Vrije Universiteit Amsterdam

Universiteit van Amsterdam

Master Thesis

RFI Mitigation in Radio Astronomy Using a Deep Learning-Based Method

Author: Saba Amiri (11980079)

1st supervisor:	supervisor name	
daily supervisor:	supervisor name	(company, if applicable)
2nd reader:	supervisor name	

A thesis submitted in fulfillment of the requirements for the joint UvA-VU Master of Science degree in Computer Science

April 13, 2020

"I am the master of my fate, I am the captain of my soul"
from Invictus*, by William Ernest Henley*

Abstract

RFI ABSTRACT

Dedication

Acknowledgement

Ack...

Nack...

Handshake failed...

Contents

List of Figures

LIST OF FIGURES

List of Tables

LIST OF TABLES

1

Introduction

1.1 Context

Radio astronomy is the field of study concerning the analysis of radio-frequency radiation originating from celestial bodies. In this field, measurements of stars, galaxies and other astronomical sources are done by their radio frequency emissions. Almost the entire radio frequency spectrum contains valuable information about the universe. Radio frequency radiation comes mostly from non-thermal radiation due to synchrotron radiation or electronic transition. This process of gathering radio signals from celestial bodies is susceptible to interference resulting from proliferation of wireless communication technologies, satellites and aircrafts, wind turbines, etc. Radio-frequency interference (RFI) is a electromagnetic disturbance that occurs in the radio-frequency spectrum that causes the degradation of signal quality in a system. These disturbances may be caused by both man-made and natural sources and effect various electromagnetic systems such as radars, mobile phones, etc. These strong radio signals vary in both time and frequency domains and disturb the radio telescope outputs, therefore adding artifacts to the final readings. The difference in the magnitude of the power received from RFI and the observed astronomical sources cause corruptions in areas of the data that cannot be recovered. Although preventive measures like constructing radio telescopes in remote locations, ground shielding and employment of band-pass filters reduce RFI considerably(5), these methods do not prevent all RFI. Therefore, methods to identify and mitigate interference are of high importance to the radio astronomy communities.

There are a variety of methods available in the literature to deal with RFI. These methods fall into three categories: Linear models, statistical methods and machine learning-based methods.

1.1.0.1 Linear Methods of RFI Mitigation

In linear methods, techniques like Singular Vector Decomposition and Principal Component Analysis are used to identify RFI (7, 8). These methods do not perform well on signals with non-linear and stochastic characteristics.

1.1.0.2 Statistical Methods of RFI Mitigation

The second category of RFI detection methods is based on the assumption that the characteristics of RFI in the time-frequency domain is different than that of the astronomical signal. Typically, the recorded astronomical sources appear as smooth over longer durations of time, while RFI usually has much higher power and is localized in time-frequency plane. One of the most notable methods in this area is AOFlagger (9). Assuming most RFI are *scale-invariant* in time and/or frequency domain - meaning that a change in the input scale will result the same change in output scale-, AOFlagger uses *Scale-Invariant Rank* operator to identify interference. This method has been widely and successfully used to identify and mitigate interference (10, 11). Nevertheless, due to variability of the structure of particular RFI signals, creating an analytical model of them is a challenging task(12). That's why there have been an increased focus on machine learning-based methods as viable methods of function approximation and modeling for RFI mitigation.

1.1.0.3 ML-Based methods of RFI Mitigation

In recent years, with the success of artificial intelligence and machine learning in classification and pattern recognition tasks, a number of ML-based methods have been proposed to address the RFI problem which will fall into the third category above. Especially with the development of deep neural networks and deep learning methods, there has been a surge in the performance and scale of certain pattern recognition and object detection tasks (13). Certain types of deep neural networks like Convolutional Neural Networks, used for object detection, don't need prior knowledge of the target classes, which makes them especially suitable to detect RFI signals from multiple sources. Therefore, it is not surprising that there have been a number of methods presented in literature for RFI detection using deep neural architectures(5, 6, 14).

In this research, we present our research on **"RFI Mitigation in Radio Astronomy Using a Deep Learning-Based Method"**. We propose a framework to identify RFI in

LOFAR correlated data. Our framework defines the task of RFI mitigation as a two-class image segmentation problem and incorporates U-Net deep neural network architecture(4) to solve the aforementioned problem. The main contributions of this work are superior performance compared to similar DL-based methods - covered in section 2.4 - and the use of non-synthetic observation data obtained from LOFAR long time archive for training and testing of the model. We use CPU and GPU resources of DAS-4[1] and SURFsara Lisa[2] compute facilities to train out deep neural network. In convention with classification-based research, this work is evaluated through the use of F1- and F2-scores, accuracy, recall and precision as well as dice similarity index. We also argue about the feasibility of the proposed system to be incorporated into available online workflows(15).

1.2 Research Questions

1.2.1 RQ-1: A performant deep learning-based solution to the problem of RFI detection in LOFAR data

Considering the big strides in both development of deep learning models and their application in various AI related tasks, it is only natural to look into their application in the problem of RFI detection. Basically, RFI can be considered as an unwanted signal artifact among observed signals by the radio telescopes. We can relate the problem of RFI detection to the more general problem of detecting an "abnormal"-in case of binary states- or "with specific characteristics" - in case of a multi-class space - set of "objects" in a data sample. Examples of deep learning methods being applied for similar problems can be found in noise detection (16, 17), signal detection (18, 19, 20) and artifact detection in images (21, 22, 23). The immense power of deep learning methods in pattern recognition without the need for domain expertise and careful engineering of features (24) would be especially helpful in the case of RFI detection since interference characteristics could vary widely based on the nature of their source (25). Deep learning-based methods have already been proposed in this area (5), but on top of some more fundamental differences, the reported performance is poor when tested on real world samples. *RQ-1* aims to find an answer for the question of whether it is possible to detect RFI using a deep neural model.

[1] https://www.cs.vu.nl/das4/
[2] https://userinfo.surfsara.nl/systems/lisa

1.2.2 Training and testing a potential deep learning-based RFI detection model on real, non-synthetic data

One important aspect of every machine learning-based system is its ability to be trained and to generalize on real world data. Packages like HIDE & SEEK (11) have been used to provide training data for machine learning-based RFI detection and mitigation methods (5). The *RQ-2* covers the challenge of training and testing the model on real world data from LOFAR while maintaining an acceptable level of performance.

2

Related Work

2.1 Radio Astronomy Basics and Introduction to LOFAR

Radio astronomy is the science of studying celestial objects using data gathered from the radio portion of the electromagnetic spectrum. Astronomers use radio telescopes to record and analyse radio waves originating from celestial bodies. Radio astronomers usually analyse objects emitting radio waves with frequencies between 3KHz and 900 GHz. The lower limit of the observed radio frequencies is dependant on the opacity of the ionosphere; The higher end of the observable spectrum is limited by the absorption from oxygen and water bands of the lower atmosphere (26).

