8-2020

Searching for Low Frequency Fast Radio Bursts with VLITE

Suryarao Bethapudi
The University of Texas Rio Grande Valley

SEARCHING FOR LOW FREQUENCY FAST RADIO BURSTS WITH VLITE

A Thesis

by

SURYARAO BETHAPUDI

Submitted to the Graduate College of
The University of Texas Rio Grande Valley
In partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2020

Major Subject: Physics

SEARCHING FOR LOW FREQUENCY FAST RADIO BURSTS WITH VLITE

A Thesis
by
SURYARAO BETHAPUDI

COMMITTEE MEMBERS

Dr. Teviet Creighton
Chair of Committee

Dr. Volker Quetschke
Committee Member

Dr. Soumya Mohanty
Committee Member

August 2020

ABSTRACT

Bethapudi, Suryarao, Searching for low frequency Fast Radio Bursts with VLITE . Master of Science (MS), August, 2020, 61 pp., 15 tables, 28 figures, 26 references.

The VLITE (VLA Low Band Ionosphere and Transient Experiment; http://vlite.nrao.edu) program performs commensal observations using 16 antennas of the Very Large Array radio telescope from 320–384 MHz. The VLITE-Fast program searches for short time-scale (<100ms) transients, such as Fast Radio Bursts (FRBs), in real time and triggers recording of baseband voltages for offline imaging. Searches are made possible by a 12 node cluster, each housing GPUs for digital signal processing. A real-time Message Passing Interface (MPI)-based co-adder incoherently sums the data streams from all the antennas to boost the signal-to-noise. To undo the dispersion effects of signal propagation through the ionized interstellar medium, the co-added stream is de-dispersed and matched-filtered to search for transients. This operation is completely performed on GPUs by the software package Heimdall . A selection logic is applied to the candidates and interesting candidates with their corresponding data are processed and packaged in a binary file along with a diagnostic plot. Furthermore, a Machine Learning classification is developed and applied on the reduced data product and, based on its decision, baseband voltages are recorded. Reduced data products collected over 126 days of on-sky operation form the VLITE-Fast Pathfinder Survey (VFPS). This pipeline has triggered on single pulses from 7 known radio pulsars. Lastly, the pipeline capabilities are tested against pure random noise and simulated injected signals.

DEDICATION

For the dreading drag until death liberates us.

ACKNOWLEDGMENTS

This work would not be what it is if not for the guidance and patience offered by Dr. Matthew Kerr. I am ever grateful to Dr. Matthew for that.

I am also grateful to all those who have supported me until now.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

CHAPTER I

FAST RADIO BURSTS

The domain of Radio Astronomy is full of surprises. What started off as an experiment to communicate across the Atlantic ocean using radio waves soon took turn when Karl Jansky recorded radio signals emanating from the direction of our galaxy's center and established the idea of using radio telescopes to *look* at the heavens. Many decades later, the same radio astronomy discovered a new type of stars which shines but in radio regime and also pulsates like a light house. These were characterized to be pulsating source of radiation and were aptly named as pulsars. Half a century later during which thousands of pulsars were discovered and documented, a new type of a signal emerged. It was a serendiptious detection of *a bright millisecond radio burst* [12] in the archieval data which brought into light, the existence of short time-scale, extremely energetic radio signals, possibly originating from beyond the galaxy. These signals are called *Fast Radio Bursts (FRB)* and the search for these signals forms the crux of this thesis.

The first discovered FRBis the [12] in 2007, colloquially known as the *Lorimer Burst* (see chapter I). Many theory papers followed but there was no corroborative detection. In the lack of which, skeptisim arose [3]. Prior to that [11] detected FRB-like pulses with DM as high as 5×10^3 originating from M87. So, these may very well have been the first detection of FRBs but due to the limited time/frequency resolution, no substantiative evidence could be established.

Astronomers were not wrong in their dubious nature. Their experience with another species of signals called Perytons [16] justifies it. Named after the mystical hybrid bird, these signals were actually caused by the microwave on the observatory site. The premature opening of the microwave door shuts off the magnetron abruptly, which produces a dispersed-like signal which was then picked up by the radio telescope. This put more shade on the celestial nature of the *Lorimer Burst*.

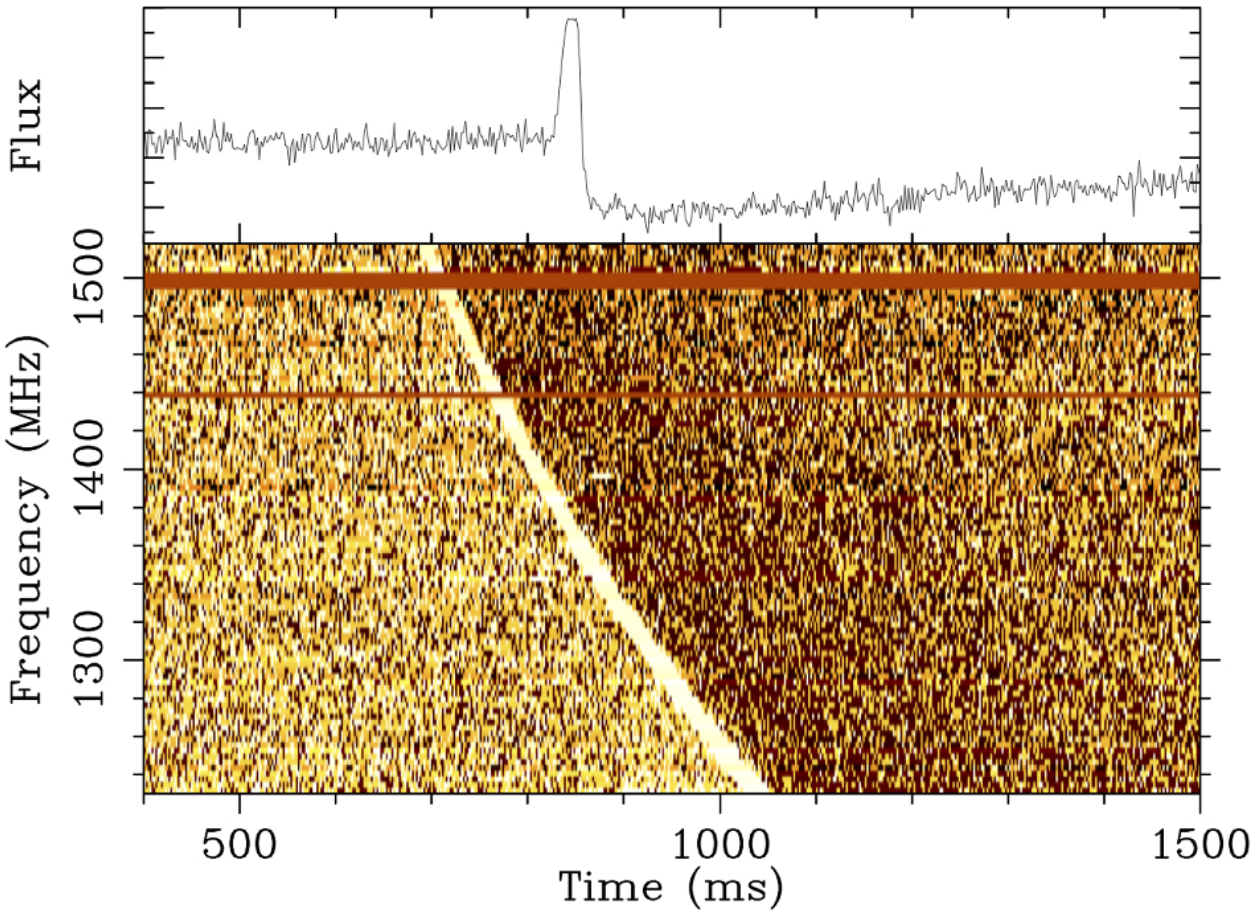


Figure 1.1: Lorimer burst. The first ever FRB detected. The curve in the filterbank (see subsection 1.1.1) is due to a propagation effect called dispersion (see subsection 1.1.2). The sheer strength of the signal is self evident when looking at the amplitude of the peak.

But then came as a surprise, [22] in 2016, almost a decade later, which reported 4 such bursts and thus set in stone the existence of such species. The hunt was on. Now, four years and many sophisticated advancements in observatory technologies later, the number of FRBs detected is in hundreds. The details of this story will be discussed in section 1.2.

Given the amount of shear energy emitted in these bursts, astronomers were not wrong in maintaining an innate assumption that these events were result of a cataclysmic event. And by the very nature of such events, FRBs were not expected to repeat. And then came as a surprise, FRB121102 in 2012, which repeated in time. Over the years since its discovery, FRB121102 is perhaps the most studied FRB with observations done from meter wavelength to milli-meter wavelength. See [20, 19].

Astronomy wouldn't be full of surprises, if FRB121102 was the only repeating FRB found. In 2019, [4, 6] found 4 repeating FRBs (henceforth called repeaters) using the Canadian Hydrogen Intensity Mapping Experiment (CHIME) telescope which prompted a fresh set of theories to emerge to explain the same.

All the discovered FRBs are catalogued in FRBCAT [14]. All the theories of FRBs are also catalogued in FRB-theorycat [17].

1.1 What are FRBs?

Before we go into FRBs, it is crucial to establish certain groundwork known in a radio astronomy setting. First being the data representation format widely used: filterbank in subsection 1.1.1. Second being the phenomenon of dispersion which affects all the radio signals propagating through the space and reaching the radio telescope. This will be covered in detail in subsubsection 1.1.2. Then, in subsection 1.1.2, FRBs are characterized.

1.1.1 Filterbank

A popular data representation format for radio astronomers is the time frequency bins, known as filterbank, where every bin has one or more of the four Stokes parameter. In a typical setting, of the four, only the Stokes I is recorded. Any radio astronomy setup involves a frequency band

$[\nu_{\text{LOW}}, \nu_{\text{HIGH}}]$ of interest discretized into NCHAN frequency channels, yielding frequency channel width, denoted by FOFF, as bandwidth divided by the number of channels. Time samples are sampled at a suitable Nyquist Sampling rate commensurate with the bandwidth $\nu_{\text{HIGH}} - \nu_{\text{LOW}}$. Let the sampling rate be TSAMP.

With the above definitions in place, starting off with voltage samples, by applying Short Time Fourier Transform and consequently taking magnitude (essentially Stokes I), filterbank is produced. In practise, a suitable digitization scheme is employed and data is digitized to NBIT bits per sample. Every consecutive time sample is separated by TSAMP in time and every consecutive frequency sample is separated by FOFF. This representation makes it easier to look at the time frequency variations and hence is popular.

1.1.2 Dispersion

Imagine sending sunlight through a prism. It is common knowledge to expect rainbow emanating from the prism. This same phenomenon is what brings a rainbow after the rain if the Sun is up. The physics behind this as follows: White light is made up of the seven colors. And, when white light goes through the prism, distinct color components get separated, and a rainbow is observed. This phenomenon is called dispersion and is not just valid for white light (or, visible band of electromagnetic spectrum). Of course, the properties of the prism changes when going to a different band of electromagnetic spectrum. But, any part of the electromagnetic spectrum can get dispersed and its frequency components can get separated. And since different frequencies of visible light correspond to different colors, we see colors of a rainbow.

Radio waves propagating in space also suffer the same fate. *Prisms* here are plasma made of electrons in the space. This plasma exists in the line of sight between the source and the observatory. A better understanding of the dispersion can be made by using the filterbank representation. Dispersion causes progressively lower frequencies to receive the signal latter than their higher frequency counterpart. See fig.

The time difference (also known as dispersion smearing) is calculated with the Equation 1.1. The DM appearing in the equation is called the Dispersion Measure which is a measure of the electron

number density in the column along the line of sight.

$$\Delta t = \mathrm{DM} \times 4.15 \times 10^{-3} \Big[\frac{1}{\nu^2_{\mathrm{LOW}}} - \frac{1}{\nu^2_{\mathrm{HIGH}}} \Big] \ \mathrm{s} \tag{1.1}$$

This effect of dispersion is to be corrected for any analysis and the process is called de-dispersion. More details about de-dispersion will be covered in chapter II.

Dispersion is purely a propagation effect. Thus, amount of dispersion provides a good marker of distance. There are two models for electron distribution in the galaxy, NE2001 [5] and YMW [25]. With the help of a model, given a pointing in Galactic coordinates (l, b), the DM contribution by the Galaxy can be estimated. This model becomes useful later when talking about FRB origin in following sections.

De-dispersion In order to compensate for the dispersion effect, de-dispersion is performed. The de-dispersion is a computationally intensive process.

With a filterbank, de-dispersion can be straightforward where frequency channels are time-shifted can do the job. However, this does not negate dispersion effects in the frequency channel, leading to what is called the in-channel smearing. Mathematically, it can be derived by using small bandwidth approximation to Equation 1.1. See Equation 1.2 where $\Delta\nu$ is the channel bandwidth and ν is the frequency of the top end of the channel.

$$\Delta\tau = \mathrm{DM} \times 4.15 \times 10^{-3} \Big[\frac{\Delta\nu}{\nu^3} \Big] \ \mathrm{s} \tag{1.2}$$

1.1.3 Characteristics

FRBs are short time-scale ($\sim$ ms), bright ($0.001 - 100$ Jy), broadband (present in *almost* all frequency channels) and extremely dispersed (very large DM) radio signals. The most distinctive feature of an FRB is its DMwhich is many fold more than the galaxy could contribute. The galaxy contribution to a signal detected at an observatory is calculated using the pointing information (at the time of detection) and using an eletron density model as discussed in subsubsection 1.1.2.

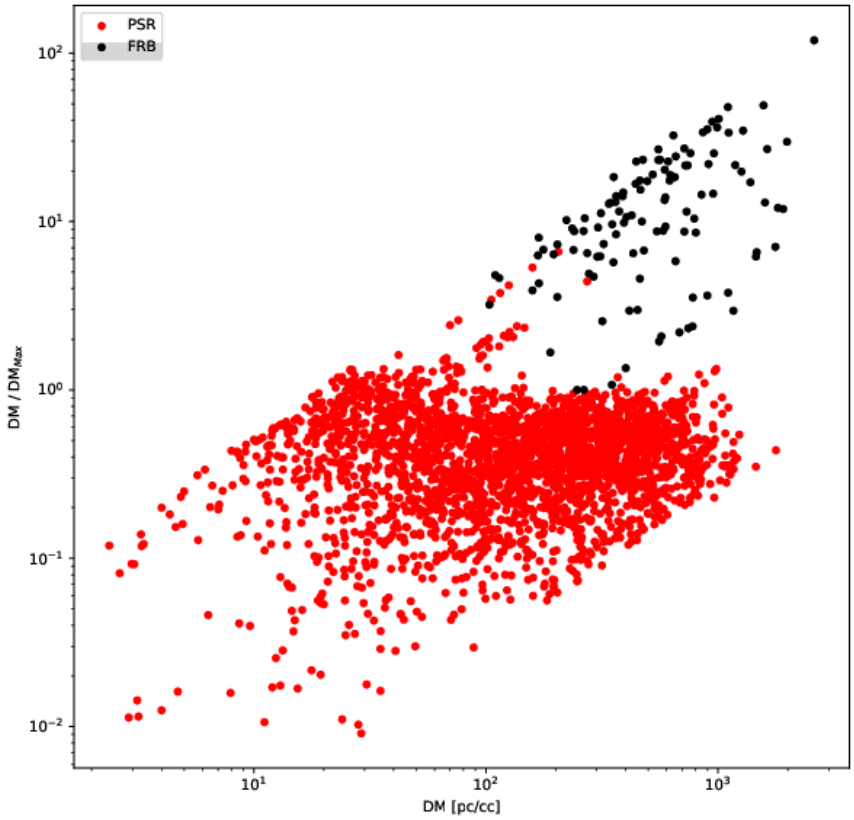


Figure 1.2: DMv/s $\frac{\text{DM}}{\text{DM}_{\text{Max}}}$. An updated version of [15, 19].

To further appreciate the high DMof FRB , the DMand $\frac{\text{DM}}{\text{DM}_{\text{Max}}}$ are plotted. DM_{Max} is maximum galactic contribution for the given pointing. See subsection 1.1.3 also [15, 19]. Clearly, all the FRBs are atleast multiple times the galactic contribution. This inference, combined with the understanding that DM is a proxy for distance, provides evidence that FRBs origin from extra-galactic and of cosmological nature.

FRBs also exhibit other features such as scintillation and scattering. Scintillation is the twinkling of stars phenomena one can observe. Scintillation presents itself in FRBs as the pulse disappearing for some frequency channels and appearing back when looking at the filterbank. This can be observed in the Figure 1.1.3. Scattering is the broadening of the pulse in the filterbank. This is also seen as an exponential tail in the frequency averaged profile. Also see Figure 1.1.3.

