performance map" for selecting the optimal kernel configuration. This technique will be illustrated in Figure 5.11 and 5.14 for RMMSE and MF-RMMSE algorithms, respectively.

5.5 Benchmark Results

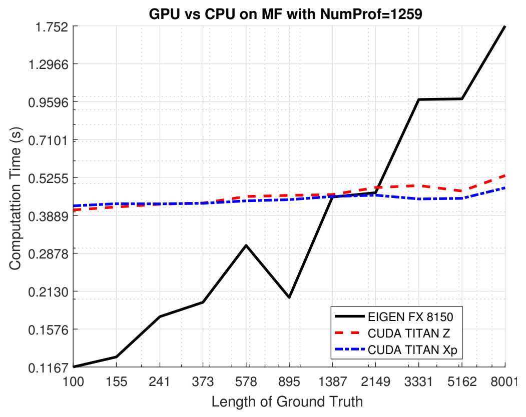
In this section, the APC algorithms implementations described in Section 5.4 will be tested. Benchmark results of various data configurations and CUDA settings for each algorithm will be provided.

5.5.1 Matched Filter

According to the results shown in Figure 5.7, the GPU implementation based on cuFFT is about 5-30 times faster than the CPU implementation based on FFTW, as long as the dataset is sufficiently large. The performance of TITAN Z and TITAN Xp are nearly the same with TITAN Xp holding a slight edge over TITAN Z.



(a) Time latency used vs number of pulses, while length of data is fixed to 5162 sample points.

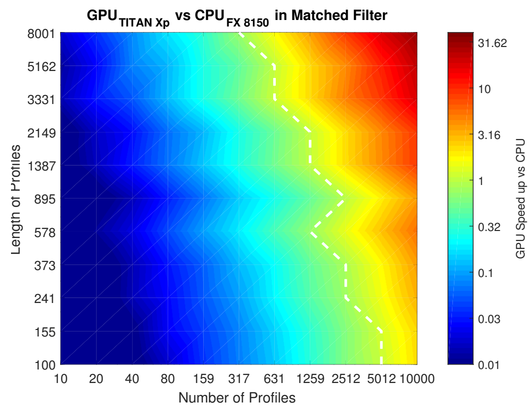


(b) Time latency used vs length of range profile, while number of pulses is fixed to 1259.

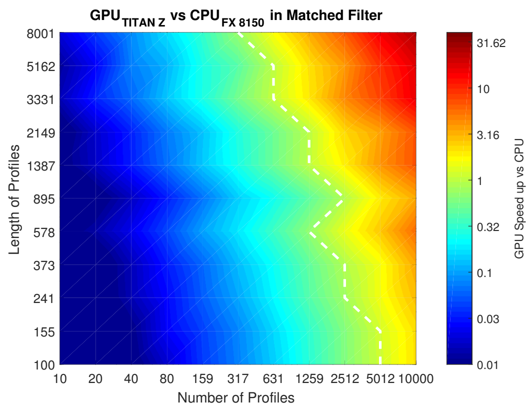
Figure 5.7: The performance of matched filter computing based on various libraries and processor usage.

Figure 5.8 provides another view on the performance comparison of the matched filter. Comparing the performances of TITAN Xp and TITAN Z in Figure 5.8, although TITAN Xp performs better than TITAN Z overall, trends of the amount of acceleration achieved with regard to the number and length of range profiles are the same for both of the GPUs used.

The white dash line in Figure 5.8 represents the equal performance between CPU and GPU platforms. It shows that about 30 times speed-up is achievable in several configurations of source profiles, as it could be seen that the



(a) TITAN Xp

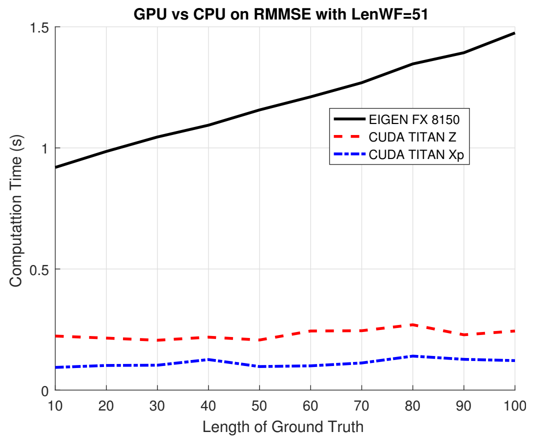


(b) TITAN Z

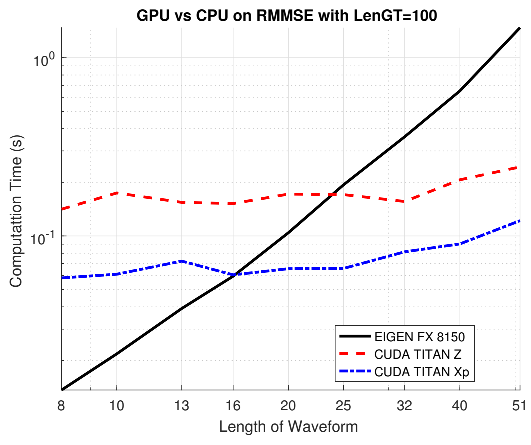
Figure 5.8: The performance comparison of Matched Filter based on GPU and CPU platforms.

acceleration curve is not quite smooth. The main reason for this phenomenon is that cuFFT library is optimized when the the length of processing time-series is $l = 2^a 3^b 5^c 7^d$, where a, b, c and d are integers, and in addition, the smaller the prime factor is, the better the performance would be [95]. Thus, along with the axis of "Number of Profiles", the acceleration performance of the GPU versus CPU monotonically increases. However, along with the axis of "Length of Profiles", the acceleration performance reaches its peak when the length of the profile is close to 2^a. It is obvious that the larger the size of data, the larger acceleration ratio can be obtained. However, the size of on-board memory limits further acceleration. When the memory size limit is reached, multiple data transfers are required to circumvent such limit, and the data transferring process could be the bottleneck of such implementation. However, the cost of data transfer can be compensated by utilizing another CUDA stream and a more sophisticated computing scheme by overlapping the computing and data transfer stream. Further acceleration would be expected and it will be exploited in future experiments.

5.5.2 RMMSE



(a) Length of Waveform = 51

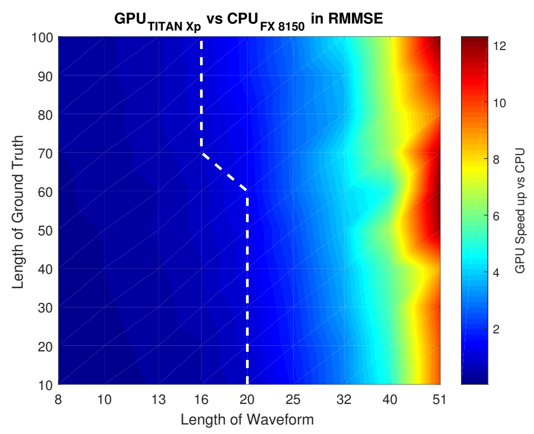


(b) Length of Ground Truth = 100

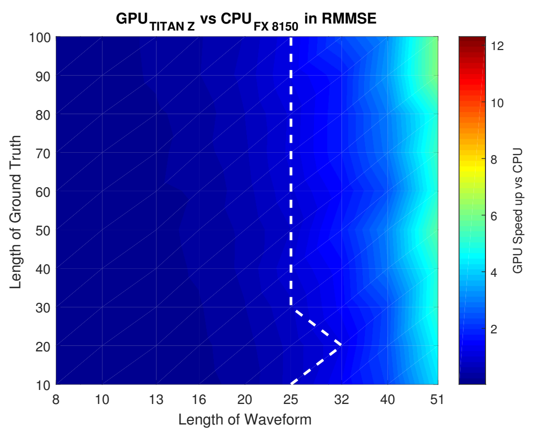
Figure 5.9: The performance comparison of RMMSE based on GPU and CPU platforms.

The performance comparison of RMMSE implementations based on GPU and CPU platforms is demonstrated in Figure 5.9. The results indicate that the length of the waveform (which is a transmitted pulse) has a larger impact in processing time compared with the length of ground truth (which is the impulse response of range profile), which is implied in the description of its computation complexity listed in Table 5.1, and the GPU-based platform performs better when the length of either parameters mentioned above is larger, as it can be seen that about 10 times acceleration is expected when the data size is sufficiently large.

Figure 5.10 provides another view on the performance comparison of RMMSE implementations. The white dash line in Figure 5.10 represents the equal performance between the CPU and GPU platforms. It shows that about 12 times speed-up is achievable in several configurations of range profiles.

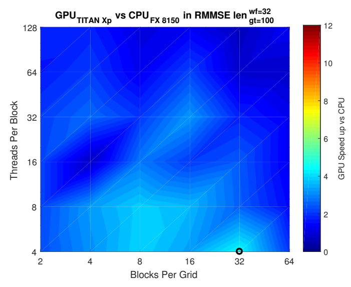


(a) TITAN Xp

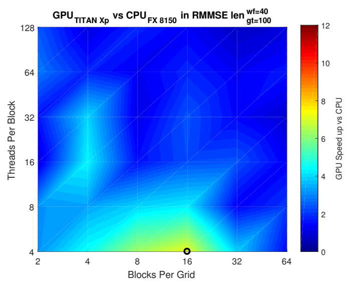


(b) TITAN Z

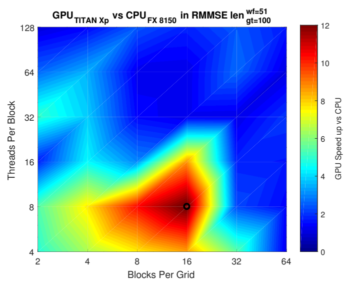
Figure 5.10: The performance comparison of RMMSE based on GPU and CPU platforms.