As for the resolution of the radio astronomy readings, single dish antennae have a physical limitation. The largest steerable dishes today are Byrd Green Bank telescope in United States with a diameter of 110x100 meters and Effelsberg telescope in Germany with a diameter of 100 meters. The largest non-steerable dish in the world is at Arecibo, Puerto Rico, with a diameter of 304.8 meters (27). Since the angular resolution of the dish is limited by its diameter, much larger dishes are needed to achieve sub-arcsecond[1] resolutions(for comparison, the angular resolution of the Efeelsberg telescope for 21cm wavelength signals is 7 arcmins.). To solve this problem, interferometers are used. An interferometer is an array of antennae or telescopes, working together to provide a high resolution reading of the sky equal to that of a single dish with a very large diameter. Separate signals from participating telescopes are combined via a method called aperture synthesis (28). The telescopes in an interferometers can be in different geographical locations with hundreds and thousands of kilometers between them.

[1] an arcsecond is $\frac{1}{206265}$ of a radian

Figure 2.1 shows basic configuration of a two telescope radio interferometer(1). It shows signals from different telescopes combined in a cross-correlator unit. A two-telescope interferometer provides us with one baseline - that is the projected separation between two antennae observed from the emission source). This setup still provides a small amount of information about the source, unless we can constantly relocate the telescopes to change the baseline and perform the observation again. Since this process is impractical, we can instead put several telescopes along the baseline. Having /(N/) telescopes along the baseline would yield $N(N-1)$ unique baselines, forming a synthesized beam. Each baseline not only adds a Fourier component to the observation reading, but the sensitivity of the whole interferometer also grows with the number of telescopes(1).

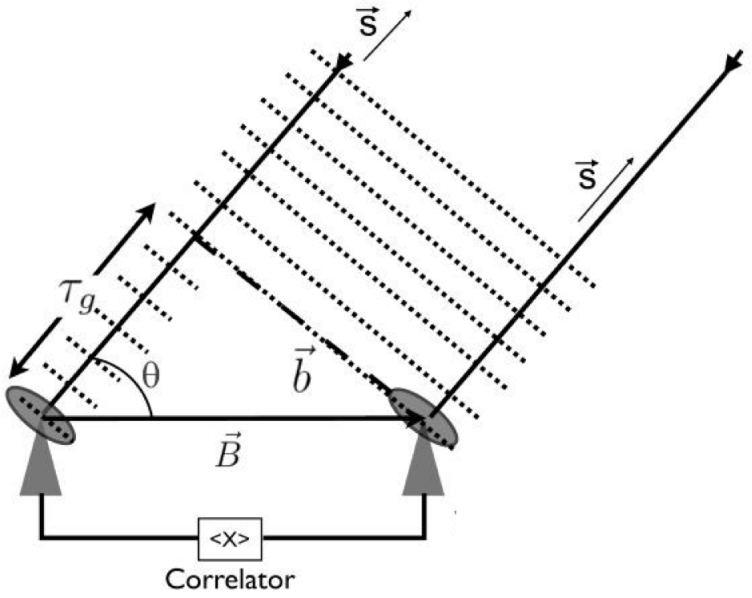

Figure 2.1: Basic radio two-telescope interferometer. $\vec{B}$ is the baseline vector separating the two antennae (1)

2.1.1 Introduction to LOw-Frequency ARray (LOFAR)

LOFAR(Figure 2.2) is a radio interferometer constructed by ASTRON[1] and located across the Europe, including the Netherlands, Sweden, France, Poland, Latvia, Germany, UK and Ireland. It was designed to make lowest frequencies in radio spectrum available to

[1]Netherlands Institute for Radio Astronomy (https://www.astron.nl/)

astrophysical researchers. Using a phase-array design, LOFAR covers low-frequency bandwidth in the range 10-240MHz (corresponding to the wavelengths 30-1.2m). Lofar has 52 dipole antenna stations, spreading from its core at the village of Exloo, Drenthe in the Netherlands. These stations, as of January 2020, include 24 core stations (stations located withing a 2 km radius of the core), 14 remote stations and 14 international stations. The stations have no moving parts, but the all-sky coverage of the dipole stations gives LOFAR a large field-of-view (2).

Figure 2.2: A 2011 aerial photograph of the Superterp, the heart of the LOFAR core. Six core stations can be seen in the circular island, with three additional LOFAR stations in three smaller islands in the lower-left and upper-right corners of the photo(2)

LOFAR combines signal processing both at the station level and at the central facilities. Thus, it is flexible at defining the frequency range, sub-bands and spectral channels. The output of LOFAR antennae at each station will go through some processing steps. The first step is to feed the siganl to Receiver Units(RCUs). The job of RCUs is to amplify and convert the input voltage from antenna into sub-band frequencies. The sub-band selection is done using a 12-bit Analogue to Digital converter and several band-pass filters. In the next step, the digitized output of the RCUs is processed in the FPGA[1]-based Remote

[1]Field Programmable Gate Array

Station Processing (RSP) units (29). Station-level processing will split the bandwidth of the digitized signal into 512 sub-bands using the poly-phase filter, followed by a 1024-point Fast Fourier Transformation (29). Each Measurement Set obtained from LOFAR Long Time Archive (LOFAR LTA) contains data related to one sub-band of an observation (2).

2.2 Basics of RFI and RFI Mitigation in Radio Astronomy

The radio band has been subject of extreme interest in cosmology in recent years. Experiments and surveys performed at Tianlai(30), SKA(31), EVLA(32), HERA(33) and LOFAR(34) on large portions of the sky provide valuable information to the scientists about our universe.

A big challenge in radio astronomy is Radio Frequency Interference (RFI). Contamination from various sources such as cell phones, satellites and airplanes, long-distance communications as well as natural phenomena adversely affects the readings of the radio astronomy instruments(35, 36, 37). The expansion of the detection range of radio telescopes has lead to an overlap between the observable range of the telescopes and the slices of radio spectrum reserved for human-based radio activity such as communication. On the other hand, the fast development of consumer technology reliant on long-distance wireless communications has lead to the expansion of their usage of previously-free frequency bands. The differences in RFI sources lead to different characteristics of the resulting signal in time and frequency domains, leading to the signal function being complex and challenging to model(12, 38). Some passive measures like building the telescopes in less populated areas, shielding the telescope buildings and utilization of band-pass filters could minimize effects of some RFI, but negating all anthropogenic and natural RFI seems to be an impossible task(39, 40). A RFI signal could potentially mask the desired signal and render the contaminated part of the reading unrecoverable. Therefore methods to deal with and mitigate RFI are highly sought after.