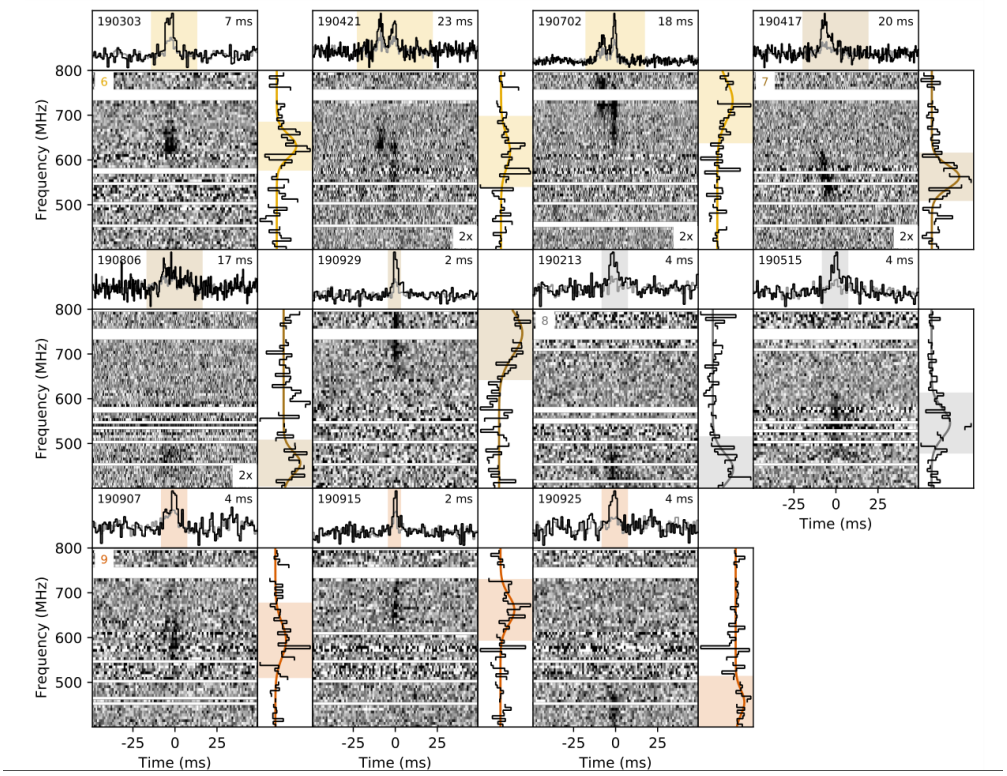


Figure 1.3: A set of repeating FRBs detected [6]. Scattering causes the broadening of the pulse.

1.2 Detections so far

Using the catalogued FRBs from FRBCAT mentioned before, a brief summary of the FRBs population is given. Firstly, section 1.2 summarizes the FRBs detected by various instruments over time. Figure 1.2 shows the sky distribution of the FRBs. If FRBs are indeed extra-galactic, the sky distribution would be isotropic.

1.3 Thesis outline

1.3.1 VLITE

This study is a part of the VLA Low Frequency Ionospheric and Transient Experiment (VLITE). VLITE is a commensual observing system of the Karl G. Janksy Very Large Array radio telescope (https://science.nrao.edu/facilities/vla) upon which VLITE-Fast is based. Hence before describing the outline, the VLITE system is described here.

A selected subset of VLA antennas are fitted with low frequency receiver units at the casegrain

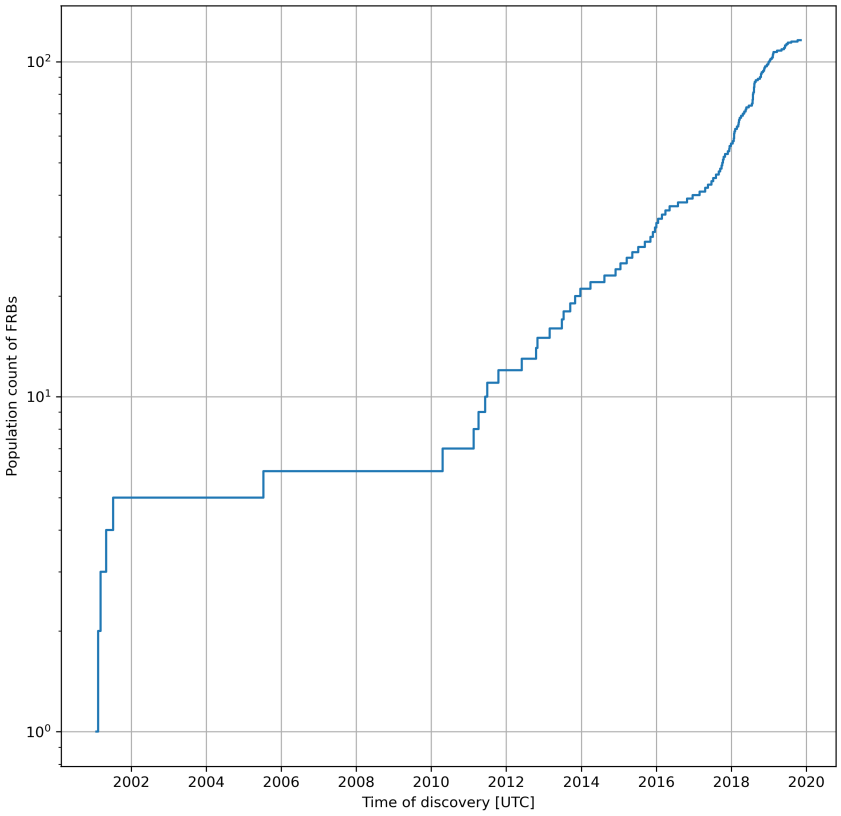


Figure 1.4: Population of FRBs over time. The exponential trend is what drives VLITE-Fast .

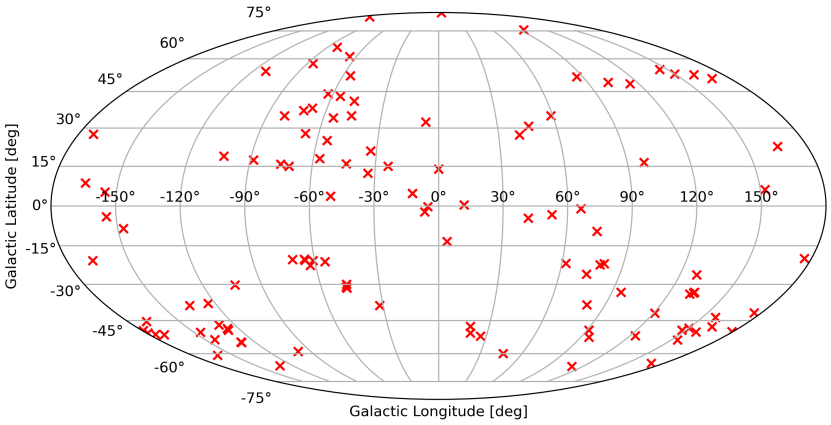


Figure 1.5: Population of FRBs over sky.

feed. Receivers are tuned to operate in $320 - 384$ MHz giving a bandwidth of 64 MHz. Due to the Mobile Users Operating System (MUOS) of the Department of Defense, the higher end of the band receives a lot of Radio Frequency Interference (RFI), as a result, the frequency band is cut off at 361 MHz. Nevertheless, the voltage data is sampled at 128 Mhz (the Nyquist frequency of 64 MHz bandwidth).

The sampled data from each of the antenna is multicast on the observatory intranet using the User Datagram Protocol (UDP) in the form of VLBI Data Interchange Format (VDIF) packets. At the time of writing, there are 16 VLA antennas participating in VLITE. VLITE also consists of 12 compute nodes with one login node which are interconnected by an Infiniband network [18]. Each compute node houses nVIDIA Graphical Processing Units (GPUs) for all the signal processing and also has a 500 GB Solid State Disk (SSD) installed.

This infrastructure is shared by two different pipelines: VLITE-SLOW and VLITE-Fast . VLITE-SLOW is an imaging pipeline which produces short time skymaps and is not the focus of this thesis. VLITE-Fast is a search pipeline which searches for FRBs.

1.3.2 VLITE as an FRB search engine

VLITE is a well suited engine to detect FRBs. The salient features are summarized here:

Commensal VLITE being a commensal system has uncontested access to the data from the VLAantennas. This leads to extremely large onsky times which are the times actively looking for signals of interest.

Large field-of-view (FOV) Each VLITE antenna is a $25-$meter dish with center frequency of 350 MHz (0.85 meters in wavelength). The diffraction limit of each antenna is $\frac{\lambda}{D}$ where D is the diameter of the dish and λ is the wavelength. VLITE achieves a diffraction limit of 0.034 radians or 1.94^o. An $\sim 2^o$ FOV covers a large portion of the sky.

Sensitivity VLITE employs 16 antennas. Use of a large number of antennas helps in reducing the overall background noise and thereby increases the sensitivity of the signal.

Inteferometer An inteferometer is a type of radio telescope with spatially separated antennas. The spatial separation helps produce a delay between any pair of antennas which is then used to triangulate (localize) the source of the signal. VLA is an interferometer. And, hence, VLITE is too. Given how far the FRBs are originating from, any task of localizing the cause/source of the signal is hindered by the inability to precisely identify the particular pointing in sky from where the signal has originated. Here lies the strength of VLITE .

With the help of these features, VLITE endevors to be a *FRB localizing search engine*.

1.3.3 Outline

The goal of the study is to establish and start a search campaign making use of the commensual observation system VLITE of the VLA radio observatory. With this in mind, a robust yet efficient realtime search pipeline was designed and coded (see chapter II). The data taken became a part of VLITE-Fast Pathfinder Survey (VFPS). The data is discussed in great detail in chapter III. A completeness analysis, in which pure random noise was sent through pipeline, is performed to test the noise response of the pipeline. In addition, artificial signals of interest were inserted into the pipeline for testing the pipeline in a controlled environment. The data products yielded by VFPS and the controlled testing have been used to develop an Aritificial Intelligence (AI) which will be covered in chapter IV. The resulting AI solution is used to vet through the entire VFPS dataset to identify any unknown signal missed previously. Results of the AI analysis are also reported in the same chapter.

CHAPTER II

INSTRUMENTATION

This chapter explains the various pieces of hardware/software which make `VLITE-Fast` possible. As with any data processing/search pipeline, the design consists of various components which are intermediated by data buffers. Such a design allows for modular component design wherein every component is assigned to be either producer or consumer depending on its position and makes it simplier. This practise is not novel in that it is known as the *producer-consumer* model.

A brief overview of the infrastructure is summarized in subsection 2.0.1. The pipeline is graphically described in chapter II. All the data products/formats are defined in section 2.7.

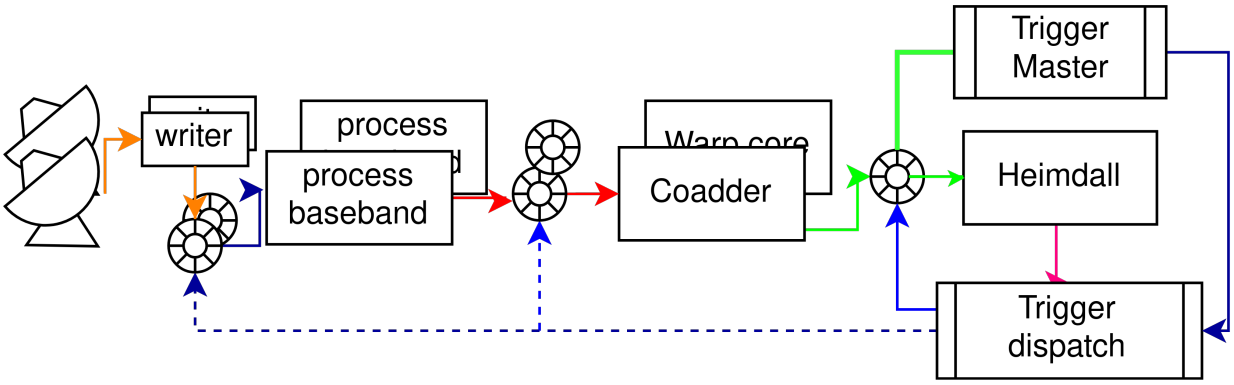


Figure 2.1: `VLITE-Fast` pipeline block diagram. See text.

2.0.1 Overview

Firstly, all the components and the data buffers technology are listed and then approached into details in separate sections.

PSRDADA Data buffer technology

Writer Baseband data packet capturing and writing

Process Baseband Baseband data to filterbank data

Coadder Co-adding filterbank data from all antennas

Heimdall Search program

Trigger mechanics Set of python scripts for various trigger level activities

TriggerHook Identifies slice of data and copies it for future analysis

TriggerMaster Realtime trigger responses

Writer and Process Baseband have been developed, written by Dr. Matthew Kerr. The author has written Coadder, Trigger mechanics, Trigger Hook and TriggerMaster.

2.1 PSRDADA

The use of an intermediate buffer is ubiqitious in any real time pipeline. In a producer-consumer model, a producer emits data which is to read by the consumer. In a realtime constrained pipeline, the datarates may not be consistent all the time which may lead to consumer waiting for new data (producer is slow), or unable to accept new data (producer is fast). Hence naturally, many intermdiate buffers are used in `VLITE-Fast` pipeline design.

`VLITE-Fast` makes use of `DADA` as an intermediate buffer technology. `DADA` is built on SYSV shared memory model thus makes use of linux system's kernels and shared memory infrastructure for offering buffering capability. `DADA` codebase is in written in *C* for linux type machines. `DADA` is versatile that it can support the following modes:

- Single producer - Single consumer (SPSC)
- Single producer - Multiple consumer (SPMC)
- Multiple producer - Single consumer (MPSC)
- Multiple producer - Multiple consumer (MPMC)

Of them all, VLITE-Fast makes use of SPSC, SPMC oweing for a simple logic which involves a single producer.

DADA can be visualized as an n-element ring buffer with each ring holding b bytes of data. In addition to the data elements, DADA also offers header blocks which can be used to store data descriptors and such information is stored in ASCII. Header is defined as key value pair with key being a string and value being any data type. chapter II shows DADA buffers as the concentric rings made of blocks.

2.2 Writer

The writer code is wholly contributed by Dr. Matthew T. Kerr

VLITE-Fast is a commensually operated search pipeline. Raw baseband voltage data comes in VLBI Data Interchange Format (VDIF) packets over User Datagram Protocol network sent by observatory level program called executor. The job of writer is to intercept these data packets and write the header and data into DADA buffer.

The raw baseband data consists of two-polarizations, 64MHz bandwidth real sampled at Nyquist sampling rate of 128MHz. Writer captures these packets, checks for time ordering and writes into DADA buffer. Time integrity check is of paramount importance since all the following components assume a time continuous flow of data.

In case writer finds that the incoming packets fail the time continuity check, it is designed to zero fill the packets in between so at least data is continuous in time. The causes for such are many. Sometimes, the ethernet network load may be high, or sometimes, the receiving nodes CPU usage is high that it can't respond in time. It has also been observed there are some nodes which are more suspectible to packet drops than other nodes. Such nodes are ignored until the underlying issue is resolved.

Another job of writer is to respond to voltage triggers (see section 2.6). For every voltage trigger received, writer writes out one second VDIF file, containing packets, to the Solid State Drive (SSD) onboard for the duration of the trigger. Care is taken that overlapping triggers don't cause duplications of raw voltage files.

2.3 Process baseband

The process baseband code contributed by Dr. Matthew T. Kerr

Reading raw baseband produced by `writer`, `process_baseband` hereafter, `pb`, performs the following operations to output filterbank data into another `DADA` buffer:

1. Channelization using Fast Fourier Transform (FFT)
2. Polarization addition
3. Bandwidthing and bandwidth normalization
4. Kurtosis filtering
5. Digitization

All the above operations are performed on `nVIDIA` Graphical Processing Units (GPUs). Channelization uses `CUDA-FFT (cuFFT)`. All other operations involve user kernel launches. Addition of polarization yields Stokes `I` intesity. The upper edge of the `VLITE-Fast` frequency band is polluted by the Mobile Users Operation System (MUOS), hence, the maximum frequency is cut off at 360MHz bringing the effective bandwidth to 42MHz.

For a large chunk ($\sim$ 1s)of data, the bandpass (time averaged frequency profile) is normalized which removes any frequency variations on the time scales of a second. Since, the signals of interest are much smaller in time and dispersed, such normalization doesn't harm them. In addition to this, there is also a kurtosis based filtering employed which removes any short timescales spurious data.

Last operation performed in this component is digitization which digitizes filterbank data with `NBIT`. Digitization scheme employed follows from [9]. Now, the output data is written to another `DADA` buffer.

2.4 MPI Coadder

Coaddition is the process of averaging time-aligned data streams to produce an averaged data stream. The operation of averaging reduces the noise floor and helps boost the signal. The

motivation for coaddition is firmly established in subsection 2.4.1. subsection 2.4.2 explains the workings which make coaddition possible in `VLITE-Fast` .