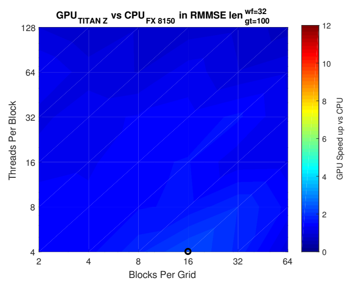
(a) TITAN Xp, Len_wf = 32.



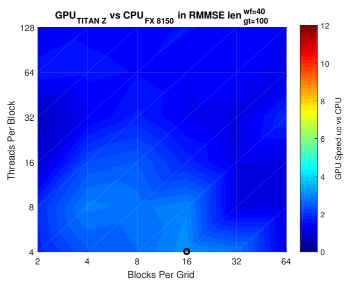
(b) TITAN Xp, Len_wf = 40.



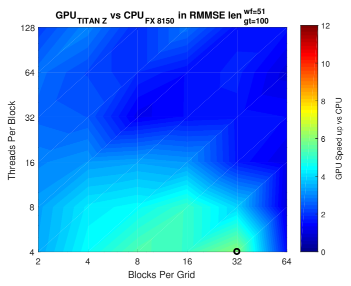
(c) TITAN Xp, Len_wf = 51.



(d) TITAN Z, Len_wf = 32.



(e) TITAN Z, Len_wf = 40.



(f) TITAN Z, Len_wf = 51.

Figure 5.11: The performance comparison of RMMSE between GPU and CPU implementations with various CUDA configurations and length of waveforms when the length of ground truth is fixed to 100 sample points. Lengths of waveforms are from 32 to 51. The black circle represents the configuration when maximum speed-up is achieved under current data format.

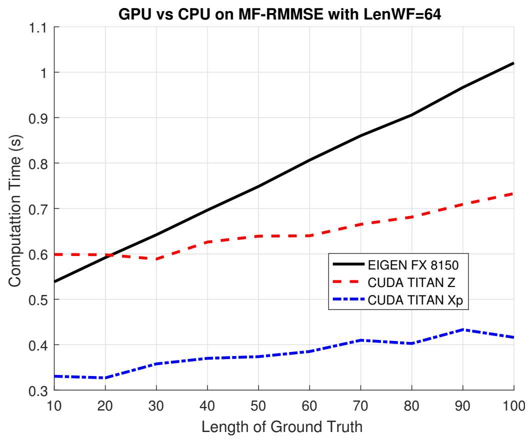
As mentioned in Section 5.4.1 and 5.4.5, the configuration of CUDA computing resources, i.e., THREADS PER BLOCK and BLOCKS PER GRID, has major impacts on the performance of the GPU version of algorithms. Plus, for more sophisticated algorithms, such as RMMSE and MF-RMMSE, utilizing a customized kernel is necessary to perform matrix-wise and element-wise manipulation as introduced in Section 5.2.2 and 5.2.3. Thus, properly distributing the GPU computing resources according to the problem data size is the key to achieving the optimum performance of the GPU implementations of these algorithms.

Figure 5.11 shows the impacts of different configurations of GPU computing resources on the acceleration performance of RMMSE algorithm implementations. The black circles in each figure represent the largest achievable speed-up of the GPU compared with the CPU counterpart in specific lengths of waveforms, while the length of ground truth is fixed at 100 sample points for all the cases, since it has less impact on the processing time as discussed earlier.

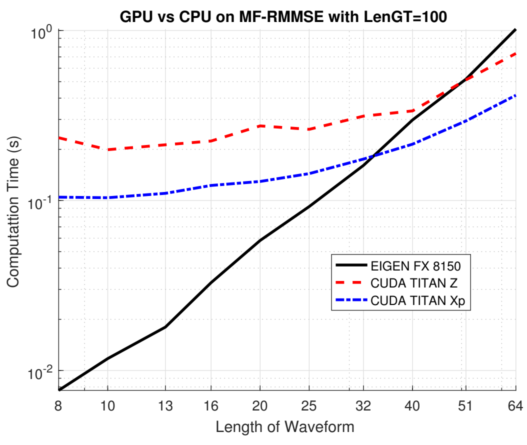
It can be seen in Figure 5.11 that there exists an optimized computing resource configuration for each specific data size. For the application of RMMSE investigated in this study, we can conclude that the best configuration of CUDA are 4 or 8 for THREADS PER BLOCK and 16 or 32 for BLOCKS PER GRID for different lengths of waveforms.

As for the performance of different generations of GPU platforms, comparing results in Figure 5.9, 5.10 and 5.11, TITAN Xp (Pascal) consistently outperforms TITAN Z (Kepler) in this test. In general, TITAN Xp runs twice faster than TITAN Z regardless of the size of dataset.

5.5.3 MF-RMMSE



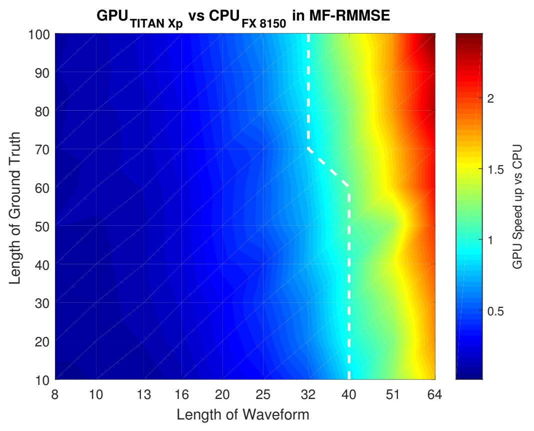
(a) Length of Waveform = 64



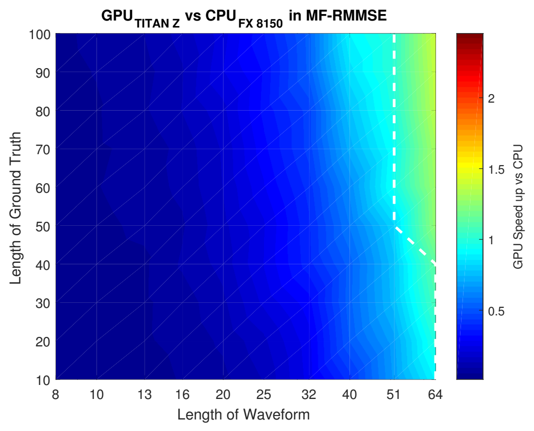
(b) Length of Ground Truth = 100

Figure 5.12: The performance comparison of MF-RMMSE based on GPU and CPU platforms.

The performance comparison of MF-RMMSE implementations based on GPU and CPU platforms is demonstrated in Figure 5.12. As can be seen in this Figure, the acceleration is not as significant as the implementation of RMMSE, which is shown in Figure 5.9. As described in Section 5.2.3, one reason for this phenomenon is that the way MF-RMMSE is designed is different from that of RMMSE [12]. It utilizes a processing window that is much smaller than the length of waveform, and it effectively reduces the impact of the length of the waveform on processing time. In addition, as mentioned in Section 5.5.2 and implied in Table 5.1, the advantage on the performance of a GPU over a CPU is more significant when the length of the waveform is large. Thus, GPU implementations may be less beneficial for MF-RMMSE than original RMMSE. However, similar to the observation of the performance result of RMMSE, GPU implementation of MF-RMMSE algorithm performs better when the length of the waveform grows longer.



(a) TITAN Xp



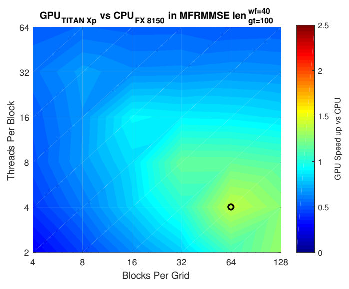
(b) TITAN Z

Figure 5.13: The performance comparison of MF-RMMSE based on GPU and CPU platforms.

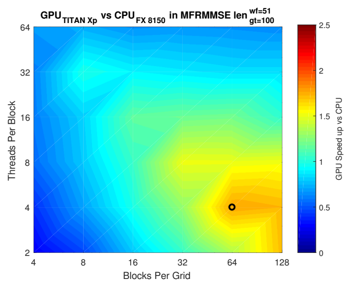
Figure 5.13 provides another view on the performance comparison of MF-RMMSE. The white dash line in Figure 5.13 represents the equal performance between the CPU and GPU implementations. It shows that about 2.5 times speed-up is achievable in several configurations of source profiles. Although MF-RMMSE algorithm utilizes matched filter outputs [12], the computation cost of a matched filter is negligible compared to MF-RMMSE. Thus, compared with Figure 5.8, the acceleration curve in Figure 5.13 is much smoother.

Similar to Figure 5.11, Figure 5.14 summaries the impacts of the GPU kernel resources on the acceleration performance of MF-RMMSE implementations. The black circles in each figure represent the largest achievable speed-up of the GPU compared with the CPU counterpart having specific lengths of waveforms, while the length of ground truth is fixed at 100 sample points for all the cases, since it has less impact on the processing time as discussed earlier.

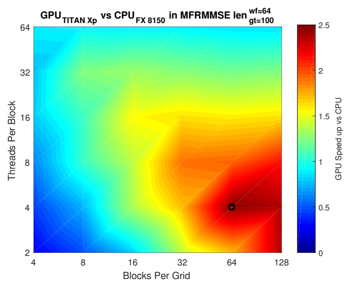
It can be seen in Figure 5.14 that an optimized computing resource con-



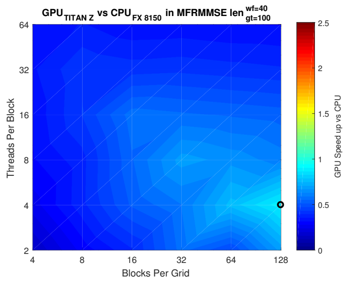
(a) TITAN Xp, Len_wf = 40.