The RFI mitigation methods can be applied either pre- or post-correlation(7, 9). Pre-correlation methods deal with data in its highest resolution form. Methods such as RFI blanking(41) are very effective at mitigating RFI. On the other hand, due to the fact that today's radio telescope continuously produce vast amounts of high-resolution data, the computational costs of online RFI mitigation via such methods are very high. Because of the limitation in both storage capacities and the data-transfer bandwidth, the pre-correlation algorithms should be applied online and in a very short time, adding to

the complexity of their infrastructure needs and rendering these methods even more demanding. Even with the proper compute infrastructure, the residual RFI artifacts need to be flagged manually in the resulting signals. Due to the tremendous amount of generated data, the manual process is unfeasible in real world scenarios. Therefore, development of automated post-correlation methods are of utmost importance to the field of radio astronomy.

We can process the signals post-correlation, having to deal with lower-resolution, smaller datasets that still maintain high level of accuracy. One way to deal with RFI is to mask the contaminated parts of the reading. The goal of this procedure would be to mask the parts containing interference while keeping the resulting data loss to the minimum. One basic assumption for RFI masking is the essential differences between RFI and celestial signals in time and frequency(9, 42); While astronomical signals are usually broad-band and have smooth variations over time, RFI often manifests as high intensity areas and appears in short bursts on time axis, either in a periodic manner or sporadic over time, depending on the nature of RFI source.

For RFI signals with low-complexity, linear modeling methods perform well. Linear approaches like correlation-based detection(43), Surface Fitting(7), Singular Value Decomposition(7) and Double Principal Component Analysis(8) are of the notable example of this family of RFI mitigation methods. These methods would perform well in situations where the RFI signal exhibits linear, periodic patterns in frequency-time plane, but are generally unable to model non-linear RFI of a more stochastic nature.

Threshold based methods for RFI detection are low-complexity methods with a notable performance in RFI mitigation. Methods such as CUMSUM(44), Kurtosis-based(45), Scale-Invariant Operator-based RFI tagging algorithm(9), the SumThreshold method(7) and Amplitude Thresholding(46) all fall under the threshold-based filtering category. Combinatory thresholding methods extend the threshold-based methods by adding more dynamicity to the data frames(7).

In recent years, with the advent of Artificial Intelligence and Machine Learning-based methods for a wide variety of tasks, particularly pattern recognition tasks similar to RFI mitigation, a number of methods have been proposed for ML-based RFI mitigation. The proven robustness of ML-based techniques for modeling complex signals and systems in use-cases such as medical applications, self-driving cars, industrial control systems and image processing, all dependant on various forms of pattern recognition and accurate distinction and modeling different mathematical functions in the input data, make them a viable solution for the problem of RFI mitigation. In the next section, we will give an overview

of different ML-based methods of RFI mitigation and provide the theoretical basis for the method and pipeline proposed in this research.

2.3 Machine Learning in RFI Mitigation

The problem of RFI mitigation, as defined in section 2.1 shows promises of being solvable by employing machine learning-based techniques. The nature of the problem - segmentation and separation of several existing signals, each with their own characteristics, which are theoretically definable by a set of mathematical functions - lends itself to various machine learning-based solutions. In this section we will go over different ML-based methods of RFI mitigation.

An obvious choice for ML-based RFI mitigation are supervised learning methods. The fully supervised learning paradigm, which requires a training dataset with known value of target function for each input sample can be applied for the problem of RFI mitigation due to abundance of available data. Methods such as Naive Bayes(47), Gaussian Mixture Models(48), Random Forests(49), Multi-class SVM(50), Multi-Batch K-Means(50) and K-Nearest Neighbors(38) are supervised learning methods of note. One important aspect in traditional supervised learning models is the curation of feature vector. In most cases, the success or failure of a model depends on the careful engineering of the feature vector as much as model selection and parameter tuning. In Wolfaardt et al.(48), for example, the authors use two different feature vectors, a reduced feature set vector and a feature vector augmented with delta frames for training. Mosiane *et al.* in (49) create their feature vector using a slide window of the size 8 second and extracting statistical features from each window for each baseline. Designing these feature sets requires careful experiments with the set of available data, in many cases the transformation of data into different planes, and in some cases feature extraction and generation of new, more efficient features from the initial set.

In recent years, deep neural networks have shown tremendous success, outperforming most "classical" machine learning models in a variety of tasks including natural language processing(51), time series classification(52) and a variety of image processing related tasks(53, 54, 55), including image segmentation(56, 57, 58, 59).

In the next sections in this chapter, we provide a high-level overview of the image segmentation task as a viable way to formulate the problem of RFI mitigation in the context of ML/DL. Next, we review recent methods focused on RFI mitigation using deep learning-

based methods, and finally we will introduce the U-Net deep neural architecture which is used in this research for the task of RFI mitigation.

2.4 Basics of Machine Learning-Based Image Segmentation

Image segmentation is one of the classification tasks in machine learning. Considering an image as a tensor and assuming that there are several semantically-different entities in that image, the task of the segmentation task is to label each pixel in the tensor with a label corresponding to a specific class. From self-driving vehicles(60) to medical imaging(61), it has been extensively researched in the previous years(62, 63).

In its simplest form, we can define the segmentation task as simple *Image Classification*. We gain no knowledge about the objects inside the image and there will be no labels produced. Instead, the ML algorithm, given proper training dataset and accurate modelling, will assign a class label to the input images, identifying the main object in that sample(64). *Localized Image Classification* performs the same classification task as *Image Classification*, but in addition to the image label, it will also localize the main detected object, providing us with a bounding box containing the object itself(65). Extending on the previous branches of image segmentation, the aim of *Object Detection* family of methods is to detect and label each object in an input image, e.g. cars, humans, tumors. Furthermore, the algorithm is expected to localize each object and provide the bounding boxes corresponding to them(66). In *Semantic Segmentation* methods, we try to not to provide the localized objects through their boundaries but to label each pixel with the corresponding object's label(59). The focus of this section henceforth will be on *Semantic Image Segmentation* due to relevance of this task to the problem of RFI mitigation. The case for the foundations of the mentioned connection will be laid out in the next chapter.

Similar to the algorithmic needs discussed in section 2.3, the need for careful design of features relevant for each image segmentation task, especially semantic segmentation, which usually requires domain expertise, engineering and extraction of new features makes these tasks technically challenging. Therefore, deep learning-based methods with their capability to detect and classify patterns without any *a priori* knowledge about the feature space makes them a viable choice for semantic image segmentation tasks.