From here on, coadder refers to the component of the `VLITE-Fast` which performs coaddition. The underlying technology upon which coaddition operation is based is the Message Passing Interface (`mpi`), hence, the coadder is also called `mpi` coadder.

Coadder reads the filterbank data written to `DADA` buffer by for all the antennas, coadds, and writes the coadded filterbank data to a `DADA` buffer residing on a specific node (also called the root node) for further operations. In short, the coadder performs, *read-coadd-write* operation in every iteration.

2.4.1 Need for coaddition

By the Central Limit Theorem, the distribution of the mean of independent identically distributed (iid) random variables is a Gaussian distribution with mean as the population mean and standard deviation (std. dev.) as population std. dev.divided by $\sqrt{N}$. Noise in the filterbank datastream coming from each of the antenna can be modeled as independent Gaussian noise. Coaddition reduces the standard deviation of the noise and hence helps in bringing out the signal. The extent of the attentuation of noise is determined by the sample size i.e, the number of antennas.

The signal boosting effect is also established while injecting signals. A large amplitude dispersed signal has to be injected in a single antenna to produce the same S/Nas in case of coadded antenna. This will be revisted in subsection 3.6.2.

2.4.2 MPI

Message Passing Interface (`mpi`) is a technology ubiqitiously used in all High Performance Computing (HPC) settings. `mpi`offers various paradigms designed and optimized for parallized jobs. One such paradigm and also the foundational element is the collective algorithm. `mpi`defines a collective algorithm as one in which all the processes have to participate as a collection, hence the name collective algorithm. Of all the collective algorithms, only two are used in `VLITE-Fast` and discussed here:

Broadcast One process has data which it needs to send to every other process.

Reduce All processes have data which is to be combined by some operator and the result is to be made available to a process.

In addition to the above collectives (parlance uesd by the HPC community to describe a collective operation), there is also the barrier operation which is used to synchronize processes.

The broadcast operation is performed when one process has data (or message) which is sent to all the other processes. At the end of this operation, all processes have identical data. The reduction operation is performed when all processes have some data which is to combined (or reduced) by some operation. The reduced data is present in a pre-defined process called the

`mpi` also offers capability of tuning by the Multi Component Architecture (MCA). MCA helps you to configure the various runtime parameters of `mpi`jobs from pre-defined options to increase the performance. The tuning options used are described in subsection 2.4.4.

2.4.3 Coaddition

The operation of coaddition involves summation of the data from all the participating processes followed by a scaling. Naturally, the reduce operation with summation operator is a perfect fit for the summation step. The scaling operation can then be performed locally. The data is read from `DADA` buffer (which is written to by). This data has been digitized to NBIT by `process-baseband`. The first operation of coaddition is to convert the NBIT data into `float32` data. This operation increases the memory footprint by four folds but prevents any overflows in reduce operation and is necessary. Next comes the actual `mpi`reduce call after which, the coadded filterbank data (in `float32`) is present in

Before any reduction is performed, it is paramount to check if all the data are time-aligned. This time alignment is checked by the `MJD`timestamp of the first datum in the array. The `ROOT` timestamp is broadcasted to all the other processes which is then compared against the process's own `MJD`timestamp. If both the `MJD`s match, actual filterbank data is used for reduction otherwise a zero-filled array with the same size as the actual filterbank data is used.

The scaling operation done after the `mpi` reduce call requires the correct number of antennas which have participated in the collective with real data and not with zero data. This is computed by another `mpi` reduce operation which is done on a single integer set to 1 if the process is participating with real data, otherwise set to 0. This operation reduces the actual number of antennas which have contributed for the coaddition.

Coaddition is the major bottleneck of the pipeline and it is also the most computationally expensive step. A typical gulp of the data is 8 seconds which with a typical NCHAN=4096, NBIT8, and sampling time amounts to 10 MB. Due to the casting to `float32`, the gulp memory footprint shoots up to 40 MB. For a typical number of antennas of 15, every iteration of coaddition involves 600 MB of floats summed across 15 antennas, making the total bandwidth of the operation to $600 \text{ MB} \times 15 = 4.8$ GB. Naturally, the performance of the pipeline depends on the optimizations of the `mpi`operations. Two of the major optimizations are discussed in the following sub-sections.

2.4.4 Reduce

Reduction step is the heart of the coaddition. Any sort of delay in the coaddition would hurt the realtime operations of the coaddition. Consequently, choice of reduction algorithm was delibrated upon. The following is heavily relied on the seminal paper [21].

The default `mpi`algorithm for reduce is the binomial tree reduction. The essence of this algorithm is that one process sends its data and other process receives it and adds it with its own data. This repeats until there is only one process. It is graphically visualized to be a binomial tree with individual node's data as the leaves and the reduced data as the root. See subsection 2.4.4.

Rabenseifner algorithm [21] is an alternate reduce algorithm which is well suited for large messages. Here, every process instead of reducing the entire message is only responsible for reducing a chunk of the message. After which, all the processes send their reduced chunk back to root. The entire process is visualized in Figure 2.4.4.

In order to compare different reduction algorithms, a suitable metric has to be introduced. Metric employed here is loosely based on [21]. Two factors govern the metric. One, total number of messages sent. Two, total number of bytes sent. The former measures the latency which is treated

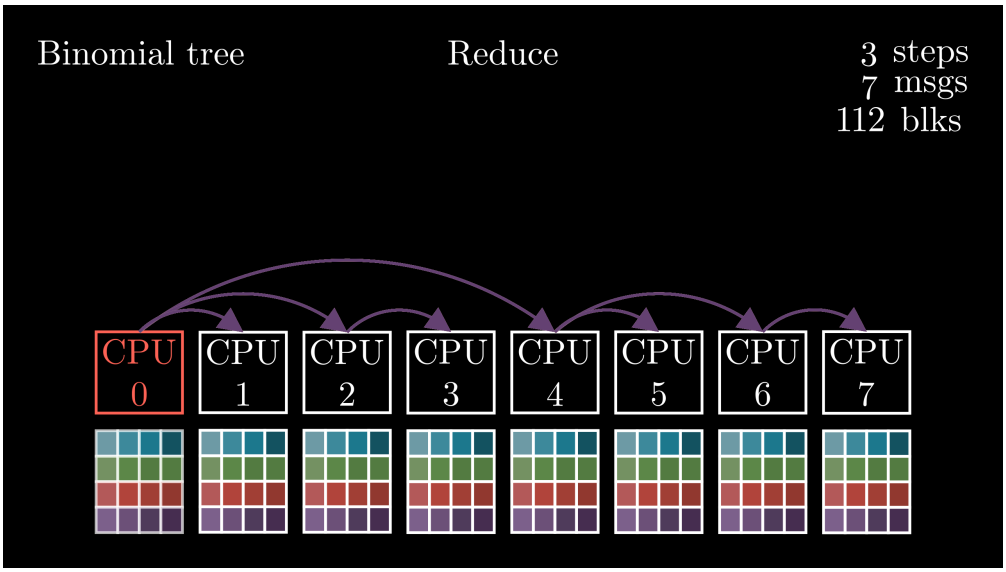


Figure 2.2: Binomial algorithm for `mpireduce`.(c.f. [21]) It looks like an inverted binomial tree, hence, the name. This graphic is an frame grabbed from this `https://youtu.be/p26iZX0sWgQ` which is also made by me.

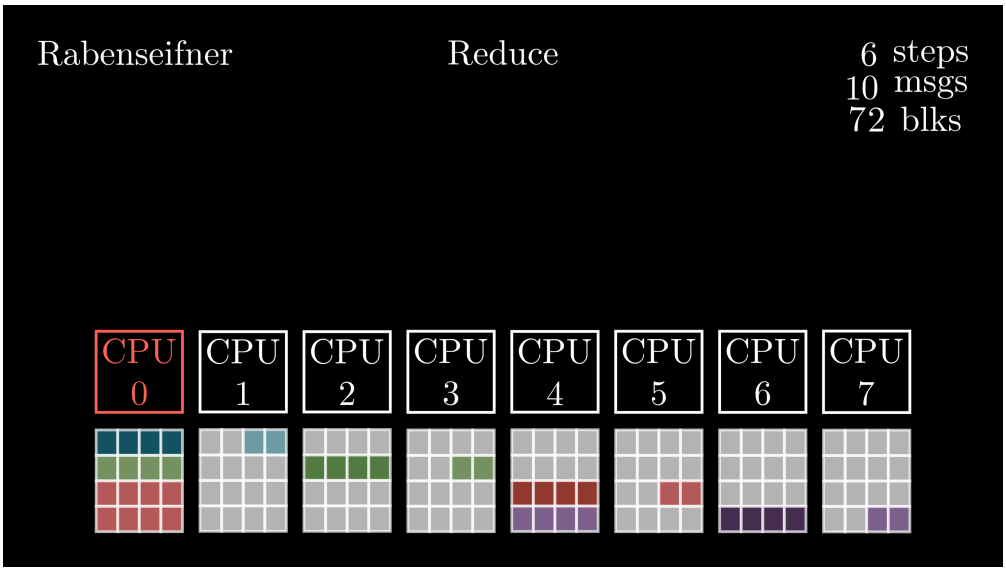


Figure 2.3: Rabenseifner algorithm for `mpireduce`.(c.f. [21]). This graphic is an frame grabbed from this `https://youtu.be/cwi5kWgizPc` which is also made by me.

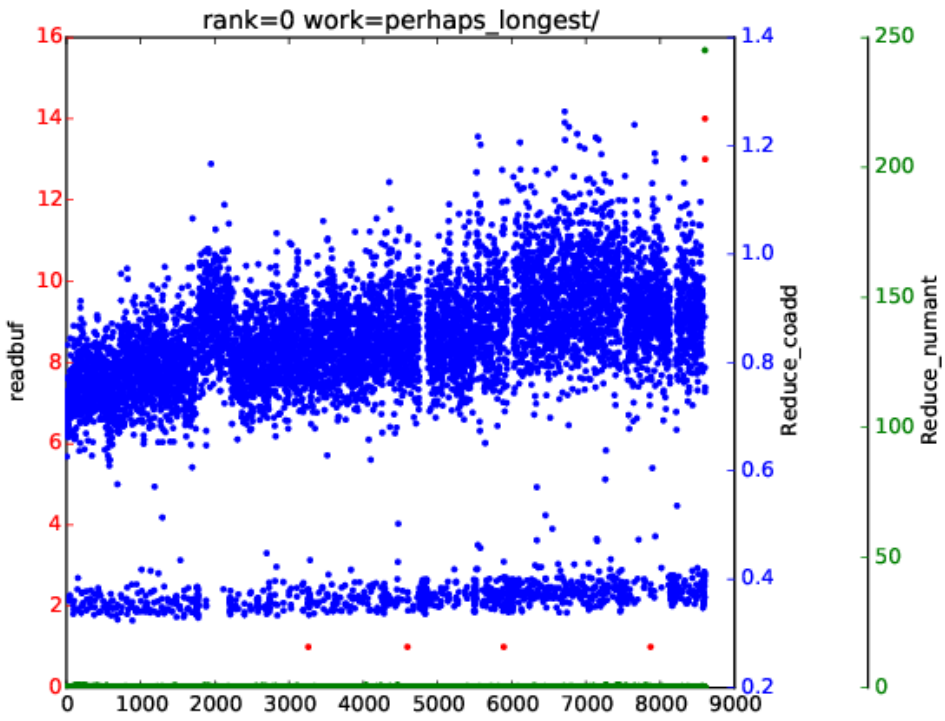


Figure 2.4: The linear trend in `reduce_coadd` (shown in blue) as number of iteration increases due to overhead on underlying transport protocol. The extremely large `reduce_numant` (shown in green) is due to the (TCP) transport protocol hanging. See text. `readbuf` (shown in red) is the time taken to a chunk of filterbank data. The x-axis in the plot is the iteration number. For this plot, the NBIT2 and chunk size was 8 seconds of data. This is from the early days of `VLITE-Fast` when 19^h was considered a long run, hence the name `perhaps_longest`.

as a constant for any size of message. The latter measures the total bandwidth requirement. The metrics are only mentioned here. Interested readers are requested to refer [21]. The metrics are tabulated in Figure 2.4.4.

The binomial reduce operation uses less messages but the total number of bytes sent is large. A linear trend is evident in the reduce_coadd. This issue caused multiple pipeline failures around 6^h mark when the lag in the step gets too high and pipeline cannot function realtime. The choice of Rabenseifner algorithm provides first aid to this issue. The linear trend is so slowly increasing that pipeline doesn't get derailed due to it.

Name	Latency	Bandwidth	Local
Binomial	log N	S	S
Rabenseifner	2 log N	$2\ \frac{p-1}{p}$ S	$\frac{p-1}{p}$ S

Table 2.1: Metric for comparing the different reduce operations. c.f. with [21]. *N* is number of processes (or number of antennas) and *S* is the total size of chunk in each antenna. Rabenseifner although has double the latency, the bandwidth is only a fraction, hence is suitable for large messages.

2.4.5 Bookkeeping

The broadcast of `MJD` is a crucial book keeping step. It ensures only time-continuous data goes down the pipeline. As discussed above, this is achieved by `ROOT` keeping `MJD` and other nodes comparing their own timestamp with it. This operation involves a broadcast of `float32` followed by simple logic where each process decides to send actual data or zero-filled data.

In the `Reduce_numant` step, the number of antennas which have participated in the coaddition is computed. It is a reduce by sum operation where those processes which participated send one and those which didn't send zero. The resultant is stored in `ROOT`. It was observed that occasionally the the time required would shoot to impossibly large times causing pipeline failure. This is observed in Figure 2.4.4. To alleviate this issue, `tcp` protocol use was dropped.

2.5 Heimdall

The search program employed by `VLITE-Fast` is Heimdall([2]). Heimdall is a Graphical Processing Unit (GPU) based search program which performs multiple de-dispersion trials and tophat matched filtering to identify dispersed signals in the data. GPUs are needed de-dispersion is extremely computationally intensive operation. Heimdall internally uses `dedisp` ([1]) for de-dispersion. Tophat matched filtering helps to boost signals. Lastly, normalization of the time series prior to searching regularizes the data. A simplified algorithm is shown here:

```
for dm in {2..1000}
do
for iwid in {1..6}
```

```
do
 de_disperse (dm)
 matched_filter (pow (2, iwid))
 normalize
 find peaks
endfor
endfor
```

Heimdall in reality is much more sophisticated and performs channel masking, and Radio Frequency Interference (RFI) excision. Interested readers are encouraged to read [2].

Heimdall reads from the DADA buffer to which coadder writes the coadded data stream. It reads a gulp size of data at a time (a parameter optimized, set to 24 seconds) and performs multiple DM-width trials, if the peaks in the resulting de-dispersed, matched-filtered time series data is significant and over a threshold, a candidate is registered at that particular DM ,width, and epoch. Heimdall sends out these candidates over to Python server running on the login node from where it is suitably acted upon (discussed in the following sections). Additionally, Heimdall also writes out the candidates to a candidate file. See subsection 2.7.1.

2.6 Trigger mechanics

Trigger is a terminology used to denote a candidate which qualifies certain selection rules and warrants follow up action. Trigger mechanics involve receiving triggers from Heimdall, applying selection logic, and performing follow up logic. There are two types of triggers:

1. Voltage trigger
2. DBSON trigger

Voltage trigger, as the name implies, triggers raw baseband voltage data. And, dbsontrigger generates a dbsonfile for every trigger. The course of action varies depending on the type of trigger. dbsontrigger is treated as the default. If a candidate is to be triggered, then it will *always* have a

Index	Name	Type	Comments
1	i0	double	UTC start epoch of signal
2	i1	double	UTC end epoch of signal
3	sn	float	Signal to Noise ratio of the signal
4	dm	float	Dispersion Measure of the signal
5	wd	float	Width of the signals in seconds
6	peak_time	float	Time since UTC epoch start of signal when signal peaks
7	meta	char[128]	Meta information

Table 2.2: Trigger struct definition. Signal here means the signal of interest. See text for more information.

dbsontrigger. Each trigger has its own distinct multicast group which makes distinction and follow up action simple and well separated.

The underlying which is passed around is the same for both the triggers. Table 2.2 shows the struct definition.

Voltage trigger handling is solely performed by the writer (see section 2.2). dbsontrigger response is much more sophisticated and will be dealt in the following sub-sections.

2.6.1 Trigger dispatch

Heimdall in addition to writing candidate files, also sends candidate data over to server which is called the trigger dispatch. The purpose of trigger dispatch is to receive candidates, apply various candidate selection logics, and multicasts triggers over to the compute nodes. Trigger dispatch is written purely in Python.