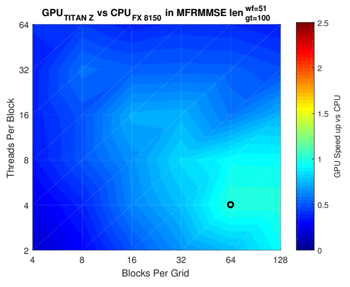
(b) TITAN Xp, Len_wf = 51.



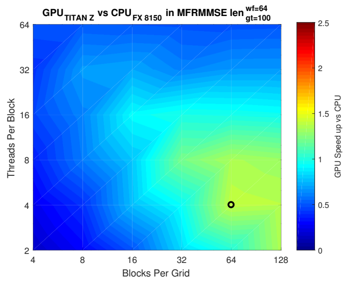
(c) TITAN Xp, Len_wf = 64.



(d) TITAN Z, Len_wf = 40.



(e) TITAN Z, Len_wf = 51.



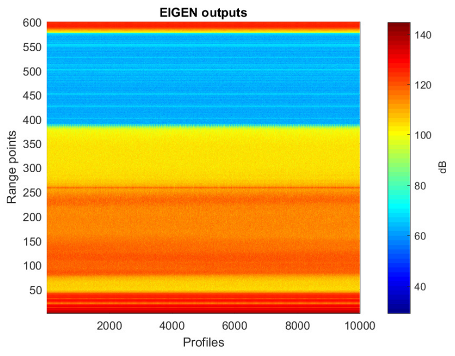
(f) TITAN Z, Len_wf = 64.

Figure 5.14: The performance comparison of MF-RMMSE between GPU and CPU implementations with various CUDA configurations and length of waveforms when length of ground truth is fixed to 100 sample points. Lengths of waveforms are from 40 to 64. The black circle represents the configuration when maximum speed-up is achieved under current data format.

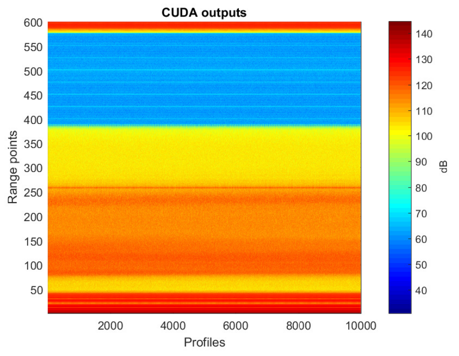
figuration may be achieved for each specific data size. For the application of MF-RMMSE investigated in this study, we can conclude that the best configuration of CUDA is 4 for THREADS PER BLOCK and 64 or 128 for BLOCKS PER GRID for different lengths of waveforms. The trend is similar to what could be found in the testing results of RMMSE in Section 5.5.2.

As for the performance of different generations of GPU platforms, comparing results in Figure 5.12, 5.13 and 5.14, TITAN Xp (Pascal) consistently outperforms TITAN Z (Kepler) in this test. For general comparison, algorithms implemented on TITAN Xp execute twice faster than TITAN Z regardless of the sizes of the datasets.

5.6 Application to Real/Measured Data

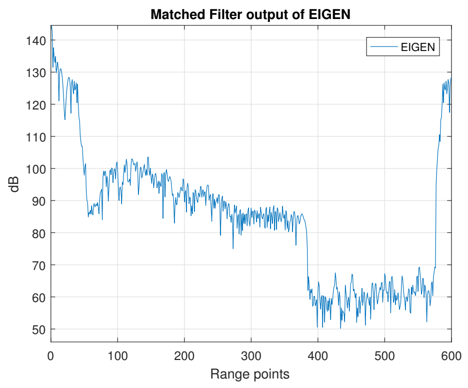


(a) Matched Filter output of EIGEN

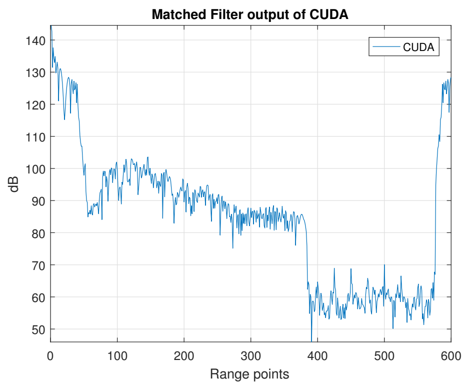


(b) Matched Filter output of CUDA

Figure 5.15: Output results of pulse compression implementation using NASA's HIWRAP radar data



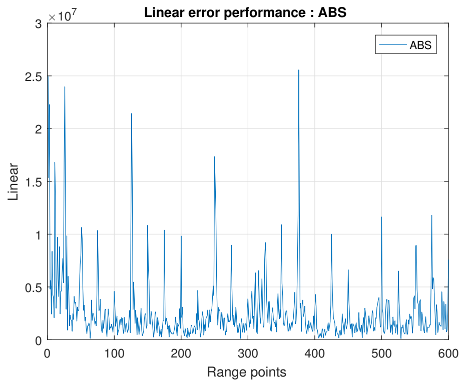
(a) Averaged Matched Filter output of EIGEN



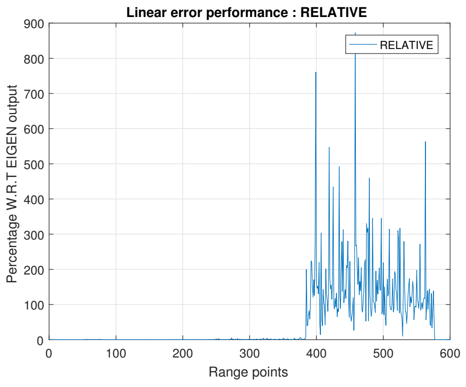
(b) Averaged Matched Filter output of CUDA

Figure 5.16: Averaged output results of pulse compression implementation using NASA's HIWRAP radar data

Matched filter is implemented on NASA's HIWRAP radar measurement data [12] with both the CPU-based method (EIGEN) and GPU-based method (CUDA). From the matched filter outputs shown in Figure 5.15, the two results are almost identical except for those positions (range points from 400 to 570)

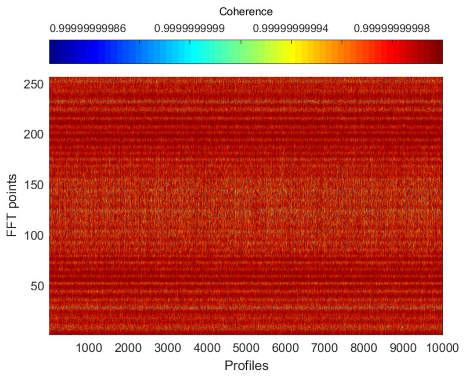


(a) Absolute linear error performance

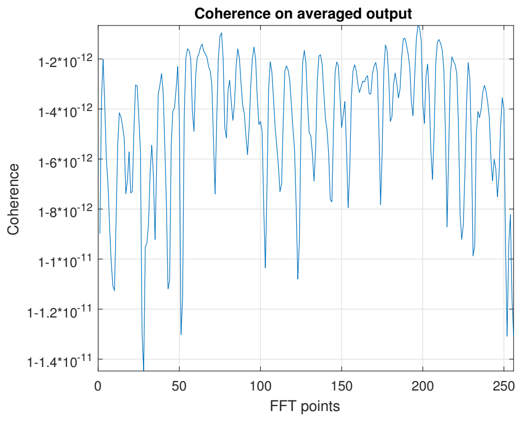


(b) Relative linear error performance

Figure 5.17: Error results of pulse compression implementation using NASA's HIWRAP radar data



(a) Magnitude-square coherence estimate between EIGEN and CUDA outputs



(b) Magnitude-square coherence estimate between averaged EIGEN and CUDA output

Figure 5.18: Coherence results of pulse compression implementation using NASA's HIWRAP radar data

with low signal power levels. Figure 5.16 is averaged matched filter outputs of EIGEN and CUDA among $10,000$ range profiles, respectively. From these two figures, it is more clear that the only noticeable difference between EIGEN and CUDA results is located within the low signal return region. Figure 5.17 is the absolute and relative errors of CUDA output comparing with EIGEN output. The errors are calculated in linear scale, not dB. Although there is no way to determine whether EIGEN or CUDA output is more "correct", the two outputs are similar to each other, thus they are able to validate each other. It could be observed in Figure 5.17 left that the errors are consistent in the same level. However, Figure 5.17 right shows that the errors are more noticeable when the Signal-to-Noise Ratio (SNR) is lower. Magnitude-square coherence between CUDA and EIGEN outputs is also demonstrated in Figure 5.18. From the coherence results in these figures, it can be concluded that the results from CUDA and EIGEN outputs are identical.