2.4.1 Image Segmentation based on Deep Learning

Before the advancement of deep neural networks and their increased application in this research area, there have been numerous methods dealing with the problem of semantic

image segmentation with various levels of success(67, 68, 69, 70). These methods include the use of morphological methods(71).

The success of deep neural architectures in solving similar problems such as DL-based image classification and object detection(72, 73) lead the way for research into their application for the semantic image segmentation. As mentioned before, the ability of such methods to extract and learn feature vectors without the need for domain knowledge and engineering a curated set of features makes them extremely attractive for this type of tasks.

There are several famous deep architectures, designed for various use-cases, with performances so extraordinary they have become standard building blocks and benchmarks for many similar tasks. AlexNet(72), ResNet(74) and GoogLeNet(75) are some of the noteworthy examples of such famous networks. Since the architecture we use in this research is based on the *Fully Convolutional Neural Network* architecture, we're going to describe the details of GoogLeNet, which is based on a *Convolutional Neural Network* architecture.

The GoogLeNet, introduced by Szegdey *et al.*, is a convolutional neural network architecture with 22 layers. The number of parameters is almost 12 times less than AlexNet, making its computational load lighter and its convergence time shorter. An inception unit is introduced in this architecture, which contain a Network in Network (NiN) module, a pooling unit, and two high-res and low-res convolutional layers. The inception module tries to cover a larger area while maintaining the higher-resolution, smaller-sized windows to preserve the fine-grained information in the image. This unit convolves the more accurate window and the less accurate one in parallel. A series of trainable Gabor filters make this possible.

A very successful extension of this CNN-based architecture, *Fully Convolutional Network (FCN)*, has been proposed by Long et al.(3). Long *et al.* used the available standard architectures and replaced the fully connected layers with convolutional layers. This allowed for the output of these layers to be spatial maps and not classification metric values. This lead to learning of hierarchical features, adding another dimension to the already-powerful mechanism of feature extraction in the available CNN-based architectures. The resulting spatial maps would then be upsampled using deconvolution operators which in turn provided dense layers of labeled pixels. Figure 2.3 depicts the innerworkings of the CNN- and FCN-based semantic segmentation.

The FCN architecture, despite its powerful features and noticeable improvements over existing architectures has several short comings. It can't encode global context because of the spatial invariance in its design. The model is not instance-aware, the computational load is still too high for real world scenarios in presence of high resolution images and

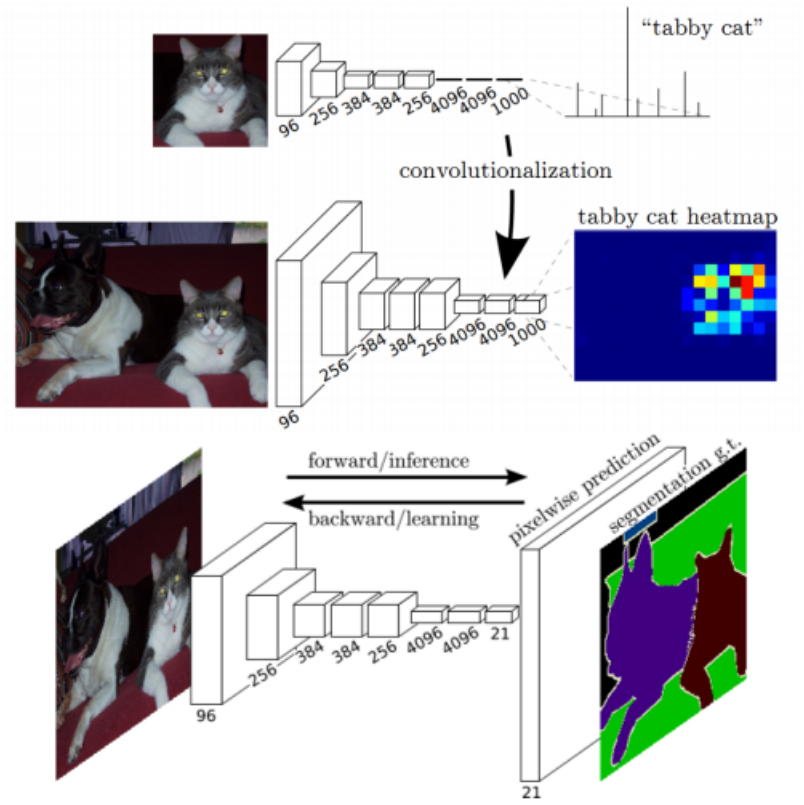

Figure 2.3: FCN architecture diagram by Long *et al.*(3). The FCN architecture produces spatial heatmaps.This is possible by replacing the fully connected layer in CNN with a convolutional layer. The upsampling is done by deconvolution operators which enables per-pixel semantic labeling

it doesn't perform well for sequence processing. There have been numerous methods presented in the literature that provide solutions for one or several of the FCN's shortcomings, e.g. (76, 77, 78, 79).

2.5 U-Net Deep Neural Network Building Blocks and Architecture

U-Net is a specific DNN architecture proposed by Ronneberger *et al.*(4) to perform image segmentation on biomedical imagery. This section defines the building blocks of U-Net DNN and then reviews the original U-Net architecture.

2.5.0.1 Convolution

Convolution layer is the main building block of Fully Convolutional Neural Networks which are used extensively for image segmentation tasks and the U-Net architecture is based upon them. The convolution layer makes use of convolution operator by applying different filters to the input data through the said operation. In convolution layer, the convolution operator is applied to the input data using a convolution filter (or kernel) of specific size(e.g. 3x3, 5x5) and produces a feature map. The convolution filter will slide over the input data in a number of steps that cover the whole input data and in each step a matrix multiplication operation is performed. The result of each multiplication operator is a scalar number which is inserted in the feature map. The sliding window operation has a *stride* parameters that determines how many pixels should the kernel move at each step, with bigger strides leading to less overlap. At each layer, different filters can be applied to the input data, resulting in a 3d feature map of the input sample.

2.5.0.2 ReLU Activation Function

An activation function is a mathematical function that determines the output of each neuron in the neural network. The input of the activation function is the weighted input of the neuron inputs. Based on the output of the activation function, the neuron state will be set to active or inactive. Activation functions are usually computationally inexpensive, since they need to be calculated many times on a large number of neurons in the neural network - especially the case in deep neural networks. One of the main roles of the activation function is to regularize the output of the neurons. Certain activation functions also have been used to introduce non-linear properties to the network. The Rectified Linear Unit (ReLU) activation function(80), defined as

$$f(x) = max(0, x)$$

is one of the most popular activation functions in deep learning settings(81). It is scale invariant and computationally efficient. Its linearity means that its slope doesn't saturate in presence of large inputs and allows the network to converge quickly. It also has a derivative function and allows for backpropagation through the network.