Each selection rule consists of cuts on S/N, DM, and practise, multiple rules are used simultaneously. Moreover, special notch filters to target pulses from pulsars are also applied. The type of selection rules used and the triggers received are discussed in great detail in chapter III. Hence, only a simple example of a rule is provided here:

$$\mathrm{S/N} \geq 8$$

$$\mathrm{DM} \geq 50\ \mathrm{pc/cc}$$

$$\mathrm{Width} \leq 100\ \mathrm{ms}$$

By default, trigger dispatch only sends out dbsontriggers. More stringent set of rules is applied in case of voltage triggers. The reason being SSD space required for voltages is much much more than that required for dbson s. A typical rule for a dbsontrigger to also be a voltage trigger is a simple cut as, S/N$\geq$ 25. Such a simple rule allows for any serendiptious voltage triggering on strong triggers.

Throttling Dispatching all the triggers received may sometimes overload the pipeline, cause lags, and may ultimately fail the pipeline. Hence, a suitable trigger throttling mechanism is put in place. Trigger rate may skyrocket for a multitude of reasons. It may be so whenever a bright radio pulsar in in the field-of-view (FOV) of VLITE-Fast . It may also be when Radio Frequency Interference (RFI) is strong, causing a large number of spurious triggers. Given the frequency band of operation ($320-384$ MHz) being not just close to walky-talkies used by on-site engineers but also being close to MUOS band, VLITE-Fast is more suspectible to RFI. Presence and effects of RFI is discussed more in detail in section 3.3. Here, only the instrumentation part of the throttling is described.

Heimdall performs searches in batches. All the candidates from a batch are sent at once after the batch has been processed. Throttling is done at a sub-batch level. A typical batch size is 30720 samples ($\sim$ 24 seconds) and the sub-batch is 8 seconds. As discussed in section 3.3, if a sub-batch sees more than 200 triggers, it is vetoed against not dispatched.

2.6.2 Trigger hook

Any type of follow-up analysis requires the filterbank data spanning for the duration of the trigger. The job of Trigger hook is to receive the dbsontrigger, slice the appropriate filterbank data, and write it along with trigger information into DADA buffers.

Input to trigger hook is the same DADA buffer used by Heimdall for its searching. Output of trigger hook are two separate DADA buffers, one for trigger information (header) and one for trigger data (data). DADA offers header block in a buffer but the number of headers is hardcoded to 8 which would severly reduce the number of triggers that can be held in buffer to 8. Hence, two separate

buffers are employed.

Trigger Hook reads the buffer and keeps track of the UTC epochs of the first sample in each data-block as it reads. Then, for a trigger received, with some pointer arithmetic, a slice of filterbank data is written to the data `DADA` buffer and trigger information is written to header `DADA` buffer.

Trigger hook also holds the option to write out a `fbson`. But, this option is not typically used since follow up action is performed anyway.

2.6.3 Trigger master

Reading from the buffer to which Trigger Hook has written the sliced filterbank along with trigger information, Trigger master performs the following:

- Generating:
  - De-dispersed filterbank binary JSON (`dbson`, see subsection 2.7.3)
  - Trigger plot (see subsection 2.7.4)
- Machine learning (ML) based classification of the trigger and baseband triggering.

The following only discuss the generation of `dbson` and trigger plot. The entirity of classification is discussed in great detail in chapter IV. Firstly, the bowtie plane is introduced. And then, the de-dispersed filterbank is described. These two are image planes which fully characterize any trigger. Infact, the ML strategy employed (chapter IV) uses these image planes as input.

The bowtie plane is computed using the Fast De-dispersion Measure Technique (FDMT, [26]). The low frequency introduces large delays which make the FDMT algorithm extremely slow. Hence, prior to FDMT, the data is first incoherently de-dispersed to first `DM`and then FDMT is applied.

2.6.4 Meta response

For every voltage trigger issued, a variety of meta data has to be packaged to do any kind of analysis with the voltage data. This job is performed by Meta response. Meta data involves the following:

1. Antenna mappings

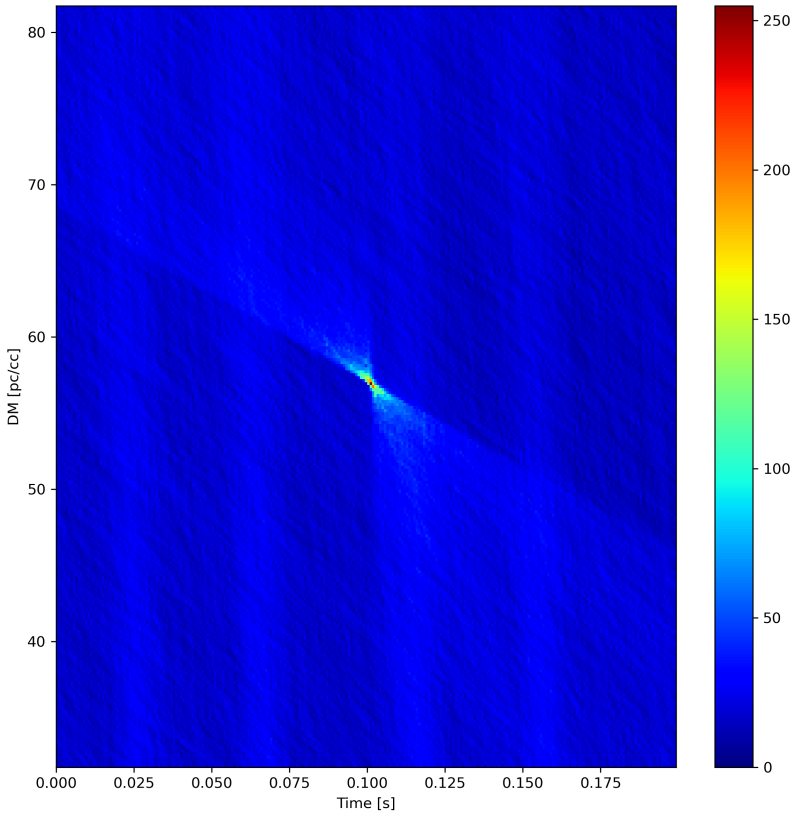


Figure 2.5: Bowtie plane. S/Nas a function of DMand time. It is called a bowtie plane since a true dispersed signal produces a bowtie. This is a trigger from Crab pulsar with S/N= 84.28 and DM= 56.75pc/cc. The data is digitized to uint8 hence takes integer values in $[0, 256)$.

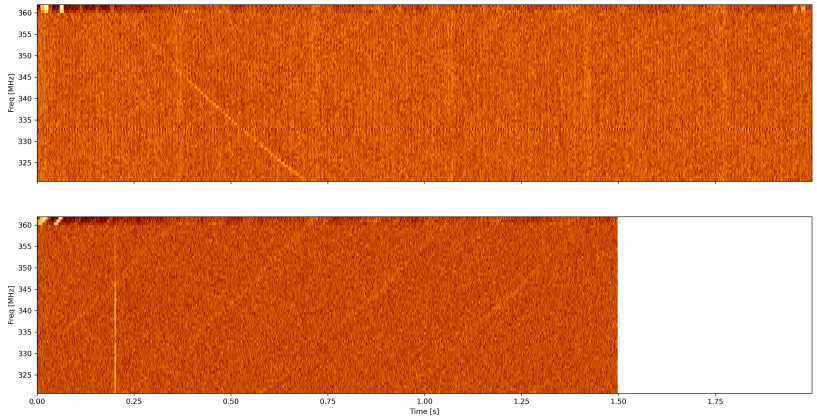


Figure 2.6: Top:Dispersed filterbank. Bottom:De-dispersed filterbank. Data is from a Crab pulsar trigger with S/N= 8.5 and DM= 56.75pc/cc.

2. Antenna delays

3. Antenna positions

4. Trigger parameters

This job is done by a Python server running on the login node. This server listens to the mcast group to which voltage triggers are issued. For a voltage trigger, it outputs a metar file as described in subsection 2.7.5, and subsection 2.7.5.

2.7 Data products

This section defines all the data products of the pipeline.

2.7.1 Candidate file

A candidate file consists of tab-separated values. Every new line has either two fields or nine fields. The definition of the file is given in subsection 2.7.1.

2.7.2 FilterBank jSON

FilterBank jSON (`fbson`) is an Universal Binary JSON (`http://ubjson.org/`) file format containing unprocessed filterbank along with header information. Filterbank data is in time major format with frequency index changing the fastest. The schema is defined in Table 2.7.1.

Type of data	Number of fields	Data
Pointing	2	Right Ascension in radians Declination in radians
Candidate	9	Signal to Noise ratio of the candidate Index of the first sample of the signal since the start of the observation Time of the first sample of signal Filterwidth of the signal Index of the DM trial DM of the signal Number of giants in the group UTC epoch of the start of the signal UTC epoch of the end of the signal

Table 2.3: Description of candidate file.

This data product is no longer in use since the trigger processing is currently done in realtime.

2.7.3 De-dispersed filterBank jSON

De-dispersed filterbank jSON (`dbson`) is an Universal Binary JSON (`http://ubjson.org/`) containing de-dispersed filterbank and bowtie plane digitized to `uint8`. The schema is defined in Table 2.7.1.

2.7.4 Trigger plot

Trigger plot is a diagnostic plot generated for every trigger. It is saved in Portable Networks Graphics (PNG, `https://tools.ietf.org/html/rfc2083`) format and rendered used PGPLOT `https://www.astro.caltech.edu/~tjp/pgplot/` library.

A trigger plot consists of bow-tie plane, de-dispersed filterbank, frequency averaged time profile, and `DM`profile. In addition to them, a trigger plot also shows S/N, width, pointing information, and time information. A typical trigger plot for a known pulsar trigger is given in subsection 2.7.4.

Type	Parameter	Present	Comments
Time	S/N	Both	Signal to Noise ratio of the trigger
	DM	Both	Dispersion Measure of the trigger
	Width	Both	Width of the trigger in seconds
	Peak time	Both	Peak time of the trigger from the start of the data
	Tstart	Both	MJD of the start of the data
	Tsamp	Both	Sampling time of the data
	Duration	Both	Duration of the data
Frequency	Fch1	Both	Frequency of the first channel in MHz
	Foff	Both	Frequency width in MHz
	Nchans	Both	Number of channels in the data
Indices	I0	Both	UTC epoch of the first sample of the data
	I1	Both	UTC epoch of the last sample of the data
	Epoch	Both	UTC epoch of the start of the observation
	Nsamps	Both	Size of the data
Parameters	Nbits	Both	Number of bits per datum
	Antenna	Both	Station ID
	Source name	Both	Name of the source observing when trigger was recorded
	RA	Both	Right Ascension of the source in radians
	Dec	Both	Declination of the source in radians
	Group	Both	String identifier
DMs	DM1	DBSON	Start DM in pc/cc in bowtie plane
	DMoff	DBSON	DM width in bowtie plane
	NDM	DBSON	Number of DM trials in bowtie plane
Data	FB	FBSON	Raw filterbank
	DD	DBSON	De-dispersed filterbank
	BT	DBSON	Bow-tie plane

Table 2.4: Description of `fbson` and `dbson`.

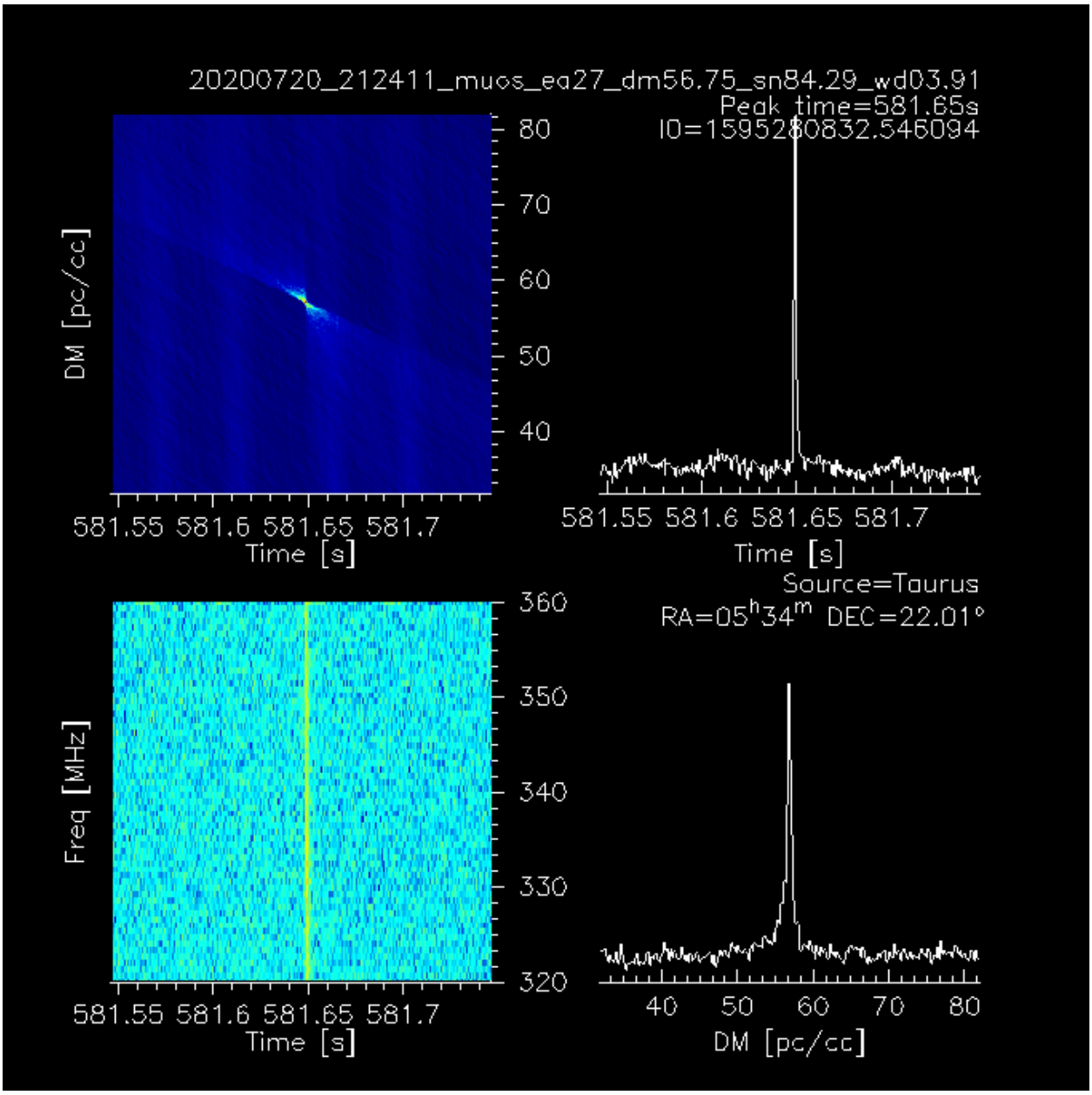


Figure 2.7: A trigger plot generated on a trigger realtime. This trigger is from the Crab pulsar. Every trigger is processed and a trigger dump (`dbson`) and trigger plot (shown here) are generated.

Type	Parameter	Comments
Header	S/N	Signal to Noise ratio of the trigger
	DM	Dispersion Measure in pc/cc of the trigger
	Width	Width of the trigger in seconds
	T0	UTC epoch of the start of the signal
	T1	UTC epoch of the end of the signal
Delays	VLITE_ANT_ID	VLITE Antenna ID
	VLA Antenna	VLA Antenna
	DIFX_HOST	Compute node connected to the antenna
	DIFX_IFACE	Network interface on the compute node
	CLK_OFFSET	Clock offset in nanoseconds
	PAD	Antenna string identifier
	LO_FIBER	RF-over-Fiber delay in nanoseconds
	ENABLE	If antenna is added to the array
Antprop	CONFIG	Array configuration identifier
	DATASETID	Dataset identifier
	CREATION	Epoch of creation
	EOPSET	Earth Orientation Parameter set. TAI_UTC, UT1_UTC and pole information.
	dots	For each EOPDAY
	ANTS	Antenna information. WIDAR_ID, PAD, X, Y, Z, OFFSET
	dots	For each antenna

Table 2.5: Schema for Meta dumped for every voltage trigger.

2.7.5 META Voltages

Meta (`meta`) is an Universal Binary JSON (`http://ubjson.org/`) containing meta information required for any voltage analysis.

2.8 Codebases

There are two pipeline codebases. Writer and process-baseband wholly contributed by Dr. Matthew Kerr are found in vlite-fast (`https://github.com/kerrm/vlite-fast`). All the other codes required for the pipeline (written by me) can be found in `asgard` (`https://github.com/shiningsurya/asgard`).

`asgard` is the pipeline code written by me which makes `VLITE-Fast` possible. It is written in

Language	LOC
C++	10 000
Python	3 600
BASH	300

Table 2.6: Lines of code (LOC) written in languages part of `asgard`.