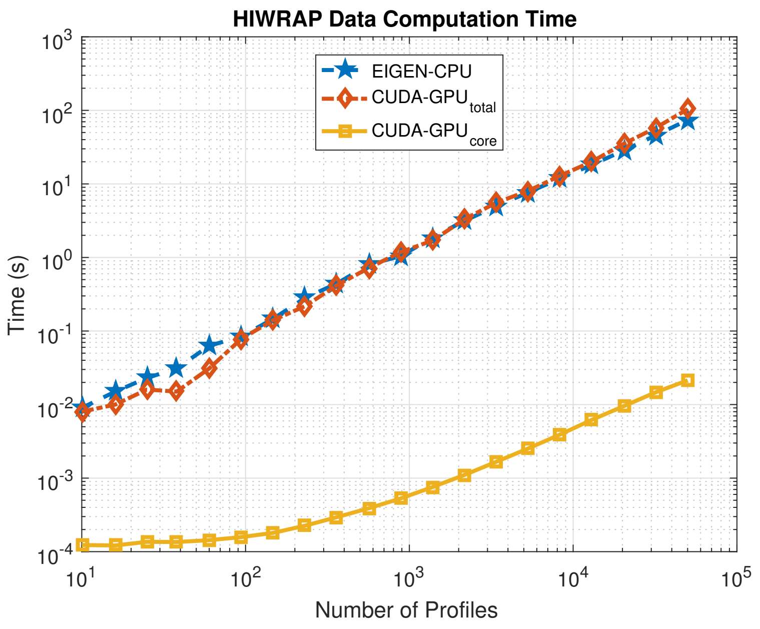


Figure 5.19: Time consumption of CPU and GPU versus number of range profiles handling in HIWRAP data, while GPU time consumption without data transfer is demonstrated separately

For the computation time evaluation, since the matched filter is a relatively simple algorithm, it is expected that data transferring process between CPU and GPU would be the bottleneck for such application as the result demonstrated in Figure 5.19. In this experiment, to fully utilize the parallel computing ability of the GPU, data transfer from host (CPU) memory to device (GPU) memory is done by a burst of all the data included (to be specific, copy all data from “THRUST host vector” to “THRUST device vector” in one command). It could be observed that, taking account of the overhead of data transfer between the CPU and GPU, the time consumption of the GPU-based approach is nearly comparable to the CPU-based approach. However, if the time of the GPU computing part is recorded separately from the data transferring part, the “true” GPU time consumption is far less than the CPU counterpart. To implement this evaluation specifically for the GPU computing part, a timer based on CUDA events is used around the GPU computing part as recommended by an official document from NVIDIA [98]. In this experiment, where only Matched filter is implemented, the time consumption of the CPU-based approach is as much as 3466 times larger than the “true” GPU-based approach computing time consumption. Here, in this section, AMD FX-8150 and NVIDIA TITAN Z are used as CPU and GPU devices, respectively.

5.7 AIR- OU/ARRC for Adaptive Beam-Forming

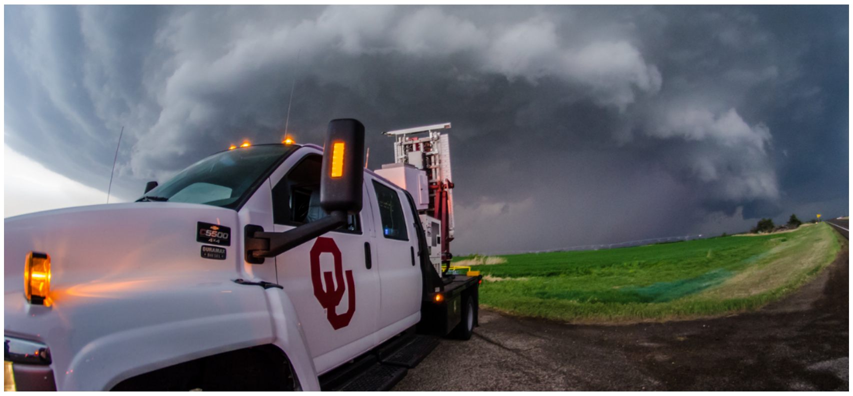


Figure 5.20: Atmospheric Imaging Radar (AIR)

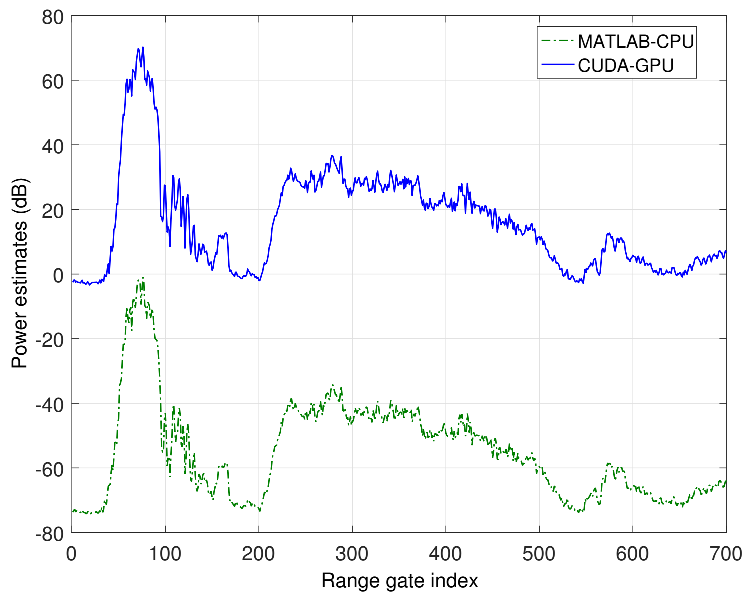


Figure 5.21: Power estimate comparison: the offset is added intentionally to distinguish the two. [102]

The very first step is to compare the data products rigorously to ensure that the GPU reproduces exactly the same results as the MATLAB scripts. This is shown in Figure 5.21. The power estimate is an intermediate data product, which can be used to derive reflectivity in weather radar. The reflectivity range profiles look the same, where as minor differences can be found due to precision differences. MATLAB uses double precision, while we used single precision on the GPU. It is worth noting that most GPUs are capable of double precision computation. However, the number of double precision cores on an SMX is much less than single precision cores in most products of the GeForce product line with exceptions of TITAN Black and TITAN Z [103].

Next, in order to test real-time capability, we tuned up the PRF to the maximum, 3185 Hz in this case, which will result in the highest data rate. The number of rage gates is 785. Because the pedestal rotates at 16°/sec, and

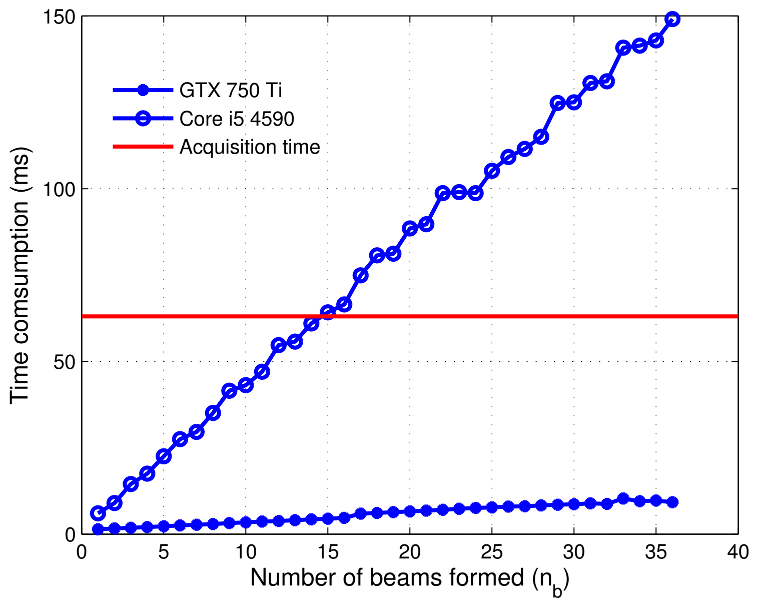


Figure 5.22: Time consumption of the GPU and CPU versus the number of beams in AIR signal processing. The red line marks the data cube acquisition time. Note that only the low-end products are tested out as the GTX-750 Ti has already met the real-time requirement.

the antenna has a horizontal beamwidth of 1°, the data acquisition time for each degree is 1/16 sec, or 63 ms. In this period, approximately 200 pulses need to be processed. Therefore, the total time to process an input data cube of $n_c = 36$, $n_s = 785$ and $n_p = 200$ needs to be less than 63 ms. Figure 5.22 shows the time consumption comparison as the number of formed beams increases. First and foremost, it is obvious that the GPU meets real-time requirements for all cases, while CPU only satisfies this theoretical requirement when the number of formed beams is no greater than 13. In practice, at least all beams without oversampling are needed, i.e., $n_b \geq n_c$. In addition, the estimated processing time needs to be within a safe range to tolerant any jitters. Therefore, a GPU is unequivocally the better choice for real-time processing in this application over a CPU. We also found an interesting fact through the experiment that CPU implementation is susceptible to other applications running at the same time. In a few occasions, even a mouse move event may affect the performance as it is up to the operating system to schedule the priority of each running process. When running on a GPU, the estimated time consumption is more stable as it is more isolated and does not share resources with other processes, which can be considered another advantage of GPUs over CPUs from a different perspective, although this could be circumvented by implementing an application-oriented operating system.