2.5.0.3 Batch Normalization

One of the problems that arise during the training of a deep neural network is that even if you normalize your input data, as the information goes through different layers the

neuron outputs will not be in the same range and could possibly blow out because their distribution is changing during the training phase. This will lead to slow down of network convergence since each layer would try to adapt to its new distribution resulting from its activation function outputs. This phenomenon is called *internal covariate shift problem.*

To address this problem, we use batch normalization(82) in each layer of the deep neural network. At each training step, the batch normalization layer calculates the mean and variance of the layer inputs as

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_{l_i}$$

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n} (x_{l_i} - \mu)^2$$

in which μ and σ^2 are respectively mean and variance of the layer inputs, n is the number of inputs and x_l represents the input value.

Next, the batch normalization layer normalizes the input values using the previously calculated statistics as

$$x_{l_i} = \frac{x_{l_i} - \mu}{\sqrt{\sigma^2 + \epsilon}}$$

where $\overline{x_{l_i}}$ is the normalized value of each input and ϵ is a constant added for numerical stability.

The problem with this operation is, this simple act of normalization might change what the layer actually represents. To address this problem and make sure that this transformation represents the identify transform, the outputs of the layer - the activations - are scaled and shifted using parameters γ and β through the following transformation

$$y_i = \gamma\overline{x_{l_i}} + \beta$$

in which y_i is the output value corresponding to the normalized input value x_{l_i}. γ and β are learned during the *Stochastic Gradient Descent* process.

2.5.0.4 Pooling Layers in CNNs

Pooling is one of the important aspects of CNNs. During the learning phase of a CNN, different areas of the network inputs are encoded into a feature map. Each level in the CNN uses the feature maps from previous layers to learn higher level features of the data. This process will make the features learned in the higher layers susceptible to the position of the features in the feature map. This dependency on the location will reduce

the generalization capabilities of the network. Instead, what we want is spatial invariance, meaning the network can detect features in the input data under different transformations, e.g. rotation and skewing. To solve this issue, pooling layers(83) are placed after each convolutional layer.

The main job of pooling layer is to reduce the resolution of the feature map. The pooling layers performs a non-linear downsampling operation by partitioning the input data into windows and returning one value per window. This operation reduces the resolution of the feature map and increasing the generalization capabilities of the higher levels in the network to detect features by making them more reliant on relative position of features rather than their exact position in the input data. The pooling operation also reduces the training costs of the network, e.g. memory needs and computational overhead.

There are different pooling strategies used for different applications of CNNs. Here we introduce max-pooling(83), which is simply reporting the maximum activation value in the pooling window. A max-pooling layer has a *Stride Size*, which determines the size of the non-overlapping square partitions the input data is split into.

2.5.0.5 Drop-Outs in Deep Neural Networks

One of the main pitfalls of trying to fit a machine learning model to training data is overfitting. Overfitting means training and fine tuning the model on the training dataset to the point that it performs very well on the training dataset, but its generalization capabilities are hindered. In this case, the model has learned the statistical noise in the data and will assume it is also part of the function we are trying to approximate.

To deal with overfitting, regularization methods have been introduced for a variety of different machine learning paradigms. In neural networks, the regularization schemes should also deal with co-adaptation. Co-adaptation is the situation in which some neurons in a neural network have more predictive influence than the others. In a co-adaptation, as the training progresses some neural links become more powerful to the point that only a small fraction of the network nodes will be actively participating the the rest would become irrelevant because of their weak links. This will lead to ineffectiveness of expanding the size of the network and capping the capabilities of neural networks. To resolve this issue in neural networks drop-outs were proposed.

Drop-out(84, 85) regularizes the network by randomly turning off or "dropping out" some of the neurons during each training phase. The effect of drop-out is that the network cannot rely on a subset of learning units to perform the prediction and all the units are involved in the process. This closely simulates having an ensemble of smaller, weaker

classifiers trained on the data and then using their collective results. The drop-out is done with a drop-out value which determines the probability of each node dropping-out during each training phase.

2.5.1 U-Net Deep Neural Network Architecture

The U-net DNN, an FCN based architecture, was first proposed by Ronneberger *et al.* for biomedical image segmentation. The U-Net architecture has two symmetric paths: Encoder and Decoder.

The encoder path, also known as the contraction path, is designed to capture the context in the input image - that is, shallow, low-level information about the image feature maps in latent space. It is comprised of convolutional and max-pooling layers stacked on top of each other. U-Net precomputes a weight map for the categorical cross entropy loss during the encoding stage. This operation is done by a stack of units, each comprised of two 3x3 unpadded convolutions, followed by a rectified linear unit(ReLU) and an output downsampling stage implemented by a max-pooling unit of the size 2x2 with the stride 2. Each downsampling layer doubles the feature channels.

The decoder path, also called expansion path, enables us to localize the features in a precise manner using deconvolution operators. In contrast to contraction path, operations in this path is done by employment of upsampling units. Each upsampling unit is followed by a 2x2 deconvolution that cuts the number of feature channels by half. The deconvolution segment is followed by a concatenation with the corresponding cropped feature map from the contraction path. Cropping is done in each convolution due to the loss of border pixels. The last stage is two convolutions of the size 3x3, each followed by a rectified linear unit. Putting these two paths together gives us an end-to-end fully convolutional network, consisting of only convolutional layers and no dense layers. This allows the architecture to accept input data of any dimension size. The final layer is a 1x1 convolution. It allows mapping of each of feature vector - of the size 64 - to the predefined number of classes. There are 23 convolutional layers in total in the U-Net architecture.

Figure 2.4 shows the U-Net architecture proposed in (4). The input image size should be selected in a way that allows 2x2 max-pooling operations. That means rectangular-shaped input with equal x- and y-sizes.

In the section 2.5.1.1 we will discuss the loss function of the original U-Net design. It is imperative to understand the loss function of the original paper in order to better understand the choice of loss function for this research, described in the next chapter.