C++, Python and BASH. The breakdown of the lines of code (LOC) written are given in section 2.8.

CHAPTER III

VLITE-FAST PATHFINDER SURVEY

This chapter summaries the entirity of the data collected in numerous runs of the `VLITE-Fast` system, and follows it up careful statistical analysis. All triggers from known sources are summarized. The whole data is bundled together as the VLITE-Fast Pathfinder Survey (`VFPS`).

3.1 Campaign runs

`VFPS` consists of data collected over multiple settings. Each campaign run is characterized by the bits of digitization used for filterbank data used and the trigger cuts used. This characterization breaks the whole of `VFPS` into multiple campaigns. Firstly, all the campaign runs are enumerated and described in brief before delving into the details in respective sub-sections.

NBIT = 2 (NB2) This was the first run of `VFPS`system. All the filterbank data was digitized to 2 bits to test the computational capabilities of the pipeline. Conservative trigger cuts were employed for the same reason.

NBIT = 8 (NB8) The filterbank data bit depth was increased to 8 in this run. The trigger cuts were somewhat more agressive than before.

Max Warp (MW) The filterbank was digitized to 8 bits same as before. The trigger cuts were drastically changed to be the most aggressive cuts possible.

The key features are summarized in section 3.1.

3.1.1 NB2

In an epoch from `2019-10-17` to `2019-12-05` spanning for 49 days, `VLITE-Fast` was onsky for 27.03 days. This resulted in capturing 10 306 triggers, yielding a trigger rate of ~ 16 hr^{-1}.

Label	Start (YYYY-MM-DD)	End (YYYY-MM-DD)	On-sky (day)	Uptime (%)	Triggers	Rate (/hr)	Comments
NB2	2019-10-17	2019-12-05	27.03	55.61	10 306	15.89	Two bit digitization
NB8	2019-12-19	2020-01-21	12.16	36.10	12 028	41.19	Eight bit digitization
MW	2020-01-22	2020-06-27	87.16	55.32	826 294	394.85	Eight bit digitization
ALL	2019-10-17	2020-06-27	126.38	49.79	848 628	279.77	All the campaigns

Table 3.1: Salient features of the campaign runs.

The uptime achieved in this was 55.6%. The main features of this run was that the data was digitized to NBIT2 hence the name. The trigger cuts were humble:

$$\mathrm{S/N} \geq 8$$

$$\mathrm{DM} \geq 50 \text{ pc/cc}$$

$$\mathrm{Width} \leq 100 \text{ ms}$$

This run didn't have the `dbson` trigger mechanics in place. Instead, for every trigger, an `fbson` file was written to disks at all the antennas including the coadded antenna as well.

3.1.2 NB8

In an epoch from from `2019-12-19` to `2020-01-21`spanning 33 days, `VLITE-Fast` was onsky for 12.16 days. During which, `VLITE-Fast` collected 12 028 triggers, yielding an event rate of ~ 41 hr^{-1}. Fraction of total time spent on sky was 36.10%. The trigger cuts were a bit more agressive than NB2 run.

$$\mathrm{S/N} \geq 7.5$$

$$\mathrm{DM} \geq 50 \text{ pc/cc}$$

$$\mathrm{Width} \leq 100 \text{ ms}$$

PSRJ	RAJ (hh:mm:ss)	DECJ (dd:mm:ss)	DM (pc/cc)	P0 (s)	W50 (ms)	N	S400 (mJy)	Time on source (hr)
J1752-2806	17:52:58.6	-28:06:37.3	50.37	0.562	6.1	2196	1100.0	2.00
J0534+2200	05:34:31.9	+22:00:52.0	56.77	0.033	3.0	24	550.0	0.84
J0742-2822	07:42:49.0	-28:22:43.7	73.73	0.166	4.200	90	296.00	
J1745-3040	17:45:56.3	-30:40:22.9	88.37	0.367	6.1	29	66.0	23.72
J2321+6024	23:21:55.2	+60:24:31	94.59	2.256	131.1	4	36.0	0.14
J1935+1616	19:35:47.8	+16:16:39.9	158.52	0.358	6.0	34	242.0	1.72
J1922+2110	19:22:53.5	+21:10:42	217.09	1.077	14.8	10	30.0	9.70

Table 3.2: Observed pulsars

3.1.3 MW

This is the longest campaign run in the VFPS. Starting from 2020-01-22 to 2020-06-27, in a total length of 157 days, VLITE-Fast achieved 55.32 uptime by being on sky for $\sim$ 87 days. Two separate trigger cuts were employed to keep trigger rates in check. The trigger cuts were as aggressive as possible, hence, collecting 826 294 triggers with trigger rate of ~ 395 hr^{-1}.

$$\begin{aligned} \mathrm{S/N} \geq 6.0 \& \mathrm{S/N} &\geq 8.0 \\ \mathrm{DM} \geq 50 \text{ pc/cc} \& \mathrm{DM} &\geq 50 \text{ pc/cc} \\ \mathrm{Width} \leq 100 \text{ ms} \& \mathrm{Width} &\in [20, 100] \text{ ms} \end{aligned}$$

3.2 Detection of pulsars

A large onsky time yields many serendiptious triggers caused by pulses from pulsars. These detections are a field test for VLITE-Fast and are tabulated in section 3.2. Based on these detections, a rudimentary argument for Field of View (FOV) (subsection 3.2.1) and sensitivity (subsection 3.2.2) are formulated.

3.2.1 Field of View

It is worthwhile to understand the spatial extent over which the PSRJ̈1752-2806 (from now on dm50) has been detected. This exercise would help us visualize how large of a field-of-view

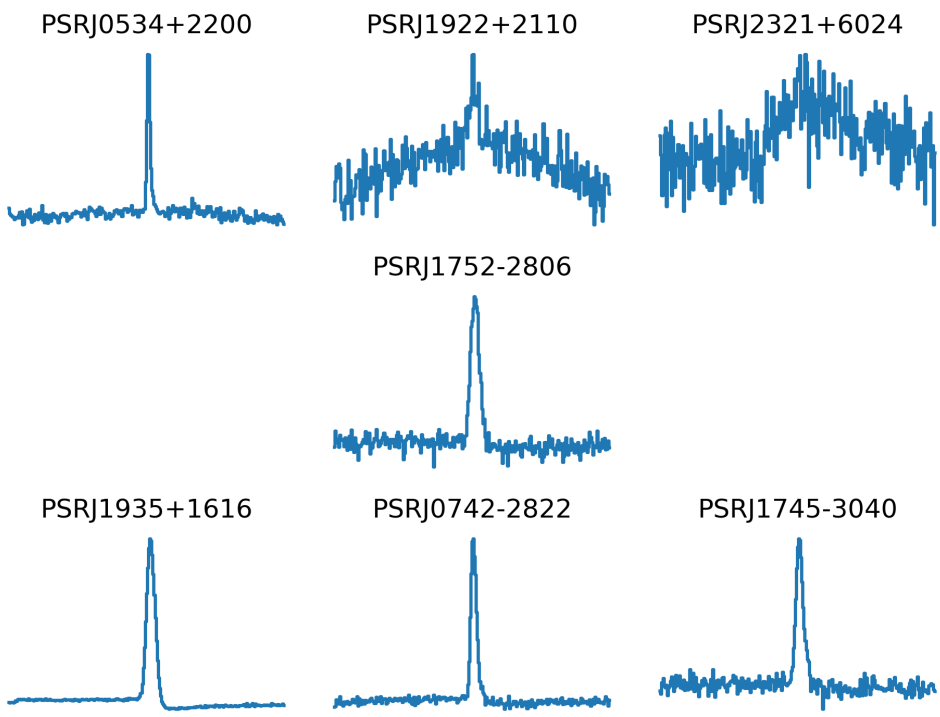


Figure 3.1: Collage of averaged pulses from the detected pulsars.

VLITE-Fast posses. Angular resolution on the basis of a diffraction limit arguments is $\sim 2^o$ (c.f. subsection 1.3.2). See subsection 3.2.1.

3.2.2 Sensitivity

The sensitivity of the VLITE-Fast can be understood using the detected set of pulsars. Since, each pulsar has a documented flux density at 400 MHz, available in PSRCAT([13]). With the help of number of pulses detected, the sensitivity of VLITE-Fast can be loosely extrapolated on the basis of the pulsars.

3.3 RFI

One of the major challenges any radio observatory faces is that of Radio Frequency Interference (RFI). These are spurious radio signals of human origin polluting the frequency band of interest. In a search pipeline such as VLITE-Fast , RFI causes large number of triggers which tax the pipeline. One of the main reasons for VLITE-Fast pipeline failing from time to time is the lag caused by serving many spurious triggers because of RFI. Naturally, with the triggers collected so

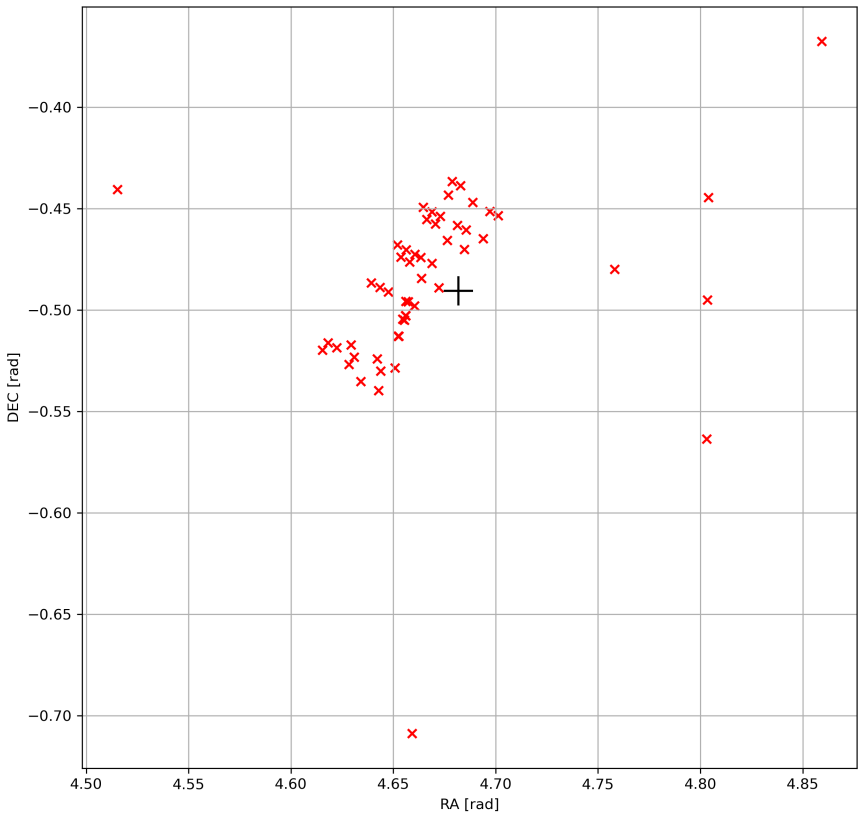


Figure 3.2: Large FOV of VLITE-Fast understood using PSRJ̈1752-2806 as a marker. Crosses represent the pointings where triggers from the pulsars were recorded. Color is coded to represent the number of triggers detected from each pointings.

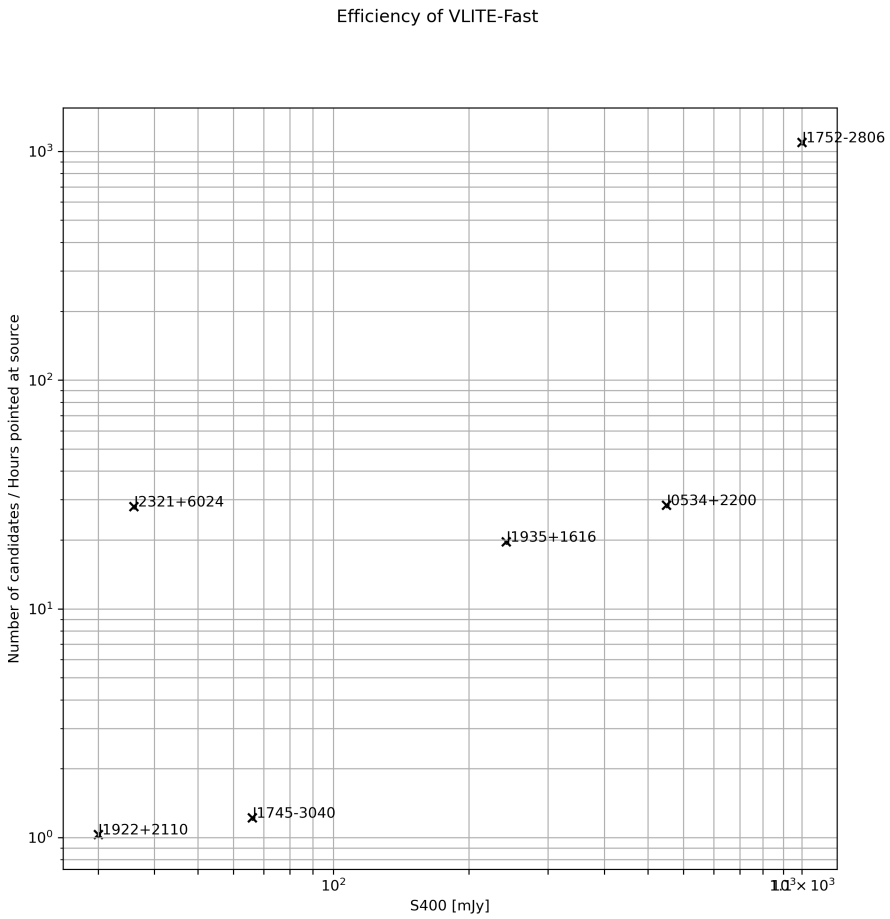


Figure 3.3: Sensitivity of `VLITE-Fast` using triggers from pulsars and documented flux density. Flux density is plotted on x-axis. Triggers received divided by time spent on sky is plotted on the y-axis.

far, a strategy can be devised to understand the triggers caused by RFI and mitigate them realtime. This is described in subsection 3.3.1. also observed recurring RFI which produces a distinct feature in the DM distribution. This artifact is discussed in subsection 3.3.2.

3.3.1 RFI contamination

Detection of a true signal (of astrophysical origin) is a rare event. Registering a large number of triggers in a short time is indicative of RFI. This fact is used to measure the amount of RFI contamination in the data. All the triggers are collected in 8-second batches and counted. If a given batch has a large number of triggers in it, it is treated as RFI. Figuring out the correct *large* number of triggers is done heuristically.

Firstly, the cumulative distribution of the number of triggers in a batch (of 8 seconds) is computed. See subsection 3.3.1. Understanding that RFI would only be contaminating a small percent of the whole data. One would expect the cumulative distribution to flatten out for larger number of triggers in a batch. Another way to interpret the same would be that only a small portion of the dataset have extremely high trigger activity. A suitable threshold is thus selected which limits the maximum number of triggers in a batch of 8 seconds but retains most of the triggers. Mathematically, this threshold which be an inflection point of the cumulative distribution function.

Based on the cumulative distribution, $95, 99\%$ percentiles turn out to be 1.125 triggers s^{-1} and 2.75 triggers s^{-1}. This is shown as the black dotted line and red solid line in the plot (subsection 3.3.1).