5.8 Artificial Intelligence (AI) Implementation Based on GPGPU Platform

5.8.1 Nvidia Jetson

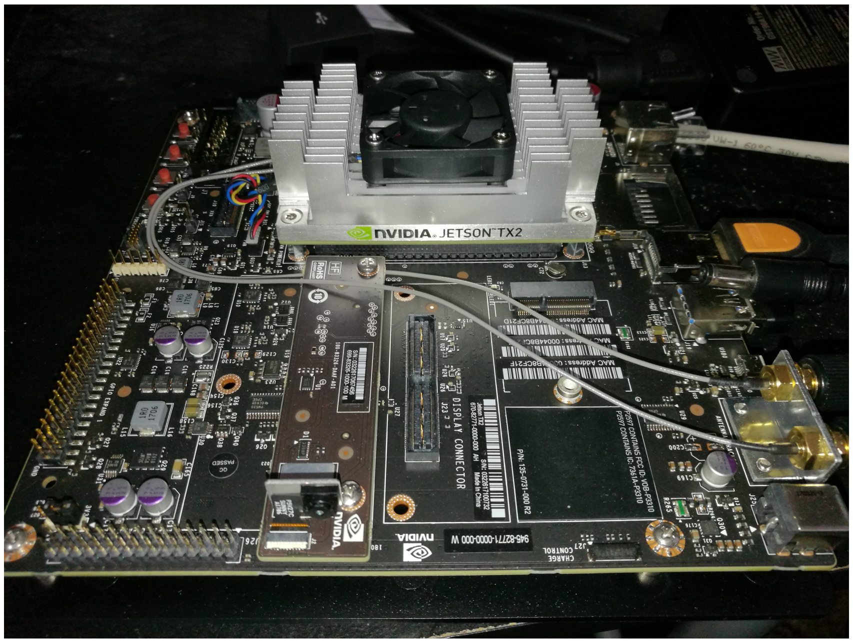


Figure 5.23: the Jetson TX2 development kit.

Nvidia Jetson is a series of embedded GPU computing development boards from Nvidia. Every generation of Jetson board comes with a respective generation Nvidia Tegra processor on board. Till 2017, there have been 3 generations of Jetson board developed by Nvidia and they are listed in Table 5.3 and the Jetson TX2 board is shown in Figure 5.23.

Table 5.3: Generations of Nvidia Jetson Boards

Jetson Model	**TK1**	**TX1**	**TX2**
Tegra Processor	K1	X1	X2
CPU Core	Quad cores ARM Cortex A15 @2.32GHz	Quad cores ARM Cortex A57 @1.73GHz	Quad cores Cortex A57 @2GHz + Dual cores Nvidia Denver2 @2GHz
GPU Core	Kepler GK20a 192 cores @951MHz	Maxwell GM20B 256 cores @998MHz	Pascal GP10B 256 cores @1300 MHz
Ram	2GB 64bits DDR3L @933MHz	4GB 64bits LPDDR4 @1600MHz	8GB 128bits LPDDR4 @1866MHz
Bandwidth	15GB/s	25.6GB/s	58.3GB/s
API Level	3.2	5.3	6.2
TDP	5W	10W	7.5W
Release Date	Q3 2014	Q4 2015	Q1 2017
GFLOPS	365	1024	1500

In addition, various deep learning frameworks are supported via the integration of CUDA cuDNN package, including Caffe, Torch, TensorFlow, etc. This will largely reduce the difficulty of migrating current machine learning algorithms to this embedded system to take advantage of the low power consumption to broaden the scenarios of applications, such as mounting a low-powered embedded machine learning system onto an aircraft to get the real-time on-site capability of turbulence detection.

5.8.2 CuDNN

The NVIDIA CUDA Deep Neural Network Library (cuDNN) is a GPU-accelerated library of primitives for deep neural networks. cuDNN provides highly tuned implementations for standard routines such as forward and backward convolution, pooling, normalization and activation layers [104]. cuDNN provides a flexible, easy-to-use C-language API as well as the support for various widely used deep learning framework, including Caffe2, MATLAB, Microsoft Cognitive Toolkit, TensorFlow, Theano and PyTorth. And according the the document provided by Nvidia [104], by utilizing cuDNN on Nvidia GPU platform, 36% overall acceleration could be achieved by integrating cuDNN into caffe comparing to use caffe alone for a deep learning task.

5.8.3 Caffe

Caffe is an open source deep learning framework originally developed by Yanqing Jia at UC Berkeley under BSD license written in C++ with a python interface [105]. It supports various deep learning architectures focusing on image classification and image segmentation. Also, as an acceleration measure, it supports the integration of cuDNN from Nvidia to utilize GPU computing resources if available. As a more flexible and modular evolution, Caffe2 is developed by Facebook providing easier interfaces and higher performances.

5.8.4 TensorFlow Framework

TensorFlow is a open source software library in the field of machine learning for varieties of perception and language comprehension tasks. It was developed by Google Brain team for the research and production in Google, and was made open source under Apache 2.0 license on Nov 9, 2015. TensorFlow is the second generation of machine learning platform for Google Brain. Although the reference implementation on Nov 9, 2015 was running on single device, TensorFlow is capable of running on multiple CPUs and GPUs with optional CUDA extensions. It could run on 64 bit desktop or server operating systems as well as mobile computing platforms like Android and iOS. The computation of TensorFlow is presented by data flow map with states. The name TensorFlow comes from the operation on multi-dimensional arrays on such neural network-like systems. Those multi-dimensional array is called "Tensor". However, this "Tensor" is not the same as the "Tensor" in mathematics. As for compatibility, TensorFlow provides a Python API, as well as C++, Haskell, Java, Rust APIs and C#, Julia, R and Scala APIs via third party packages.

5.8.5 Benchmark Tests

The benchmarks across PC and embedded system are performed with Caffe framework. As was explained in Section 5.8.3, parallel computing on GPGPU is supported via CUDA cuDNN library. Therefore, the benchmark performances would give us the idea of differences in time between PC and embedded system. The GPU devices for PC and embedded system in this benchmark testing are listed in Table 5.4. As can be seen in Table 5.4, the embedded system (Jetson TX2) only comes with about 1/10 of the computation capability comparing with PC counterpart (TITAN Xp), it consumes much less than 1/10 of power. Furthermore, the embedded system also comes with a ARM CPU core, which will let the system operate independently, while the GPU counterpart on PC will need an additional host machine to operate.

Table 5.4: Comparison between Jetson TX2 and TITAN Xp

	TITAN Xp	**Jetson TX2**
GPU Core	Pascal GP102-450-A1 30 SM - 3480 cores @1405MHz	Pascal GP10B 2 SM - 256 cores @1300 MHz
Ram	12GB 384bits GDDR5X @1425MHz	8GB 128bits LPDDR4 @1866MHz
Bandwidth	547.7GB/s	58.3GB/s
API Level	6.1	6.2
TDP	250W	7.5W
Release Date	Q2 2017	Q1 2017
GFLOPS	10790	1500

The environments of this benchmark are Windows 10, CUDA 8.0 and cuDNN 5.1 for PC and JetPack 3.1 for Jetson TX2 SoC. The benchmark programs are AlexNet [106] and GoogleNet [107]. AlexNet is an 8-layer deep learning network with 5 convolution layers and drop-out features. And GoogleNet is a 22-layer deep learning network with 9 inception modules. These two networks are widely used as the benchmark of performance when they are under training and/or after deployment.

Table 5.5: Benchmarks of Deep Learning Networks on GPUs and CPUs

Time(ms)	**AMD Ryzen 7 1700**	**Jetson TX2 CPU**	**Nvidia TITAN Xp**	**Jetson TX2 GPU**
AlexNet Ave Forward Pass	427.552	6367.5	4.48479	54.8356
AlexNet Ave Backward Pass	289.067	6260.26	5.36093	78.7918
AlexNet Ave Forward-Backward	716.72	12627.9	10.1594	133.781
AlexNet Total 50 Iterations	35836	631394	507.97	6689.05
GoogleNet Ave Forward Pass	1143.88	14124.9	20.7222	139.969
GoogleNet Ave Backward Pass	594.508	13096	19.9822	172.066
GoogleNet Ave Forward-Backward	1738.84	27221.1	41.2649	312.27
GoogleNet Total 50 Iterations	86942	1361055	2063.245	15613.5

The results of benchmarks across different systems are presented in Table 5.5. Jetson TX2 is performed on Max-P mode during benchmark procedure, which will have a TDP of 15W. The benchmarks are performed with command “caffe time”. From Table 5.5, it could be observed that the TITAN Xp performs about 10 times faster than Jetson TX2 GPU while Jetson TX2 GPU outperforms AMD Ryzen 7 1700 CPU about 10 times as well. The performance gap between TITAN Xp and Jetxon TX2 agrees the difference in computation power listed in Table 5.4. Considering the power consumption of Jetson TX2 SoC, this embedded platform shows a great potential as the platform of implementing the turbulence detection algorithm developed in this research, and makes such system available on board in the aircraft. Furthermore, the embedded system is scalable, which will have the ability to achieve a higher performance when needed by building a cluster.

The Neural Network profiled in this section is AlexNet (2012) and GoogleNet (2015). AlexNet consists of 5 conv layers, max-pooling layers, dropout layers, and 3 fully connected layers, which is much more complicated than the neural network developed in Chapter 4. The neural network developed in this research to detect and predict turbulence is a simple 3 fully-connected layers network. The reason for this is that it is mathematically proved that a neural network with three fully connected layers is capable to approximate any functions with arbitrary precision. As for the computation load, the three fully connected layers in AlexNet have 1000-4096 nodes which is more much than the number of nodes in the network developed in this research, i.e., about 184 nodes maximum in all three layers. Therefore, it is reasonable to assume the computation load for AlexNet is at least 1000 times higher (three layers with at least 10 times more nodes ignoring the additional load from other AlexNet layers) than the network developed in this research.