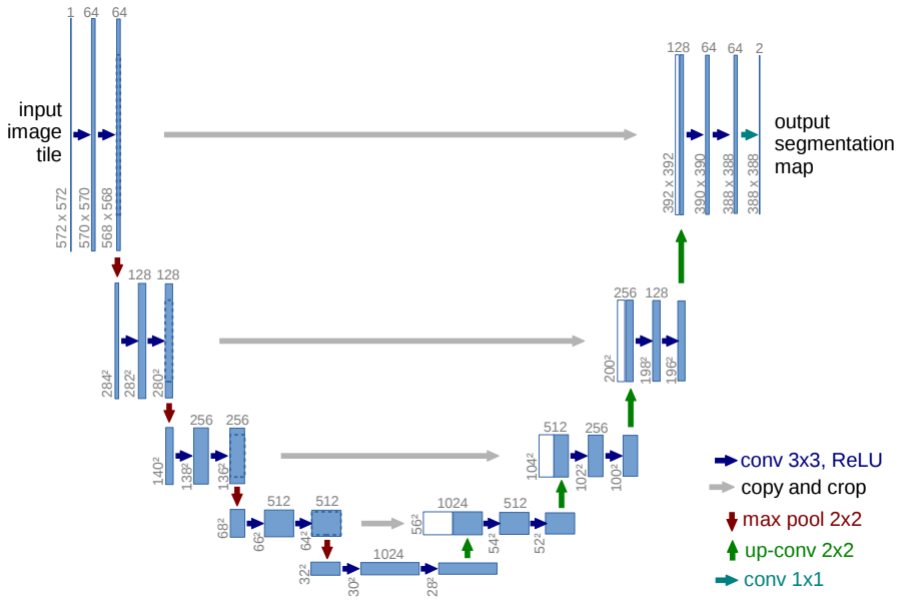

Figure 2.4: U-Net architecture diagram by Ronneberger *et al.*(4). The input image size is 572x572x3. Each blue unit represents a multi-channel feature map, with the number of channels noted on top. The left path contracts the data, while the right path expands the data through upsampling operations. The arrows represent different operations.

2.5.1.1 Original U-Net Loss Function

The loss function of the original U-Net is a pixel-wise soft-max over the final feature map combined with cross entropy. The soft-max is defined as

$$p_k(x) = e^{a_k(x)}/(\sum_{k'=1}^{K} e^{a_{k'}(x)})$$

in which the number of channels is K, $p_k(x)$ is the approximated minimum function and $a_k(x)$ is the activation in feature channel k at the pixel position $x \subset \Omega$ given $x \subset \mathbb{Z}^2$. The cross entropy function, denoted as

$$E = \sum_{x \in \Omega} w(x) log(p_{l(x)}(x))$$

is used to penalize the deviation of $p_l x(x)$ from 1. In cross entropy formula the class label is defined as $l : \Omega \mapsto \{1, ..., K\}$. l is assigned to each pixel. $w : \Omega \mapsto \mathbb{R}$ is the weight map introduced to give more importance to some pixels over the others.

2.6 RFI Mitigation Based on Deep Learning

Due to the attraction of deep learning-based methods for pattern recognition tasks, several deep learning-based methods have already been proposed in the context of RFI mitigation.

Akeret *et al.* in (5) propose a U-Net deep neural network for RFI mitigation. The basic network architecture is the same as the original scheme proposed by Ronneberger *et al.*(4). Different number of layers(3,4,5) and features(16, 64) have been tested for the proposed U-Net. The network is trained using synthetic data from HIDE & SEEK package(11) which simulates radio data with *burst* RFI contamination. Using this dataset, the network reports an Area-Under-the-Curve (AUC) score of 0.96 and a F-1 score of 0.85, compared to the 0.75 score of the same measure by SEEK package's *SumThreshold* function. The trained network is then tested on the data from Bleien Observatory. The results of this simulation are not reported, but the ROC and precision-recall charts show inferior performance compared to the *SumThreshold* method for Bleien dataset.

Burd et al. (14) have employed a *Long Term Short Memory Recurrent Neural Network (LSTM RNN)* for RFI mitigation. The network is trained using RFI-contaminated data from Giant Metre-Wave Radio Telescope (GMRT). GMRT records data at 610MHz in a bandwidth of 33KHz divided into 256 channels. The data comes from 30 antennas leading to 435 baselines. The LSTM model, due to its temporal nature(86) is trained using sequences of data, with each block having the format:

$$[TimeStamp, Polarization, Baseline]$$

The proposed RNN architecture consists of 1024 LSTM cells(87), with a sigmoidal activation function. The cost function for each cell is *Sigmoid Cross Entropy with Logits*, minimized using *Adam* optimized. The reported F-1 score is 0.17.

In (6), Kerrigan et al. use an FCN architecture for RFI mitigation using synthetic amplitude and phase data from HERA(33) using a Hera simulator, namely `hera_sim` [1]. The network architecture is very similar to U-Net, with a fully connected convolutional layer after the downsampling feature construction layer and an upsampling stage for per-pixel class label assignment. The deviation from U-Net architecture here is that the uniformity of number of feature layers for each convolutional layer is broken and instead, an image pyramid approach is used with the number of features increasing as it it nears the fully

[1] https://github.com/HERA-Team/hera_sim

connected convolutional layer and decreases along the path towards the label prediction layer. The authors report F-1 scores of 0.90, 0.70 and 0.28 for *narrowband*, *narrowband burst* and *broadband burst* types of RFI in simulated data respectively.

Yang et al.(88) propose a modified U-Net based architecture called RFI-Net to detect RFI in input signals. They modify the standard U-Net by increasing the depth of the network on both encoding and decoding paths. Also, they add residual units to their network. Residual units(89, 90) help prevent a deep neural network's degeneration; That is, during the training process, after reaching a certain threshold in training accuracy with training errors increasing afterwards. These errors are inherent to the deep neural network because of biased calculations done by many layers of the network(91). Two types of residual units have been designed for this architecture: *upsampling residual units* and *downsampling residual units*. The residual units connect three layers of the network as one unit. The authors claim this connection stabilizes the network's update process and the process of gradient disappearance, caused by the inability of the middle layers to update their parameters effectively, is slowed down.

The authors use simulated data generated by HIDE(11) to train their network. The Five-hundred-meter Aperture Spherical radio Telescope (FAST)(92) is mentioned as their target site for RFI detection, but the only real data they use if from Bleien observatory. There is no indication of the characteristics of the data/RFI generated for training or the real data used for testing. Regardless, the authors report F-1 score of 0.93 for the test of their model on real data from Bleien.

3

Method

In the previous chapters we presented the RFI mitigation problem, gave an overview of different families of methods to deal with it, and focused on ML-based methods for RFI mitigation. We especially focused on deep learning-based methods and summarized the state of the art in literature. We also provided some context about the task of image segmentation, with special emphasis on semantic image segmentation. We also provided the theoretical basis for the U-Net deep neural networks.