3.3.2 DM=150 pc/cc artifact

Short time RFI in a single frequency channel does not produce spurious triggers since bandpass normalization on the time windown containing the RFI cancels its intensity. However, when there is short time RFI in separate channels, it is a different story. If such a short time RFI is coincident in time, any search pipeline would register it as a DM=0 pc/cc signal and can be filtered out easily. But, if the same RFI has a time offset between the frequency channels, it causes the same search pipeline to register spurious triggers at that DM which culls the time offset and hence

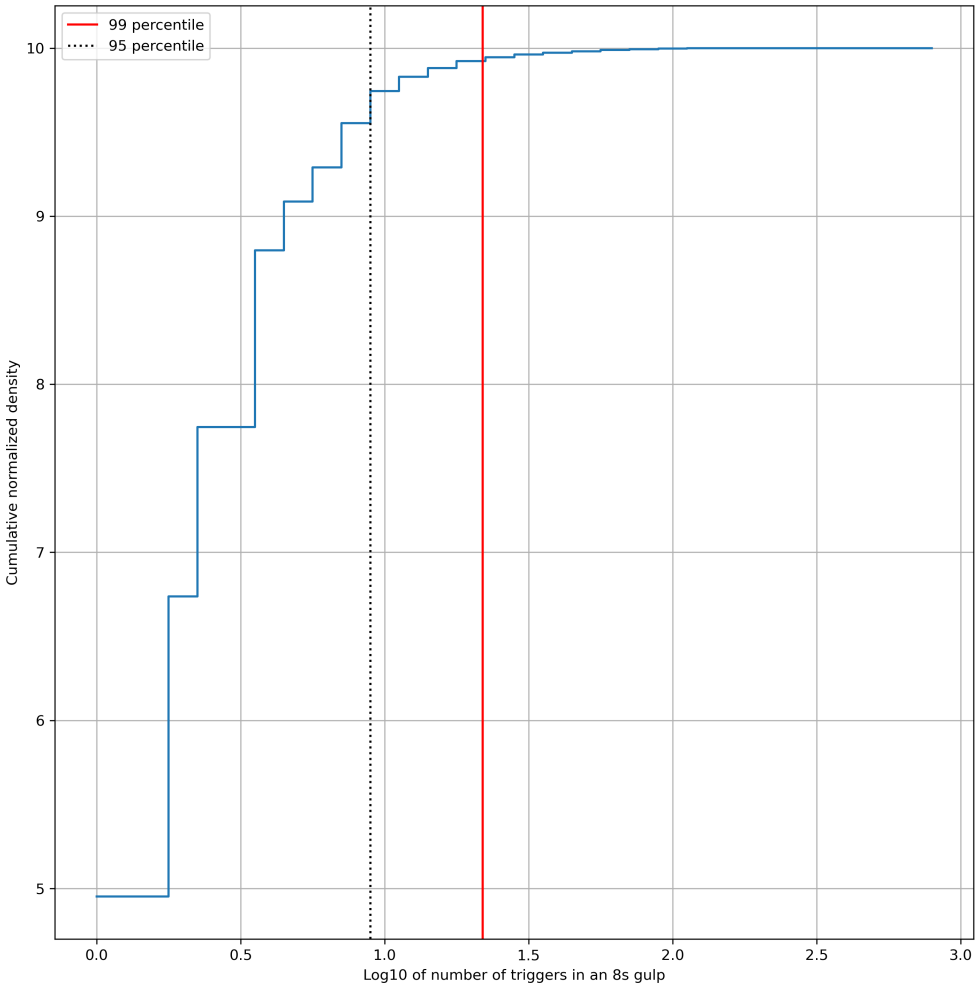


Figure 3.4: Cumulative distribution of number of candidates received in an 8-second gulp. The lines correspond to $95, 99\%$ percentiles which mean 1.125 triggers s^{-1} and 2.75 triggers s^{-1}.

are much more difficult to excise.

A short time RFI exisiting only in specific frequency channels having time offset between the channels behaves like a dispersed signal. And a candidate is registered for that DMwhich aligns the time offset. This effect is observed in VFPSwhich causes triggers at a range of DM s centered around DM=150 pc/cc, hence the name. See the de-dispersed filterbank and frequency averaged profile in subsection 3.3.2.

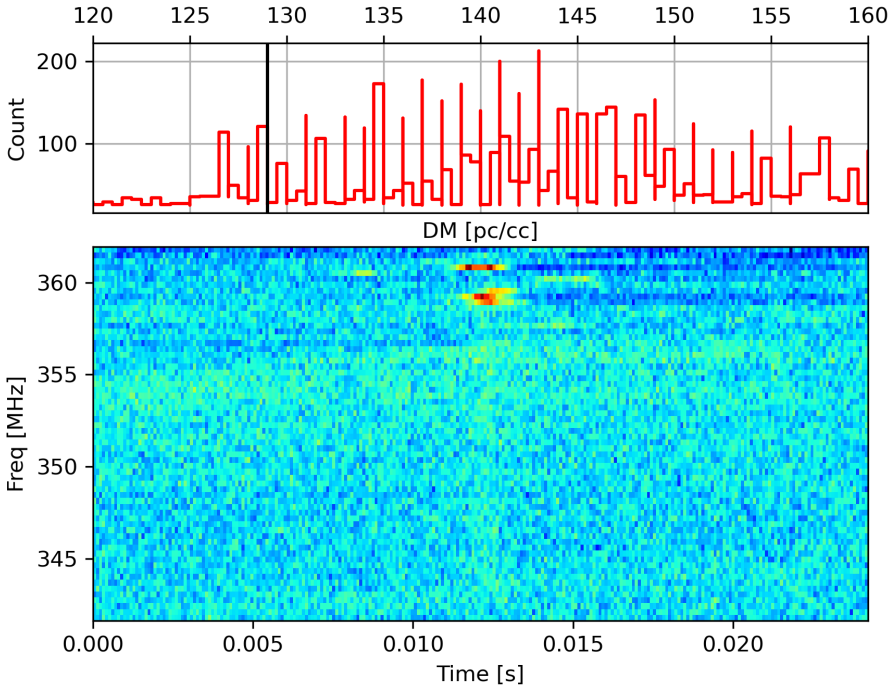
3.4 Heimdall triband structure

A complete statistical analysis of all the triggers also uncovered the extent of capability of Heimdall in its DM-width trials. See Table 3.4 which is expected to be uniform in the DM-width parameter space. However, for large DM s, Heimdall employs a time averaging window (known as tscrunching) which reduces the sensitivity in the width. This is captured by the quantization seen in the width space for large DM.

Heimdall performs the tscrunching operation by default. This operation is a feature of the underlying de-dispersion code called dedisp([1]) which is used by Heimdall. For large DM, the in-channel smearing (introduced in subsubsection 1.1.2) is very high and at times exceeds the sampling time. In such a case, time averaging is performed which increases the sampling rate and the in-channel smearing is kept less than the sampling time. If the in channel smearing is much more than the sampling time, the signal is lost. Any amount of de-dispersion would not bring out the signal.

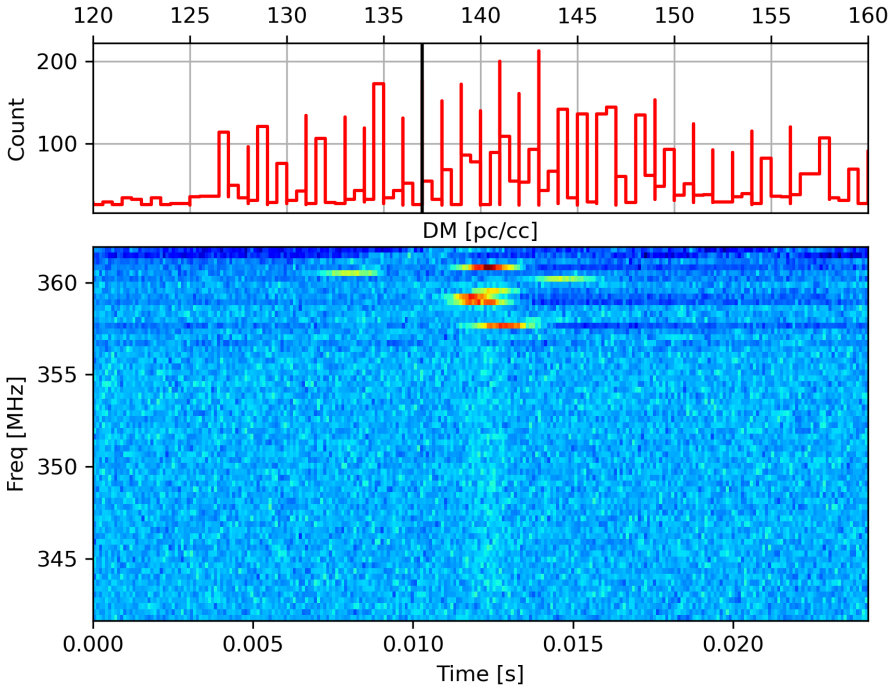
In all the runs so far, the adaptive_dt was enabled. An effect of this feature is that for weak S/N, large DMsignals if the width is not near any of the quantized widths, the matched filtering would not boost the signal causing it to be passed as non-detection.

An artifact of this is seen in excess triggers registered at specific DM s beyond which tscrunching is performed prior to searching. These DM s are found to be DM=347.165 pc/cc and DM=790.695 pc/cc for which the in-channel smearings at highest, lowest and central frequencies are tabulated at section 3.4.



0.45

Figure 3.5: Solid black line represents DM=129 pc/cc.



0.45

Figure 3.6: Solid black line represents DM=137 pc/cc.

Figure 3.7: Top: Distribution of triggers contaminated with DM150 RFI. Bottom: De-dispersed filterbank showing the narrow band RFI.

Table 3.3: In-channel smearing at DM s where tscrunching is activated for the lowest and highest frequency channels. The sampling time is 781.25 μs and frequency channel width is 655.255kHz.

DM (pc/cc)	Frequency (MHz)	Smearing (ms)	Time units
347.165	361.941	19.9	25
	340.973	23.8	30
	320	28.8	36
790.695	361.941	45.3	58
	340.973	54.2	69
	320	65.6	84

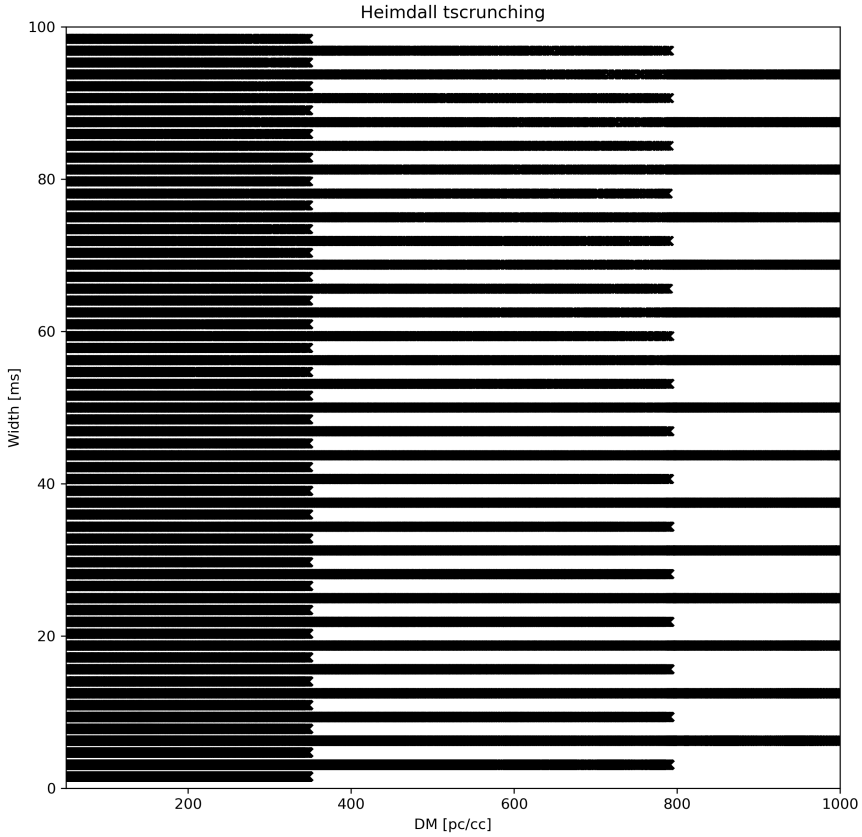


Figure 3.8: DM-Width space of all the Heimdall triggers. Although the maximum width is set to 100 ms, this plot only extends until 20 ms. This space is expected to be uniform. The quantization seen in the width for large DMis an affect of heimdall *adaptive_dt*.

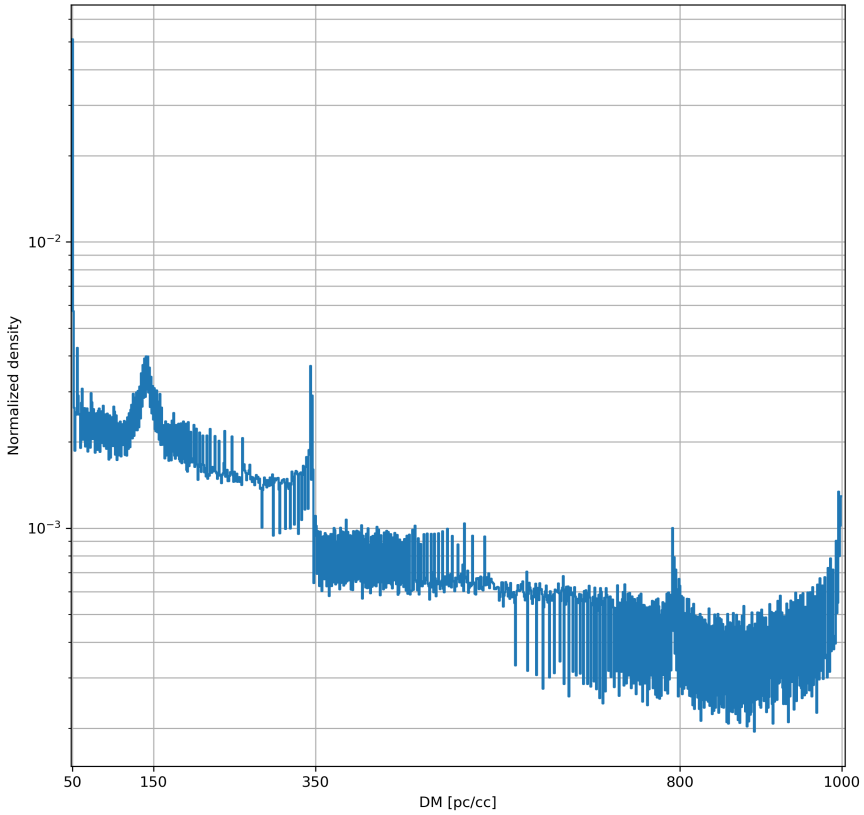


Figure 3.9: Histogram of DM of all the triggers recorded. The sharp line at DM=50 pc/cc is due to the PSRJ̈1752-2806. The sharp lines at DM=350 pc/cc, DM=800 pc/cc and DM=1000 pc/cc are due to heimdall triband structure. See section 3.4. The very broad peak at DM=150 pc/cc is due to narrow band RFI. See subsection 3.3.2.

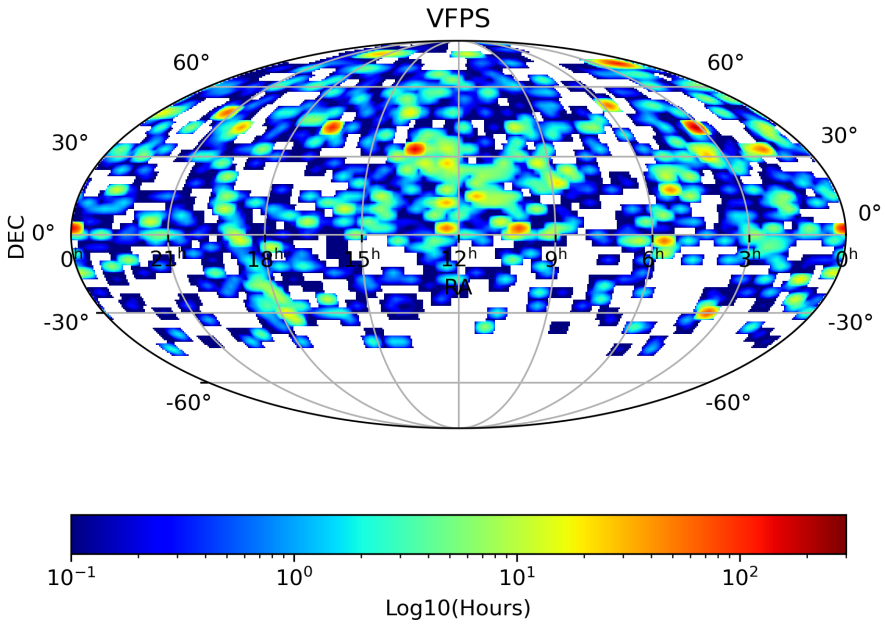


Figure 3.10: Skymap of time spent on pointing in hours. Axes are Right Ascension/Declination.

3.5 Summary statistics

For a total uptime of $\sim$ 126 days, the distribution of time spent on each pointing in shown in section 3.5. The trigger rates are tabulated in Figure 3.5. Since the trigger logic was significantly changed among campaigns, the trigger rates are only shown for MW campaign.

S/N cut	Trigger rate (/hr)	S/N cut	Trigger rate (/hr)
6	394.85	7.5	50.28
6.5	165.10	8	33.9
7	85.92	10	13.58

Table 3.4: Trigger rates for different S/N cuts in real data. Since the trigger logic was different in all the campaigns. These trigger rates are computed only from the MW campaign where the S/N cut was the least possible. There are 826 924 triggers in this campaign. See text for more information.

S/N cut	Trigger rate (/hr)	S/N cut	Trigger rate (/hr)
6	117.2	7.5	3
6.5	23.1	8	2
7	6.5	10	0.12

Table 3.5: Trigger rates for different S/N cuts. Notice the extremely steep decline in trigger rate at low S/N. This shows the extent of low S/N being whitenoise triggers. See text for more information.

3.6 Simulation runs

In an effort to better understand the capabilities of `VLITE-Fast` , data was taken in controlled enviroments. Those controlled environments and the results of the data taking are surmised here.