In practice, when a network is well trained and taken into operation, the only operation the network will need to perform is one run of forward pass. As seen in Table 5.5, a forward pass for AlexNet takes about 55ms in average on Jetson TX2 platform. Therefore, the forward passing time for the network developed in this research would be less than 55μs on Jetson TX2, which is way below the threshold to be considered as a “Real-Time” algorithm. Therefore, a single Jetson TX2 embedded system would provide enough computation power to support the approach developed in this research for turbulence detection and prediction task in real-time.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

As for the GPGPU implementation for the radar processing chain, the feasibility of GPGPU-based implmementation of advanced pulse compression algorithms is explored, which is a key element and usually a bottleneck of an end-to-end radar data processing chain. For Adaptive Pulse Compression (APC), GPGPU-based solutions show great potentials of accelerating such resource-demanding algorithms without introducing additional processing error. During the investigation of algorithm execution acceleration of the GPGPU, a relationship is discovered between the computing resource configuration on GPU hardware and acceleration performance compared with CPU-based platforms under different sizes and structures of data being handled. For future reference, an optimal way is proposed to allocate resource on GPUs for better performance based on empirical data. In addition, as a major obstacle of various GPGPU implementations, the overhead of the data transfer between CPU (host) and GPU (device) might be enormous for some demanding applications. One instance of such applications is analyzed, and the result indicates that a more sophisticated and efficient memory management scheme is required to unleash the parallel computing capability of GPU.

For hazard detection and classification challenge, both theoretical and practical aspects are investigated. For the theoretical part, the difficulty remains in the lack of adequate understanding and investigation of the physical model of weather hazards to develop an accurate and quantifiable relation between the radar measurements and natural hazards. For the turbulence detection example, it seems unlikely to gain adequate knowledge of the nature of turbulence to derive an accurate theoretical relation between measurements and turbulence intensity in the near future, unless there were unforeseen revolutionary breakthroughs coming. However, with the rapid development of machine learning technology, to get an adequately accurate estimation of turbulence intensity from measurements without accurate physical models, becomes a possibility. An approximated relation between measurements and turbulence intensity could be derived and developed through the training process of the machine learning approach. Although the training principle is simple and straightforward, the results after training process are inexplainable, i.e., there is no way to explain the reasoning behind the values, a.k.a. network weights, after training. Therefore, the machine learning approach could be considered a "black box" process. For the practical part, the difficulty remains in meeting time and power consumption requirements for certain applications. As for time requirement, a solution would be less useful if the results could not been generated in real time. Similar to the application of radar signal processing chain, the neural network contains a lot of operations capable of being parallelized both in the training and inferencing stages. Therefore, GPGPU offers advantages on accelerated algorithm implementations.

For the theoretical part of solution, as shown in Chapter 3, a machine learning based approach for the problem of turbulence detection is developed and tested. The ANN shows a potential of handling this kind of tasks, which have enormous measured data available but lack of physical models.

For the practical part of solution, GPU is considered as the accelerator to ensure that such solution meets real time requirement. As desctibed in Section 5.8.5, the Jetson TX2-embedded platform shows potential for implementing ANN on CUDA cores. In addition, as for the low power consumption, with a regular TDW of 7.5W, it is well suited for the Space Weight and Power (SWaP) restricted applications, and that will make the turbulence detection develop in this research a viable candidate to be implemented on-board.

6.2 Future Work

For the theories and algorithms, first, a further investigation is needed in those detection results which the ANN could not produce outputs because of the non-unanimous votes. However, the number of vote received and the decimal outputs of networks before being converted to classification could be easily obtained. Therefore, it may be beneficial to incorporate those intermediate outputs into the final detection decision process. Second, a deeper neural network could be utilized with the convolution layers plus the existing connected layers. The task under investigating in this research is similar to image recognition tasks, which is currently successfully tackled by deeper neural network frameworks [106], [107]. Therefore, it is reasonable to expect a performance enhancement after those methods have been explored. Third, the feature selection process as used in this research is a trial and error process with the combination of pre-knowledge of turbulence and limitation of radar measurements. However, features picked from such process may not be optimal, which may lead to the redundancy and over-fitting. In addition, there might be some other unused radar measurements and their derivatives better suited as input features. Those problems might be resolved by an automated feature selection process while and before the network being trained.

For the practical part, the highest priority is to migrate the networks trained in the current and future stages to the GPU-based platform and eventually to the embedded Jetson Platform to meet SWaP requirements. Second, a cluster of multiple embedded systems should be considered as the algorithms become more complicated and time-consuming. A balance between the SWaP and the sophistication of the algorithm developed should be considered. Last but not least, field tests are needed after the design and implementation have been finalized as part of the validations.

References

[1] I. Annex *et al.*, “3, meteorological service for international air navigation”, *International Civil*, 2010.

[2] S.-H. Kim, H.-Y. Chun, and P. W. Chan, “Comparison of turbulence indicators obtained from in situ flight data”, *Journal of Applied Meteorology and Climatology*, no. 2017, 2017.

[3] L. Li, G. M. Heymsfield, P. E. Racette, L. Tian, and E. Zenker, “A 94-ghz cloud radar system on a nasa high-altitude er-2 aircraft”, *Journal of Atmospheric and Oceanic Technology*, vol. 21, no. 9, pp. 1378–1388, 2004.

[4] L. Li, G. Heymsfield, J. Carswell, D. Schaubert, J. Creticos, and M. Vega, “High-altitude imaging wind and rain airborne radar (hiwrap)”, in *Geoscience and Remote Sensing Symposium, 2008. IGARSS 2008. IEEE International*, IEEE, vol. 3, 2008, pp. III–354.

[5] T. Balz and U. Stilla, “Hybrid gpu-based single-and double-bounce sar simulation”, *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, no. 10, pp. 3519–3529, 2009.

[6] W. Chapman, S. Ranka, S. Sahni, M. Schmalz, U. Majumder, L. Moore, and B. Elton, “Parallel processing techniques for the processing of synthetic aperture radar data on gpus”, in *IEEE International Symposium on Signal Processing and Information Technology, Bilbao, Spain*, IEEE, Bilbao, Spain, 2011, pp. 573–580.

[7] M. Lambers, A. Kolb, H. Nies, and M. Kalkuhl, “Gpu-based framework for interactive visualization of sar data”, in *IEEE International Geoscience and Remote Sensing Symposium*, IEEE, Barcelona, Spain, 2007, pp. 4076–4079.

[8] F. Zhang, G. Li, W. Li, W. Hu, and Y. Hu, “Accelerating spaceborne sar imaging using multiple cpu/gpu deep collaborative computing”, *Sensors*, vol. 16, no. 4, p. 494, 2016.

[9] O. Ponce, P. Prats-Iraola, M. Pinheiro, M. Rodriguez-Cassola, R. Scheiber, A. Reigber, and A. Moreira, “Fully polarimetric high-resolution 3-d imaging with circular sar at l-band”, *IEEE Transactions on Geoscience and Remote Sensing*, vol. 52, no. 6, pp. 3074–3090, 2014.

[10] *Introduction to Radar Systems*. Tata McGraw Hill, 2001, ISBN: 9780070445338. [Online]. Available: `https://books.google.com/books?id=zlZom9QkjCkC`.

[11] S. Wang, Z. Li, and Y. Zhang, “Application of optimized filters to two-dimensional sidelobe mitigation in meteorological radar sensing”, *IEEE Geoscience and Remote Sensing Letters*, vol. 9, no. 4, pp. 778–782, 2012.

[12] Z. Li, Y. Zhang, S. Wang, L. Li, and M. Mclinden, “Fast adaptive pulse compression based on matched filter outputs”, *IEEE Transactions on Aerospace and Electronic Systems*, vol. 51, no. 1, pp. 548–564, 2015.

[13] D. Atlas, K. R. Hardy, and K. Naito, “Optimizing the radar detection of clear air turbulence”, *Journal of Applied Meteorology*, vol. 5, no. 4, pp. 450–460, 1966.

[14] D. Atlas and R. Srivastava, “A method for radar turbulence detection”, *IEEE Transactions on Aerospace and Electronic Systems*, no. 1, pp. 179–187, 1971.

[15] A. B. Smith and R. W. Katz, “Us billion-dollar weather and climate disasters: Data sources, trends, accuracy and biases”, *Natural hazards*, vol. 67, no. 2, pp. 387–410, 2013.

[16] O. Wright and F. C. Kelly, *How we invented the airplane*. McKay Philadelphia, 1953.

[17] R. W. Gruenhagen, *Mustang: The Story of the P-51 Fighter*. Arco, 1976.

[18] B. Sweetman, *f-22 raptor*. Zenith Imprint, 1998.

[19] R. Goyer, “Cessna 172: Still relevant?”, *Flying*, vol. 139, 2012.

[20] D. J. Ingells, *747: story of the Boeing super jet.* TA B-Aero, 1970.

[21] Wikipedia, *United airlines flight 826 — wikipedia, the free encyclopedia*, [Online; accessed 20-August-2017], 2017. [Online]. Available: `https://en.wikipedia.org/w/index.php?title=United_Airlines_Flight_826&oldid=793683548`.

[22] ——, *Delta air lines flight 191 — wikipedia, the free encyclopedia*, [Online; accessed 20-August-2017], 2017. [Online]. Available: `https://en.wikipedia.org/w/index.php?title=Delta_Air_Lines_Flight_191&oldid=783551411`.

[23] ——, *Low level windshear alert system — wikipedia, the free encyclopedia*, [Online; accessed 20-August-2017], 2017. [Online]. Available: `https://en.wikipedia.org/w/index.php?title=Low_level_windshear_alert_system&oldid=780676014`.

[24] ——, *Usair flight 1016 — wikipedia, the free encyclopedia*, [Online; accessed 20-August-2017], 2017. [Online]. Available: `https://en.wikipedia.org/w/index.php?title=USAir_Flight_1016&oldid=795392396`.

[25] ——, *Airborne wind shear detection and alert system — wikipedia, the free encyclopedia*, [Online; accessed 20-August-2017], 2016. [Online]. Available: `https://en.wikipedia.org/w/index.php?title=Airborne_wind_shear_detection_and_alert_system&oldid=725472079`.