In this chapter, based on the context provided previously we will formulate the RFI mitigation problem as a form of semantic segmentation and present our solution based on the U-Net architecture and the modifications we have done to the original scheme.

3.1 Problem Formulation

The Semantic Image Segmentation problem, as described in the previous chapter, deals with identifying classes of objects in an n-dimensional tensor and labelling each pixel with the corresponding class denominator. In an image processing task, the input data would be a tensor of the size MxNxC, where M and N are the length and width of the image and C is the number of channels in the image, i.e. 3 for an RGB image. Theoretically, we can extend any of the dimensions and even add more dimensions; The segmentation algorithm would still be able to detect the artifact in the n-dimensional space. This makes it perfect for our RFI detection problem.

The input data is in the shape of an n-dimensional tensor, each dimension representing one semantic aspect of the data. Based on this perspective on data and algorithms, defining the RFI detection and tagging then becomes a binary semantic segmentation problem. We'd have two classes: non-RFI pixels and RFI-contaminated pixels. The job of the

algorithm would be to label each pixel with the corresponding label. Each "pixel" in our data sample would be a smaller tensor containing data from the smallest unit of resolution in any reading.

Image segmentation algorithms might forego some general morphological preprocessing steps to help algorithm make better classifications of pixels, but since the input data is not in the form of an "image", the assumptions and domain knowledge about them are irrelevant and no image-specific preprocessing steps will be taken for the input data.

3.2 U-Net Architecture

We use the basic U-Net architecture described in (4) and make necessary adjustments. The input is a 160x3072 tensor. The network has an encoding and a decoding path, with convolution and deconvolution operators with the filter size of 3x3.

In contraction path, each convolution operator is follower by a batch normalization unit. The batch normalization segment is followed by a ReLU activation and dropout. Last, we downsample the unit input by using a 2x2 max-pooling operator with the stride of the length 2. As with the original architecture, each downsampling step doubles the number of filters.

The expansive path, which mirrors the contracting path, contains upsampling units followed by a 2x2 deconvolution operator. The deconvolution operator halves the number of feature channels. The results will be then concatenated with feature map from the parallel unit in the encoding path. Next, two 3x3 convolution operators followed by batch normalization units, drop outs and a ReLU unit would follow. In the final layer, as in original architecture, a 1x1 convolution operator maps the feature vector to our binary classes.

Total number of parameters for our network is 2,164,305, of which 2,161,361 are trainable. Figure 3.1 shows the general architecture of our U-Net network.

3.3 Dataset and Preprocessing

The training and test datasets have been obtained from averaging pipeline of LOFAR Long Time Archive (LTA)[1]. Data from the project Epoch of Reionization (EoR)[2] was selected for this work. 107 measurement sets related to different consecutive sub bands

[1] https://lta.lofar.eu/

[2] http://www.lofar.org/astronomy/eor-ksp/epoch-reionization.html

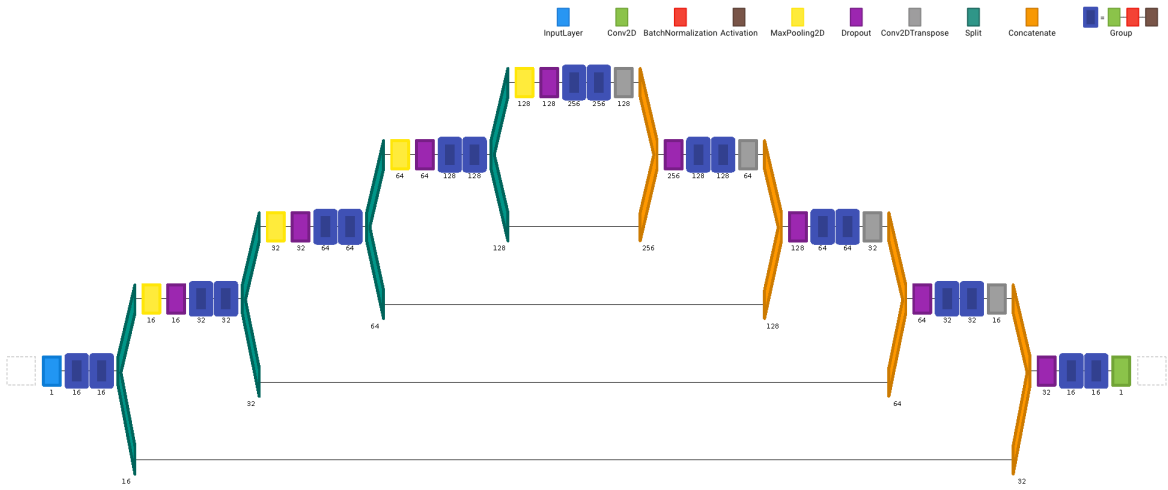

Figure 3.1: The proposed U-Net architecture

from observation *L548671* were obtained from LOFAR LTA for training and validation. 107 separate measurement sets were obtained as the holdout set to measure the performance of the model. Table 3.1 contains the high level structure of the measurement sets obtained from LOFAR LTA.

We use columns *"Data"*, *"Flag"*, *"ANTENNA1"* and *"ANTENNA2"* for this work. The *"Data"* column contains the data from each reading. The *"Flags"* column contains the RFI flags for each "pixel" in the *"DATA"* column. Since each measurement set contains correlations between two antenna pairs, we use columns *"ANTENNA1"* and *"ANTENNA2"* to get the list of antennas in each measurement set and differentiate between them. *"DATA"* and *"FLAG"* column have the shape 329439x15x4 and *"ANTENNA1"* and *"ANTENNA2"* columns have the shape 329439x1. There are 2211 unique $(ANTENNA1, ANTENNA2)$ pairs, each with 149 data points; Thus, the shape of the sample associated to each antenna pair would be 149x4x15. The first dimension is the *number of the data points per readings for the current antenna pair*, second dimension indicates the *number of polarizations* and the third dimension indicates the *number of channels*.

3.3.1 Selecting Polarizations

Since most of the RFI is found in the first polarization, we're going to use only the data from the first polarization. This will reduce both the volume of the data going through our deep neural network and the computational load of training the network. For each measurement set, we obtain a list of unique antenna pairs from *"ANTENNA1"* and *"ANTENNA2"* and

for each antenna pair, we will keep the first polarization and drop the rest. That will leave us with a data array of the shape 149x15x1. It should be noted that since the underlying DNN framework we're using(Keras[1] on TensorFlow[2]) doesn't support complex numbers, we take the absolute value of the complex values of data and use it as the input.