3.6.1 Pure noise

In this, `VLITE-Fast` data coming from the antennas was discarded and replaced with Gaussian random noise with mean 0 and standard deviation 33.313. Whatever triggers registered in such a run are purely noise triggers. This simulation was run for $\sim$ 72 hours collecting $\sim$ 10 000 triggers.

The distribution of the S/Nregistered is in subsection 3.6.1. The rates are tabulated for different S/Ncuts in Figure 3.6.1.

This exercise proved fruitful in understanding how the pipeline responds to pure noise. Pure white noise has no signal content whatsoever. These triggers are result of pure noise data that looks exactly like a real single. This is picked up by the pipeline. There is no technique, or procedure that can be applied to alleviate such triggers. Hence, they are only argued upon in a statistical sense. See Figure 3.6.1. The S/Ncut in the MW run was 6 which yielded a trigger rate of about 395 hr^{-1}. This exercise shows that about 58% of the triggers are due to pure random noise which is a lot.

3.6.2 Injected

A dispersed signal of known `DM`, amplitude, and is added on top of random noise, and sent through the pipeline. The signal is then recovered from the triggers collected. This exercise shows how receptible `VLITE-Fast` pipeline is. Since the known signal is embedded on top of random

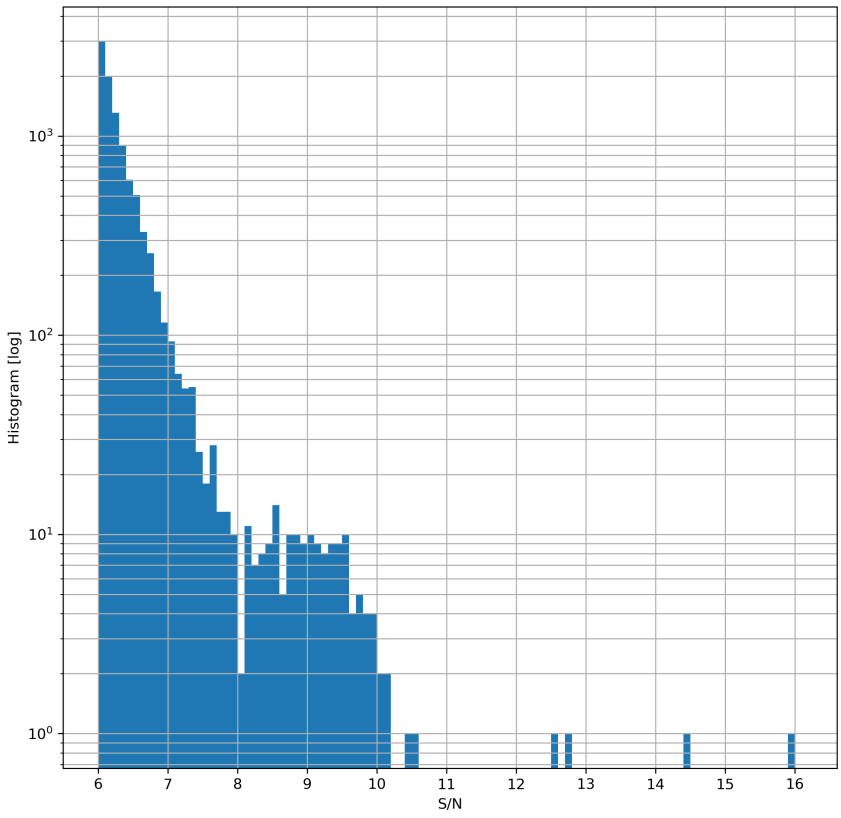


Figure 3.11: Trigger rates observed at different S/Nwhen only sending random noise through the pipeline. This figure shows how noise dominated lower S/Nis. For low S/N(6|8), the relation is exponential. The valley seen at 8 is result of changing trigger cuts used in dispatch in the MW campaign (See section 3.1).

noise, there are also many noise triggers which are registered. Such triggers are later filtered by comparing the time of injection with the time of trigger's peak.

This simulation run was designed to inject 15 FRBs in 2-minutes, yielding a trigger rate of 450 hr^{-1}. Due to the Heimdall tri-band structure (see section 3.4), which makes the former less suspectible to high DM triggers for widths far from the quantized width, the trigger rate captured was 308 hr^{-1}. Fig. 3.6.2 shows four subplots. The triband structure is plotted in top left. The rest capture the distributions of registered S/N, DM and widths.

The decreasing density for large S/N,DM is because of Heimdall triband structure. Due to the quantization in the width for large DM, many high DM signals are not registered hence, a downward trend is observed in S/N, DM.

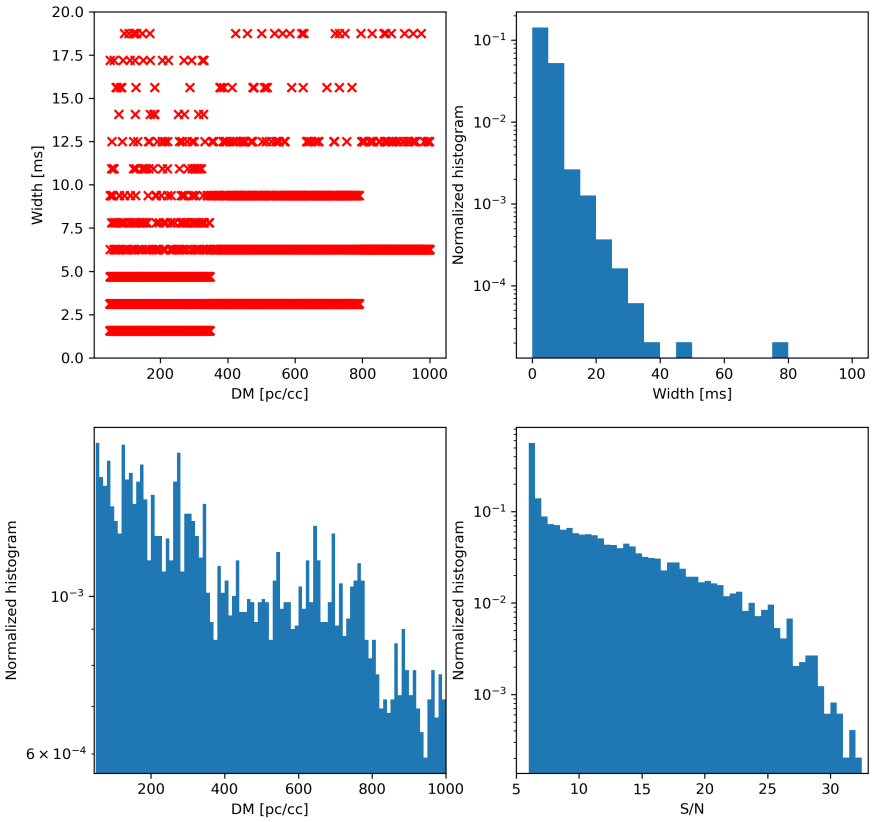


Figure 3.12: Top left: DM v/s showing quantization in at high DM consistent with triband structure (see text in section 3.4) Top right: Distribution of registered width. Bottom left: Distribution of registered DM. Bottom right: Distribution of registered S/N. Injections parameters are derived from a uniform distribution and the same is expected in the registered S/N, DMand widths. However, due to the triband structure, it is not achieved. The drop in density at high S/Nis attributed to some percentage of injected triggers (high DM) are failed to register.

CHAPTER IV

MACHINE LEARNINGS

This chapter describes the Machine Learning (ML) / Artificial Intelligence (AI) system in the VLITE-Fast which identifies triggers worthy of a closer look. First and foremost, the motivation is discussed in section 4.1. The ML model is introduced and described in section 4.2. Lastly, the newly *learnt* ML model is applied to the whole of VFPS and results are examined in section 4.3.

4.1 Coherent analysis

VLITE-Fast operates on an *incoherent* level. In all of the pipeline, only the power is considered. There is no phase considered anywhere. Hence the name, *incoherent*. This lack of phase greatly simplifies the pipeline design but comes at a cost.

For an array of N antennas, co-addition in *powers (incoherently)* (see subsection 2.4.1) only boosts the signal (or specifically, the signal-to-noise, S/N) by $\sqrt{N}$. A *coherent* analysis would boost the S/N by N. The reason being the phase information.

In addition to the S/N boost, a *coherent* analysis would also provide good localization capability. Given a set of relative phases, one can translate to geometric delays and from there geometric path differences from asrophysical sources. This ability is of great value since FRBs are known to be originating from outside the galaxy by which small angular variations could lead to astronomically large separated distances due to large radii.

4.1.1 Coherent analysis with VLITE-Fast

The cost of doing a *coherent* analysis real time is prohibitive. In order to be able to do *coherent* analysis, voltage data has to be recorded. A second of raw voltage data consisting of two polarization measures about 250 MB. A typical VLITE-Fast compute node has 450 GB of

dedicated Solid State Disk (SSD). Meaning, a `VLITE-Fast` compute can only have 1800 s (or 30 minutes) of raw voltage data before getting filled. Given the extremely large volume of voltage data, it is impractical to record voltages all the time. Hence, `VLITE-Fast` performs searches *incoherently* and triggers voltages for a coherent follow-up analysis.

It is also not practical to trigger voltages on all the triggers. It would be so as if voltages are recorded all the time since the trigger rates are high. So only a subset of the triggers received are to be allowed to trigger voltages. Naturally, such a subset has to be selected on the basis of the signal's merit of being a real signal of astrophysical origin. The question of how to decide what triggers to trigger on is the main goal of this chapter.

4.2 Machine Learning

The objective of the AI is to given a trigger, identify if a following *coherent* analysis has to be initiated. In an ideal situation, it would be desirous to have an AI solution which can identify all the true signals (i.e., the signals of interest). However, training such an AI solution is extremely difficult. The signals of interest are rare. Hence, any dataset produced would posses this assymmetric, which then would make training difficult. Hence, this approach is abandoned. Succiently, the mission of the AI is to select those triggers which are worthy of a second closer look.

Any AI solution is only as good as the data used for training it. Keeping this in mind, the dataset is carefully created, which is discussed in subsection 4.2.1. The model and training procedure used is described in subsection 4.2.2. The results from the training are showcased in subsection 4.2.3.

4.2.1 Dataset

The problem at hand is a *binary classification* problem. Every input is to be mapped to one of the two classes, either the `TRUE` class or the `FALSE` class. Hence, the dataset used for training is so judiciously chosen such that it contains good representation of both the classes.

Triggers from known radio pulsars with S/N$\geq$ 7 constitute a part of the `TRUE` class. All the injected and received triggers also constitute the `TRUE` class. RFI is manually selected using the

Dataset	Class	Number
VFPS	-	818 848
Injected	True	8 480
Pulsars	True	23 574
TRUE	-	32 054
RFI	False	8 207
DM150 RFI	False	20 660
FALSE	-	28 867
Unseen	True	1 716
Unseen	False	1 700

Table 4.1: The breakdown of `VFPS`dataset used for ML. See text.

creiterion described in subsection 3.3.1. The `DM150 RFI` is also selected and both form the `FALSE` class. The actual breakdown of the cardinalities are given in subsection 4.2.1.

A separate dataset is chalked out and treated as *unseen* dataset. This dataset will not be used in training or validation but will be used to see how the model performs on data which is not seen by it in any way.

4.2.2 Model and training

A typical Convolutional Neural Network (CNN) is used here. The design of the CNN is depicted in subsection 4.2.2. The non-linear activation used is the leaky-rectifier ([24]). To avoid overfitting dropouts ([7, 23]) are used after every layer. To alleviate the covariant shift seen in deep layers, a batchnormalization layer is also added ([8]) after certain layers.

The input of the CNN is two 32×32 image plane of de-dispersed filterbank and bowtie plane. The output is a two element vector showing the probabilities of the input belonging to both the classes. A `dbson`actually holds bowtie image of size 256×256 and de-dispersed filterbank of 64×256. These are reduced by block-means to same size of 32×32 and brought to $[0, 1]$ by dividing by 255.

Training was done in minibatches of 10. Two data augmentation transforms were employed:

1. Frequency axis flip of de-dispersed filterbank.

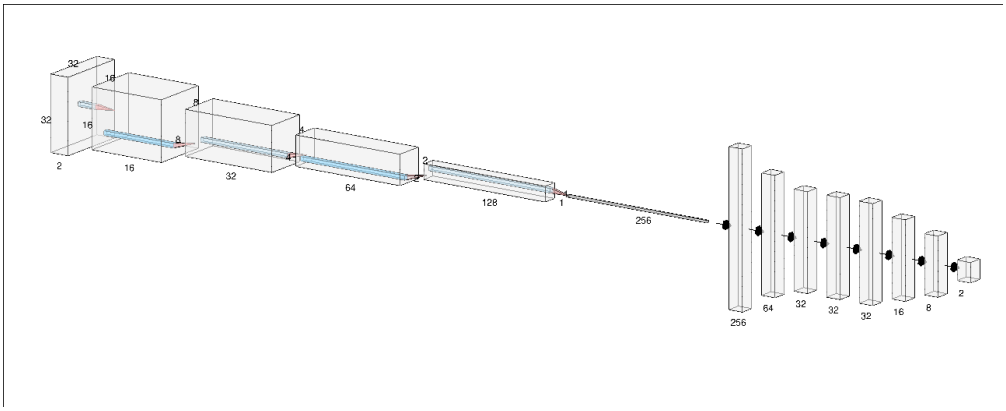


Figure 4.1: CNN used in the AI. The numbers adjacent to the block represent the dimension of the block. Every block is followed bya dropout ([23, 7]) and leaky-ReLU ([24]). In addition, blocks with channels $32, 128$ also have a batchnormalization ([8]). See text for model details. See `https://github.com/shiningsurya/trishul/tree/master/Python` for code.

2. Time, `DM`axes flip of bowtie plane.

It is ensured that these two transformations don't cause the model to learn spurious features. Adam optimizer ([10]) was used for optimizing. A step learning rate was employed. The loss function was CrossEntropyLoss since this is a classification problem. The script which does the training is `https://github.com/shiningsurya/trishul/blob/master/Python/tscae_ig`. Interested readers are encouraged to read the code.

4.2.3 Training results

The main training results are tabulated in Table 4.2.3. The learning curves are plotted in Table 4.2.3. The confusion matrices are tabulated in Table 4.2.3.

The metrics used to evaluate the performance are listed below. These metrics are computed from the confusion matrix. A confusion matrix is a matrix showing how the AI has done the classification. A member of any class (`TRUE` or `FALSE`) can be classified as any other class by the AI. This breakdown is tabulated in confusion matrix. A typical confusion matrix is shown in subsection 4.2.3.

A binary classification problem has two classes (`TRUE`, `FALSE`). Hence, a total of four possibilities can arise:

	TRUE	FALSE
TRUE	TP	FP
FALSE	FN	TN

Table 4.2: A binary classification confusion matrix. Each row corresponds to what the AI thought the class would be. Each column is the ground truth. If a candidate actually belongs to `FALSE` class (second column), if AI labels it as `TRUE`, it is treated as False Positive (FP). See text for more information.

True Positive (TP) When the AI classifies a `TRUE` class member correctly as a `TRUE` member.

False Positive (FP) When the AI classifies a `FALSE` class member incorrectly as a `TRUE` member.

It is *falsely* labeled as *positive*.

True Negative (TN) When the AI classifies a `FALSE` class member correctly as a `FALSE` member.

False Negative (FN) When the AI classifies a `TRUE` class member incorrectly as a `FALSE` member.

It is *falsely* labeled as *negative*.

Using these four definitions, one computes various point statistics to measure the perforamce.

Recall shows the ability of AI to recover all the `TRUE` cases. Mathematically, it is written as $\frac{\text{TP}}{\text{TP+FN}}$.

Precision shows the ability of AI to identify the `FALSE` cases. Mathematically, it is written as $\frac{\text{TP}}{\text{TP+FP}}$.

Accuracy measures the gross performance of AI to classify both the cases. Mathematically, it is written as $\frac{\text{TP+TN}}{\text{TN+FP+FN+TP}}$.

False Positive Rate measures the fraction of false positives over all the `FALSE` class. Mathematically, it is written as $\frac{\text{FP}}{\text{FP+TN}}$.

4.3 VFPS ML inferences

Having developed an ML solution trained on a very small subset, the AI is now run on the entire `VFPS`dataset. Triggers selected by AI (henceforth just called selected triggers) are analyzed and reported.

Datasets	Recall	Precision	Accuracy	FPR
Training	94.92	99.29	97.15	0.006
Validation	94.60	99.24	97.01	0.006
Unseen	73.54	99.14	86.38	0.006

Table 4.3: Training results shown in various metrics computed. See text for metrics definitions.

Training		Validation		Unseen	
22 933	160	5 735	39	1 689	11
1 125	21 017	299	5 236	454	1 262

Table 4.4: Confusion matrices for different datasets. See text for more information.