[26] R. Sharman and T. Lane, *Aviation Turbulence: Processes, Detection, Prediction.* Springer International Publishing, 2016, ISBN: 9783319236308. [Online]. Available: `https://books.google.com/books?id=jv6ODAAAQBAJ`.

[27] H. Lamb, *Hydrodynamics*, ser. Dover Books on Physics. Dover publications, 1945, ISBN: 9780486602561. [Online]. Available: `https://books.google.com/books?id=237xDg7T0RkC`.

[28] M. I. Skolnik, "Introduction to radar", *Radar Handbook*, vol. 2, 1962.

[29] M. G. C. Clark, *Deflating British Radar Myths of World War II.* Pickle Partners Publishing, 2014.

[30] J. D. Kraus, "Heinrich hertz-theorist and experimenter", *IEEE Transactions on Microwave Theory and Techniques*, vol. 36, no. 5, pp. 824–829, 1988.

[31] J. C. Maxwell, "A dynamical theory of the electromagnetic field", *Philosophical transactions of the Royal Society of London*, vol. 155, pp. 459–512, 1865.

[32] L. Brown, "A radar history of world war ii", *Institute of*, 1999.

[33] D. Atlas, "Advances in radar meteorology", *Advances in geophysics*, vol. 10, pp. 317–478, 1964.

[34] R. C. Whiton, P. L. Smith, S. G. Bigler, K. E. Wilk, and A. C. Harbuck, "History of operational use of weather radar by us weather services. part i: The pre-nexrad era", *Weather and Forecasting*, vol. 13, no. 2, pp. 219–243, 1998.

[35] S. D. Perera, Y. Pan, Q. Zhao, Y. R. Zhang, D. Zmic, and R. J. Doviak, "A fully reconfigurable polarimetric phased array testbed: Antenna integration and initial measurements", in *Phased Array Systems & Technology, 2013 IEEE International Symposium on*, IEEE, 2013, pp. 799–806.

[36] S. Perera, Y. Pan, Y. Zhang, X. Yu, D. Zrnic, and R. Doviak, "A fully reconfigurable polarimetric phased array antenna testbed", *International Journal of Antennas and Propagation*, vol. 2014, 2014.

[37] R. C. Whiton, P. L. Smith, S. G. Bigler, K. E. Wilk, and A. C. Harbuck, "History of operational use of weather radar by us weather services. part ii: Development of operational doppler weather radars", *Weather and forecasting*, vol. 13, no. 2, pp. 244–252, 1998.

[38] D. Turnbull and J. MCCARTHY, "The faa terminal doppler weather radar(tdwr) program", in *International Conference on the Aviation Weather System, 3 rd, Anaheim, CA*, 1989, pp. 414–419.

[39] J. E. Evans and D. M. Bernella, "Supporting the deployment of the terminal doppler weather radar (tdwr)", *LINCOLN LABORATORY JOURNAL*, vol. 7, no. 2.1994, 1994.

[40] R. J. Doviak and D. S. Zrnic, *Doppler radar and weather observations.* Courier Corporation, 1993.

[41] P. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.* Harvard University, 1975. [Online]. Available: `https://books.google.com/books?id=z81XmgEACAAJ`.

[42] S. McCartney, *ENIAC: The triumphs and tragedies of the world's first computer.* Walker & Company, 1999.

[43] R. M. Russell, "The cray-1 computer system", *Commun. ACM*, vol. 21, no. 1, pp. 63–72, Jan. 1978, ISSN: 0001-0782. DOI: `10.1145/359327.359336`. [Online]. Available: `http://doi.acm.org/10.1145/359327.359336`.

[44] H. Fu, J. Liao, J. Yang, L. Wang, Z. Song, X. Huang, C. Yang, W. Xue, F. Liu, F. Qiao, *et al.*, "The sunway taihulight supercomputer: System and applications", *Science China Information Sciences*, vol. 59, no. 7, p. 072 001, 2016.

[45] A. Shan, "Heterogeneous processing: A strategy for augmenting moore's law", *Linux Journal*, vol. 2006, no. 142, p. 7, 2006.

[46] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, and K. Skadron, "A performance study of general-purpose applications on graphics processors using cuda", *Journal of parallel and distributed computing*, vol. 68, no. 10, pp. 1370–1380, 2008.

[47] Wikipedia, *Geforce 10 series — wikipedia, the free encyclopedia*, [Online; accessed 30-July-2017], 2017. [Online]. Available: `https://en.wikipedia.org/w/index.php?title=GeForce_10_series&oldid=791417070`.

[48] X. Yu, Y. Zhang, A. Patel, A. Zahrai, and M. Weber, "An implementation of real-time phased array radar fundamental functions on a dsp-focused, high-performance, embedded computing platform", *Aerospace*, vol. 3, no. 3, p. 28, 2016.

[49] X. Yu, "Digital signal processor based real-time phased array radar backend system and optimization algorithms", 2017.

[50] Wikipedia, *Terminal doppler weather radar — wikipedia, the free encyclopedia*, [Online; accessed 13-November-2017], 2017. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Terminal_Doppler_Weather_Radar&oldid=803108141.

[51] B. Isom, R. Palmer, R. Kelley, J. Meier, D. Bodine, M. Yeary, B.-L. Cheong, Y. Zhang, T.-Y. Yu, and M. I. Biggerstaff, "The atmospheric imaging radar: Simultaneous volumetric observations using a phased array weather radar", *Journal of Atmospheric and Oceanic Technology*, vol. 30, no. 4, pp. 655–675, 2013.

[52] J. McCarthy, "Ascribing mental qualities to machines.", STANFORD UNIV CALIF DEPT OF COMPUTER SCIENCE, Tech. Rep., 1979.

[53] W.-Q. Wang, "Detecting and mitigating wind turbine clutter for airspace radar systems", *The Scientific World Journal*, vol. 2013, p. 8, 2013.

[54] J. Perry and A. Biss, "Wind farm clutter mitigation in air surveillance radar", *Aerospace and Electronic Systems Magazine, IEEE*, vol. 22, no. 7, pp. 35–40, 2007, ISSN: 0885-8985. DOI: 10.1109/MAES.2007.4285990.

[55] R. Nepal, J. Cai, and Z. Yan, "Micro-doppler radar signature identification within wind turbine clutter based on short-cpi airborne radar observations", *IET Radar, Sonar & Navigation*, vol. 9, no. 9, pp. 1268–1275, 2015.

[56] Y. Bengio, *Learning Deep Architectures for AI*, ser. Foundations and Trends(r) in Machine Learning. Now Publishers, 2009, ISBN: 9781601982940. [Online]. Available: http://books.google.com/books?id=cq5ewg7FniMC.

[57] A. Ng, "Sparse autoencoder", *CS294A Lecture notes*, p. 72, 2011.

[58] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm", *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 1–38, 1977.

[59] C. J. Wu, "On the convergence properties of the em algorithm", *The Annals of statistics*, pp. 95–103, 1983.

[60] Y. Wang and N. L. Zhang, "Severity of local maxima for the em algorithm: Experiences with hierarchical latent class models.", in *Probabilistic Graphical Models*, Citeseer, 2006, pp. 301–308.

[61] D. M. Chickering and D. Heckerman, "Efficient approximations for the marginal likelihood of bayesian networks with hidden variables", *Machine Learning*, vol. 29, no. 2-3, pp. 181–212, 1997.

[62] L. Devroye, L. Györfi, and G. Lugosi, *A Probabilistic Theory of Pattern Recognition*, ser. Applications of mathematics : stochastic modelling and applied probability. Springer, 1996, ISBN: 9780387946184. [Online]. Available: `http://books.google.com/books?id=uDgXoRkyWqQC`.

[63] F. Kabeche, Y. Lemaître, A. Protat, S. Kemkemian, and J.-P. Artis, "Development and evaluation of a new real-time detection method of atmospheric turbulent structures by an airborne x-band doppler radar", in *Radar Conference-Surveillance for a Safer World, 2009. RADAR. International*, IEEE, 2009, pp. 1–6.

[64] A. Monakov and Y. Monakov, "Detection of turbulence with airborne weather radars using space-time filtering", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 46, no. 4, pp. 2131–2137, 2010.

[65] L. B. Cornman and R. K. Goodrich, "Doppler radar measurements of turbulence", in *Aviation Turbulence*, Springer, 2016, pp. 121–148.

[66] Y. Averyanova, "Statistical algorithm for turbulence detection using polarization features of radar reflections from rain", in *Radar Symposium (IRS), 2015 16th International*, IEEE, 2015, pp. 593–596.

[67] G. R. Widmann, M. K. Daniels, L. Hamilton, L. Humm, B. Riley, J. K. Schiffmann, D. E. Schnelker, and W. H. Wishon, "Comparison of lidar-based and radar-based adaptive cruise control systems", SAE Technical Paper, Tech. Rep., 2000.

[68] S. Franke, X Chu, A. Liu, and W. Hocking, "Comparison of meteor radar and na doppler lidar measurements of winds in the mesopause region above maui, hawaii", *Journal of Geophysical Research: Atmospheres*, vol. 110, no. D9, 2005.

[69] J. Delanoë, A. Protat, O. Jourdan, J. Pelon, M. Papazzoni, R. Dupuy, J.-F. Gayet, and C. Jouan, "Comparison of airborne in situ, airborne

radar–lidar, and spaceborne radar–lidar retrievals of polar ice cloud properties sampled during the polarcat campaign", *Journal of Atmospheric and Oceanic Technology*, vol. 30, no. 1, pp. 57–73, 2013.

[70] M. Fang, R. J. Doviak, and V. Melnikov, "Spectrum width measured by wsr-88d: Error sources and statistics of various weather phenomena", *Journal of Atmospheric and Oceanic Technology*, vol. 21, no. 6, pp. 888–904, 2004.