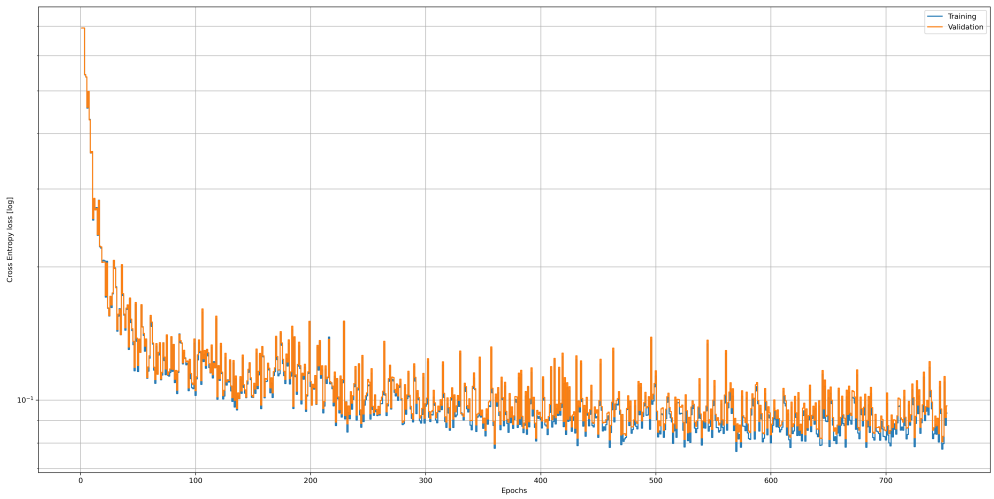


Figure 4.2: Running for 753 epochs. Model still can run for more epochs. GPU-based run done on Azure cloud took around a minute per epoch. This is a run lasting $\sim 12^h$.

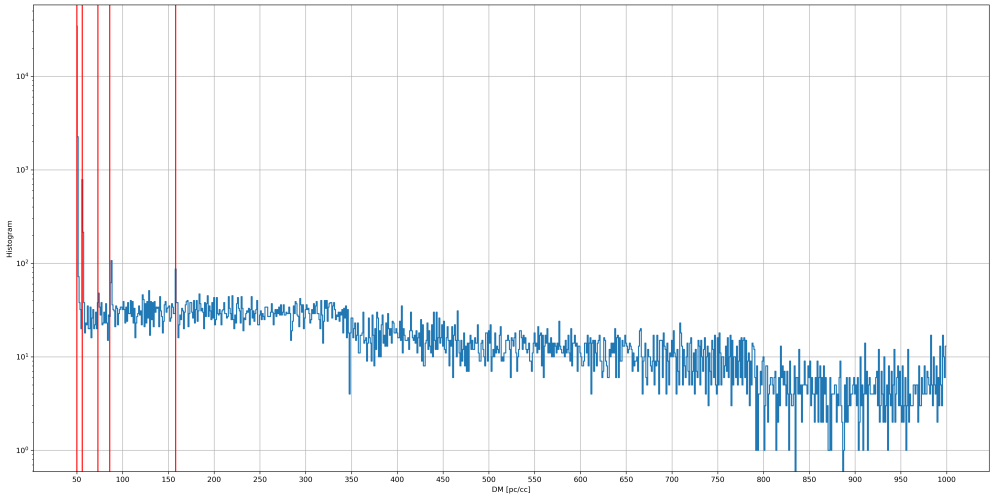


Figure 4.3: DM distribution of AI selected triggers. Red vertical lines correspond to triggers from known pulsar (see section 3.2).

The DM distribution of selected triggers is plotted in section 4.3. Naturally, the triggers from known pulsars are registered in bulk and are selected by the AI showing it's capability. These bins are marked by solid red vertical lines. See section 3.2 for the complete list of pulsars detected.

Since the pulsar triggers are already established, to make numbers managable, all known pulsar triggers are DM-matched and dropped. Performing a manual vetting yields one positive detection of trigger Figure 4.3. PSRCAT([13]) doesn't have any pulsar around the DM near the field. According to YMW16([25]), the Galactic DM contribution is around ~ 19 implying an extremely high extra-galactic DM contribution.

4.4 Ending remarks

The AI solution discussed here can be perfected. Moreover, the end goal of this AI is for it to be incorporated into the VLITE-Fast pipeline for realtime vetting of the model.

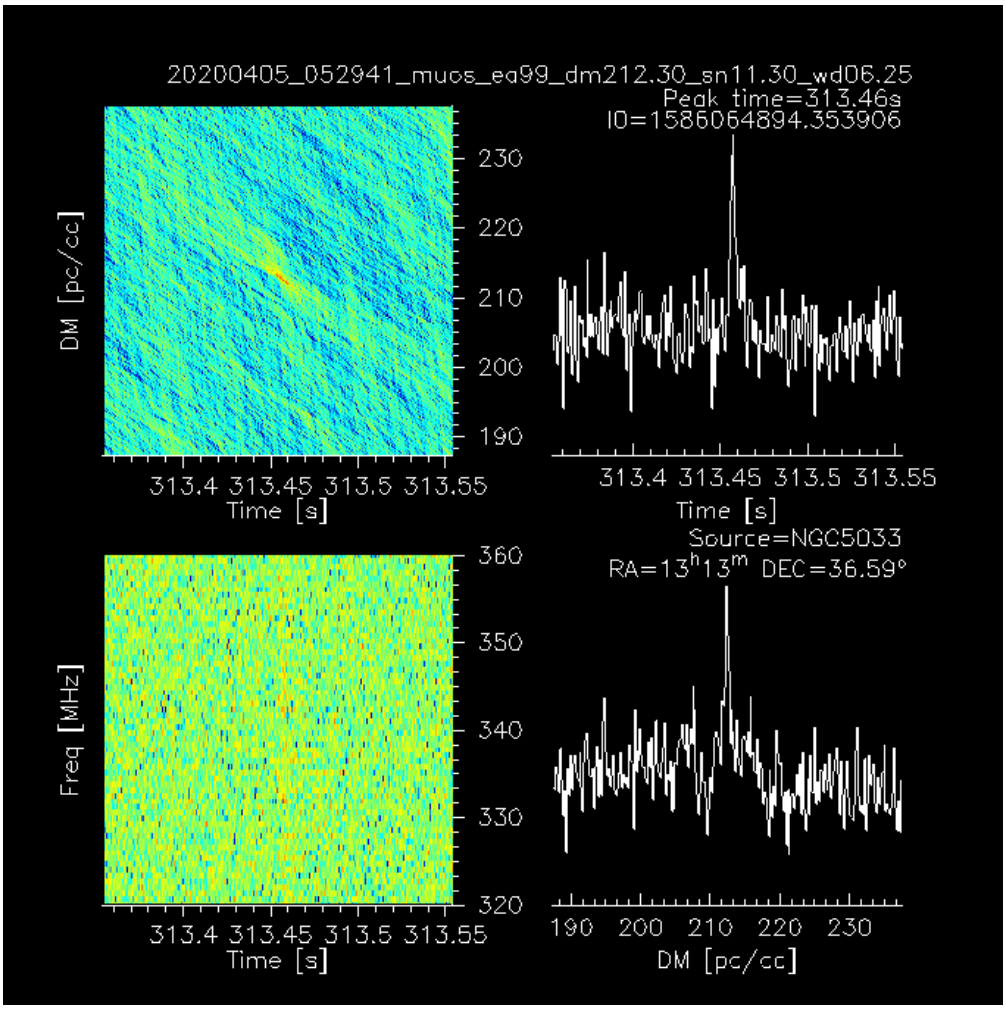


Figure 4.4: A possible FRB candidate found after using AI developed in chapter IV on VFPS dataset

BIBLIOGRAPHY

[1] B. R. BARSDELL, M. BAILES, D. G. BARNES, AND C. J. FLUKE, *Accelerating incoherent dedispersion*, , 422 (2012), pp. 379–392.

[2] ——, *Spotting Radio Transients with the Help of GPUs*, in Astronomical Data Analysis Software and Systems XXI, P. Ballester, D. Egret, and N. P. F. Lorente, eds., vol. 461 of Astronomical Society of the Pacific Conference Series, Sept. 2012, p. 37.

[3] S. BURKE-SPOLAOR, M. BAILES, R. EKERS, J.-P. MACQUART, AND I. CRAWFORD, FRONEFIELD, *Radio Bursts with Extragalactic Spectral Characteristics Show Terrestrial Origins*, , 727 (2011), p. 18.

[4] CHIME/FRB COLLABORATION, B. C. ANDERSEN, K. BANDURA, M. BHARDWAJ, P. BOUBEL, M. M. BOYCE, P. J. BOYLE, C. BRAR, T. CASSANELLI, P. CHAWLA, D. CUBRANIC, M. DENG, M. DOBBS, M. FANDINO, E. FONSECA, B. M. GAENSLER, A. J. GILBERT, U. GIRI, D. C. GOOD, M. HALPERN, A. S. HILL, G. HINSHAW, C. HÖFER, A. JOSEPHY, V. M. KASPI, R. KOTHES, T. L. LANDECKER, D. A. LANG, D. Z. LI, H. H. LIN, K. W. MASUI, J. MENA-PARRA, M. MERRYFIELD, R. MCKINVEN, D. MICHILLI, N. MILUTINOVIC, A. NAIDU, L. B. NEWBURGH, C. NG, C. PATEL, U. PEN, T. PINSONNEAULT-MAROTTE, Z. PLEUNIS, M. RAFIEI-RAVANDI, M. RAHMAN, S. M. RANSOM, A. RENARD, P. SCHOLZ, S. R. SIEGEL, S. SINGH, K. M. SMITH, I. H. STAIRS, S. P. TENDULKAR, I. TRETYAKOV, K. VANDERLINDE, P. YADAV, AND A. V. ZWANIGA, *CHIME/FRB Discovery of Eight New Repeating Fast Radio Burst Sources*, , 885 (2019), p. L24.

[5] J. M. CORDES AND T. J. W. LAZIO, *Ne2001.i. a new model for the galactic distribution of free electrons and its fluctuations*, 2002.

[6] E. FONSECA, B. C. ANDERSEN, M. BHARDWAJ, P. CHAWLA, D. C. GOOD, A. JOSEPHY, V. M. KASPI, K. W. MASUI, R. MCKINVEN, D. MICHILLI, Z. PLEUNIS, K. SHIN, S. P. TENDULKAR, K. M. BANDURA, P. J. BOYLE, C. BRAR, T. CASSANELLI, D. CUBRANIC, M. DOBBS, F. Q. DONG, B. M. GAENSLER, G. HINSHAW, T. L. LAND ECKER, C. LEUNG, D. Z. LI, H. H. LIN, J. MENA-PARRA, M. MERRYFIELD, A. NAIDU, C. NG, C. PATEL, U. PEN, M. RAFIEI-RAVANDI, M. RAHMAN, S. M. RANSOM, P. SCHOLZ, K. M. SMITH, I. H. STAIRS, K. VANDERLINDE, P. YADAV, AND A. V. ZWANIGA, *Nine New Repeating Fast Radio Burst Sources from CHIME/FRB*, , 891 (2020), p. L6.

[7] G. E. HINTON, N. SRIVASTAVA, A. KRIZHEVSKY, I. SUTSKEVER, AND R. R. SALAKHUTDINOV, *Improving neural networks by preventing co-adaptation of feature detectors*, 2012.

[8] S. IOFFE AND C. SZEGEDY, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, 2015.

[9] F. JENET AND S. ANDERSON, *The effects of digitization on nonstationary stochastic signals with applications to pulsar signal baseband recording*, Publications of the Astronomical Society of the Pacific, 110 (1998), pp. 1467–1478.

[10] D. P. KINGMA AND J. BA, *Adam: A method for stochastic optimization*, 2014.

[11] I. R. LINSCOTT AND J. W. ERKES, *Discovery of millisecond radio bursts from M 87*, , 236 (1980), pp. L109–L113.

[12] D. R. LORIMER, M. BAILES, M. A. MCLAUGHLIN, D. J. NARKEVIC, AND F. CRAWFORD, *A Bright Millisecond Radio Burst of Extragalactic Origin*, Science, 318 (2007), p. 777.

[13] R. N. MANCHESTER, G. B. HOBBS, A. TEOH, AND M. HOBBS, *The australia telescope national facility pulsar catalogue*, The Astronomical Journal, 129 (2005), p. 1993–2006.

[14] E. PETROFF, E. D. BARR, A. JAMESON, E. F. KEANE, M. BAILES, M. KRAMER, V. MORELLO, D. TABBARA, AND W. VAN STRATEN, *FRBCAT: The Fast Radio Burst Catalogue*, , 33 (2016), p. e045.

[15] E. PETROFF, J. W. T. HESSELS, AND D. R. LORIMER, *Fast radio bursts*, , 27 (2019), p. 4.

[16] E. PETROFF, E. F. KEANE, E. D. BARR, J. E. REYNOLDS, J. SARKISSIAN, P. G. EDWARDS, J. STEVENS, C. BREM, A. JAMESON, S. BURKE-SPOLAOR, S. JOHNSTON, N. D. R. BHAT, P. C. S. KUDALE, AND S. BHAND ARI, *Identifying the source of perytons at the Parkes radio telescope*, , 451 (2015), pp. 3933–3940.

[17] E. PLATTS, A. WELTMAN, A. WALTERS, S. P. TENDULKAR, J. E. B. GORDIN, AND S. KANDHAI, *A living theory catalogue for fast radio bursts*, , 821 (2019), pp. 1–27.

[18] T. SHANLEY, *Infiniband*, Addison-Wesley Longman Publishing Co., Inc., USA, 2002.

[19] L. G. SPITLER, J. M. CORDES, J. W. T. HESSELS, D. R. LORIMER, M. A. MCLAUGHLIN, S. CHATTERJEE, F. CRAWFORD, J. S. DENEVA, V. M. KASPI, R. S. WHARTON, B. ALLEN, S. BOGDANOV, A. BRAZIER, F. CAMILO, P. C. C. FREIRE, F. A. JENET, C. KARAKO-ARGAMAN, B. KNISPEL, P. LAZARUS, K. J. LEE, J. VAN LEEUWEN, R. LYNCH, S. M. RANSOM, P. SCHOLZ, X. SIEMENS, I. H. STAIRS, K. STOVALL, J. K. SWIGGUM, A. VENKATARAMAN, W. W. ZHU, C. AULBERT, AND H. FEHRMANN, *Fast Radio Burst Discovered in the Arecibo Pulsar ALFA Survey*, , 790 (2014), p. 101.

[20] L. G. SPITLER, W. HERRMANN, G. C. BOWER, S. CHATTERJEE, J. M. CORDES, J. W. T. HESSELS, M. KRAMER, D. MICHILLI, P. SCHOLZ, A. SEYMOUR, AND A. P. V. SIEMION, *Detection of Bursts from FRB 121102 with the Effelsberg 100 m Radio Telescope at 5 GHz and the Role of Scintillation*, , 863 (2018), p. 150.

[21] R. THAKUR, R. RABENSEIFNER, AND W. GROPP, *Optimization of collective communication operations in mpich*, The International Journal of High Performance Computing Applications, 19 (2005), pp. 49–66.

[22] D. THORNTON, B. STAPPERS, M. BAILES, B. BARSDELL, S. BATES, N. D. R. BHAT, M. BURGAY, S. BURKE-SPOLAOR, D. J. CHAMPION, P. COSTER, N. D'AMICO, A. JAMESON, S. JOHNSTON, M. KEITH, M. KRAMER, L. LEVIN, S. MILIA, C. NG, A. POSSENTI, AND W. VAN STRATEN, *A Population of Fast Radio Bursts at Cosmological Distances*, Science, 341 (2013), pp. 53–56.

[23] J. TOMPSON, R. GOROSHIN, A. JAIN, Y. LECUN, AND C. BREGLER, *Efficient object localization using convolutional networks*, 2014.

[24] B. XU, N. WANG, T. CHEN, AND M. LI, *Empirical evaluation of rectified activations in convolutional network*, 2015.

[25] J. M. YAO, R. N. MANCHESTER, AND N. WANG, *A new electron-density model for estimation of pulsar and frb distances*, The Astrophysical Journal, 835 (2017), p. 29.

[26] B. ZACKAY AND E. O. OFEK, *An Accurate and Efficient Algorithm for Detection of Radio Bursts with an Unknown Dispersion Measure, for Single-dish Telescopes and Interferometers*, , 835 (2017), p. 11.

BIOGRAPHICAL SKETCH

Suryarao "Surya" Bethapudi likes to solve problems. He is always eager to learn new things.

If given a choice and if it was possible, Surya wishes to be a field and be everywhere in this Universe and see everything and understand everything.

Surya earned a Master of Science in Physics from the University of Texas Rio Grande Valley in 2020.

email: shining.surya.d8@gmail.com