[71] D. Zrnic, "Spectrum width estimtes for weather echoes", *IEEE Transactions on Aerospace and Electronic Systems*, no. 5, pp. 613–619, 1979.

[72] J. K. Williams and G. Meymaris, "Remote turbulence detection using ground-based doppler weather radar", in *Aviation Turbulence*, Springer, 2016, pp. 149–177.

[73] V. M. Melnikov and R. J. Doviak, "Turbulence and wind shear in layers of large doppler spectrum width in stratiform precipitation", *Journal of Atmospheric and Oceanic Technology*, vol. 26, no. 3, pp. 430–443, 2009.

[74] K. Hon and P. Chan, "Application of lidar-derived eddy dissipation rate profiles in low-level wind shear and turbulence alerts at hong kong international airport", *Meteorological Applications*, vol. 21, no. 1, pp. 74–85, 2014.

[75] J. K. Williams, "Using random forests to diagnose aviation turbulence", *Machine learning*, vol. 95, no. 1, pp. 51–70, 2014.

[76] C. Rudin and K. L. Wagstaff, *Machine learning for science and society*, 2014.

[77] Y. P. Ostrovsky, F. Yanovsky, and H Rohling, "Turbulence and precipitation classification based on doppler-polarimetric radar data", in *Radar Symposium, 2006. IRS 2006. International*, IEEE, 2006, pp. 1–4.

[78] E. B. Tchernev, R. G. Mulvaney, and D. S. Phatak, "Investigating the fault tolerance of neural networks", *Neural Computation*, vol. 17, no. 7, pp. 1646–1664, 2005.

[79] T. M. Mitchell, "Machine learning. 1997", *Burr Ridge, IL: McGraw Hill*, vol. 45, no. 37, pp. 870–877, 1997.

[80] C. M. Shun and S. Lau, “Terminal doppler weather radar(tdwr) observation of atmospheric flow over complex terrain during tropical cyclone passages”, in *PROC SPIE INT SOC OPT ENG*, vol. 4152, 2000, pp. 42–53.

[81] R. Srivastava, A. Jameson, and P. Hildebrand, “Time-domain computation of mean and variance of doppler spectra”, *Journal of Applied Meteorology*, vol. 18, no. 2, pp. 189–194, 1979.

[82] W. P. Mahoney III and K. L. Elmore, “The evolution and finc-scale structure of a microburst-producing cell”, *Monthly weather review*, vol. 119, no. 1, pp. 176–192, 1991.

[83] P. Chan, P Zhang, and R Doviak, “Calculation and application of eddy dissipation rate map based on spectrum width data of a s-band radar in hong kong”, *MAUSAM*, vol. 67, no. 2, pp. 411–422, 2016.

[84] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, *Machine learning: An artificial intelligence approach.* Springer Science & Business Media, 2013.

[85] C. Sammut and G. I. Webb, *Encyclopedia of machine learning.* Springer Science & Business Media, 2011.

[86] Wikipedia, *Confusion matrix — wikipedia, the free encyclopedia*, [Online; accessed 28-September-2017], 2017. [Online]. Available: `https://en.wikipedia.org/w/index.php?title=Confusion_matrix&oldid=794367294`.

[87] S. D. Blunt and K. Gerlach, “Adaptive pulse compression”, in *IEEE Radar Conference*, IEEE, Philadelphia, PA, USA, 2004, pp. 271–276.

[88] ——, “Adaptive pulse compression via MMSE estimation”, *IEEE Transactions on Aerospace and Electronic Systems*, vol. 42, no. 2, pp. 572–584, 2006.

[89] S. D. Blunt and T. Higgins, “Dimensionality reduction techniques for efficient adaptive pulse compression”, *IEEE Transactions on Aerospace and Electronic Systems*, vol. 46, no. 1, pp. 349–362, 2010.

[90] Z. Li, R. Nepal, Y. Zhang, and W. Blake, “Efficient pulse compression for lpi waveforms based on a nonparametric iterative adaptive ap-

proach", in *SPIE Defense+ Security Radar Sensor Technology XIX and Active and Passive Signatures VI*, SPIE, Baltimore, MD, USA, 2015, pp. 94610X–94610X.

[91] R. Nepal, Y. Zhang, Z. Li, and W. Blake, "Matched filter based iterative adaptive approach", in *SPIE Defense+ Security Radar Sensor Technology XX*, SPIE, Baltimore, MD, USA, 2016, pp. 982 912–982 912.

[92] D. D. Feldman and L. J. Griffiths, "A projection approach for robust adaptive beamforming", *IEEE Transactions on Signal Processing*, vol. 42, no. 4, pp. 867–876, 1994.

[93] S. A. Vorobyov, A. B. Gershman, and Z.-Q. Luo, "Robust adaptive beamforming using worst-case performance optimization: A solution to the signal mismatch problem", *IEEE Transactions on Signal Processing*, vol. 51, no. 2, pp. 313–324, 2003.

[94] L. Lei, J. P. Lie, A. B. Gershman, and C. M. S. See, "Robust adaptive beamforming in partly calibrated sparse sensor arrays", *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1661–1667, 2010.

[95] Nvidia, *CUDA C Programming Guide*, (2017, accessed 31 May 2017). [Online]. Available: `https://docs.nvidia.com/cuda/cuda-c-programming-guide/`.

[96] M. Frigo and S. G. Johnson, "FFTW: Fastest Fourier transform in the west", *Astrophysics Source Code Library*, 2012.

[97] G. Guennebaud, B. Jacob, *et al.*, *EIGEN v3*, http://eigen.tuxfamily.org, 2010.

[98] Nvidia, *CUDA C best practices guide*, (2016, accessed 22 November 2016). [Online]. Available: `http://docs.nvidia.com/cuda/cuda-c-best-practices-guide/`.

[99] L. Ma, R. D. Chamberlain, and K. Agrawal, "Analysis of classic algorithms on GPUs", in *High Performance Computing & Simulation (HPCS), 2014 International Conference on*, IEEE, 2014, pp. 65–73.

[100] L. Ma, "Modeling Algorithm Performance on Highly-threaded Many-core Architectures", PhD thesis, Washington University in St. Louis, 2014.

[101] M. Wolfe and P. C. Engineer, "Understanding the cuda data parallel threading model a primer", *PGI Insider, February*, 2010.

[102] F. Kong, Y. R. Zhang, J. Cai, and R. D. Palmer, "Real-time radar signal processing using gpgpu (general-purpose graphic processing unit)", in *SPIE Defense+ Security*, International Society for Optics and Photonics, 2016, pp. 982 914–982 914.

[103] Wikipedia, *Geforce 700 series*, (2016, accessed 13 September 2016). [Online]. Available: `https://en.wikipedia.org/w/index.php?title=GeForce_700_series&oldid=749513806`.

[104] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer, "Cudnn: Efficient primitives for deep learning", *arXiv preprint arXiv:1410.0759*, 2014.

[105] Wikipedia, *Caffe (software) — wikipedia, the free encyclopedia*, [Online; accessed 10-October-2017], 2017. [Online]. Available: `https://en.wikipedia.org/w/index.php?title=Caffe_(software)&oldid=798557658`.

[106] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[107] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

Appendix A

Summary of Contributions

Journal Papers:

[1] J. Cai and Y. Zhang, ”General purpose graphic processing unit implementation of adaptive pulse compression algorithms”, *Journal of Applied Remote Sensing*, Vol. 11, No. 3, pp. 035009, Aug. 2017.

[2] R. Nepal, J. Cai, and Y. Zhang, ”Micro-Doppler radar signature identification within wind turbine clutter based on short-CPI airborne radar observations”, *IET Radar, Sonar & Navigation*, Vol. 9, No. 9, pp. 1268-1275, Dec. 2015.

[3] J. Cai, Y. Zhang, R. Doviak and P.W. Chan, ”Diagnosis and Classification of Typhoon-Associated Low-Altitude Turbulence Using HKO-TDWR Radar Observations and Machine Intelligence”, *IEEE Transactions on Geoscience and Remote Sensing*, Submitted and under revising, Nov 2017

Conference Papers:

[1] J. Cai, Y. Zhang, F. Kong and L. Li, ”Acceleration of advanced radar processing chain and adaptive pulse compression using gpgpu”, in *Proceedings of the 24th High Performance Computing Symposiumm, Society for Computer Simulation International*, 2016, pp. 7:1-7:6.

[2] J. Cai and Y. Zhang, ”Acceleration of generalized adaptive pulse compression with parallel GPUs”, in *SPIE Defense+ Security, International Society for Optics and Photonics*, 2015, pp. 946107–946107.

[3] F. Kong, Y. Zhang, J. Cai and R. Palmer, ”Real-time radar signal processing using GPGPU (general-purpose graphic processing unit)”, in *SPIE*

Defense+ Security, International Society for Optics and Photonics, 2016, pp. 982914–982914.

[4] J. Cai, R. Mirza, Y. Zhang and J. Tilley, ”Concept Design and Feasibility Studies for a Ka-band, UAS-based Cloud Sensing Radar”, in *37th Conference on Radar Meteorology, AMS*, 2015

[5] Z. Li, Y. Zhang, S. Wang and J. Cai, ”Superresolution processing for multifunctional LPI waveforms”, in *Proc. SPIE Defense, Security, Sens.—Radar Sensor Technol. XVIII*, 2014, pp. 90770M

[6] Y. Zhang, Z. Li, B. Cheong, J. Cai, R. Nepal, H. Suarez, W. Blake, J. Andrews and T. Yong, ”Optimized radar signal processing for a low-cost, solid-state airborne radar”, in *36th Conference on Radar Meteorology, AMS*, 2013