3.3.2 Resizing

Since in the encoding path of the U-Net, the pooling units have a stride of 2, the dimensions of the network should be divisible by two, and since we have four max-pooling operation during the encoding path, each dimension should be divisible by 16. Thus, because our dimensions are 149x15x1, we need to resize our input accordingly. After trying several different resizing schemes, we decided to choose the size 160x32x1. It ensures the divisibility by 16 without losing information in the process. After the resize operation, the number of pixels change from 2235 to 5120.

3.3.3 Stacking

The next step in our data preparation phase is stacking, in which we concatenate data from different channels of observation. For each unique antenna pair, we perform the previously described operations and then concatenate the newly processed sample to the already existing array of samples along the data dimension. For example after processing 10 antenna pairs(160*32*x1), the stacked array of baselines would have the shape 160x*320*x1.

3.3.4 Normalization

To normalize the data, we tried two different strategies: to normalize the data after the resizing stage and before concatenation; and after concatenation as a one-time operation on the whole array. We opted to normalize the data using first approach, since the second approach in our tests resulted in inaccurate, heavily biased classifiers. We believe this is because when we normalize the whole array, if a small number of samples have extremely high values in their data - due to RFI - then normalizing over all of them results in a very skewed dataset. On the other hand, normalizing over each sample results in a more smooth data distribution over sample data, and a much more accurate classifier.

[1] https://www.tensorflow.org/guide/keras
[2] https://www.tensorflow.org/

3.3.5 Train-Test Split

After applying all of the above-mentioned data for all of the obtained measurement sets, we do split the samples randomly into train, test and validation(hold-out) datasets with a 70%-20%-10% ratio respectively.

3.4 Pipeline and Experiments

The pipeline for this work was implemented using Python 3.7. The pipeline was built on top of Tensorflow 2-GPU(93) and Keras(94). The U-Net was coded manually using building blocks of Keras/TF. The Casacore library(95) was used to process measurement set data from LOFAR LTA. Image resize was done using scikit-image library(96) with linear method without employing Gaussian anti-aliasing.

The initial experiments were done on normal CPU-nodes on DAS-4 with two quad-core processors of 2.4 GHz frequency and 24 gigabytes of RAM. The jobs were submitted to the SLURM batch queuing system.

In later stages, for the training of the deep neural network on data obtained from LOFAR LTA we moved our workload to Lisa GPU cluster of SURFsara. On the LISA, we reserved nodes with four Titan-V GPUs, each having 12 gigabytes of RAM. The nodes have quad-core 1.7 GHz CPUs and 256 gigabytes of RAM. The jobs were sent to the batch processing system of LISA and the test stage on hold-out data was timed. At any time during the experiments, as per specifications of the LISA system, there were no other jobs running on the reserved machine, making the time-measurements more accurate.

The final network was trained 5 times, each time with a differently randomized data split. The reported results in the next chapter are the average of the 5 runs. The plots are taken from the last run.

Column Name	Description
UVW	Vector with uvw coordinates (in meters)
FLAG_CATEGORY	The flag category, NUM$_CATflagsforeachdatum$
WEIGHT	Weight for each polarization spectrum
SIGMA	Estimated rms noise for channel with unity bandpass response
ANTENNA1	ID of first antenna in interferometer
ANTENNA2	ID of second antenna in interferometer
ARRAY_ID	ID of array or subarray
DATA_DESC_ID	The data description table index
EXPOSURE	The effective integration time
FEED1	The feed index for ANTENNA1
FEED2	The feed index for ANTENNA2
FIELD_ID	Unique id for this pointing
FLAG_ROW	Row flag - flag all data in this row if True
INTERVAL	The sampling interval
OBSERVATION$_ID$	ID for this observation, index in OBSERVATION table
PROCESSOR_ID	Id for backend processor, index in PROCESSOR table
SCAN_NUMBER	Sequential scan number from on-line system
STATE_ID	ID for this observing state
TIME	Modified Julian Day
TIME_CENTROID	Modified Julian Day
DATA	The data column
FLAG	The data flags, array of bools with same shape as data
LOFAR_FULL_RES_FLAG	flags in original full resolution
WEIGHT_SPECTRUM	weight per corr/chan

Table 3.1: LOFAR LTA Averaging Pipeline Measurement Set Structure. Data is related to the project Epoch of Reionization (EoR), pipeline *L548671*

4

Results and Discussion

4.1 Experiments History

Before presenting the results from the final model, this section presents some important parts of the research process of this work and tries to rationalize the choices made for final the model.

4.1.1 Model Tuning and Improvements

Numerous experiments were done on datasets with different sizes to evaluate the performance of the original U-Net model for the use-case of this work. Based on the results, a number of modifications have been done to the original architecture which result in higher performance of the model.

4.1.1.1 Batch Normalization

The first modification is in the contraction path: After each convolutional unit, a batch normalization unit was added. To increase the generalization capabilities of the network, we tried higher learning rates to compensate for the noise in the *Gradient Descent* over our loss function, but the results were not satisfactory; Thus, we added a batch normalization method which not only allowed the training to be faster via reducing the number of epochs necessary to converge, but also by keeping the activation values in check - reducing the covariance shift of the hidden unit values - it allowed us to increase the learning rate and achieve better generalization when testing the network with real, unsynthetic data(97). Coupled with the dropout segment-with a low value, based on model performance under different experiments-, it also adds some perturbation to the activations of its correspond-

ing layer. This process of adding perturbations to the layer activations will lead to less overfitting and better generalization of the model without losing too much information.

4.1.1.2 Mini-Batching

Our aim is to have a model that is both accurate in semantic classification of pixels and has a low enough computational load-both during training and inference, so that we can potentially plug it into an online learning pipeline. Therefore, we resort to using mini-batch stochastic gradient descent in our model. This method will reduce the computational load and training time considerably.

Mini-batching, along with all the advantages it brings, creates a big problem for our use-case: since our dataset is skewed and we have class imbalance, mini-batching will result in a strong bias in the model; Because most mini-batches will contain very few or no RFI-contaminated pixels, the model will converge after a few epochs and classifies every pixel as non-RFI.

To solve this problem, we tried different loss functions and came up with a combination of Dice Similarity and Cross Entropy. Section 4.1.2 describes the selection process.

4.1.2 Selection of Loss Measure

One of the most challenging aspects of this project was the selection of the loss measure. Due to the class imbalance in our data, a classifier that classifies every pixel as non-RFI contaminated would achieve a very high level of accuracy, but wouldn't be useful. In this section, the original loss measure used in the U-Net paper will be examined and the rationale and details of our proposed loss function are explained.

4.1.2.1 Cross Entropy Loss Function

Given

$$\mathbf{P} = \begin{cases} p & Y = 0 \\ 1 - p & Y = 1 \end{cases}$$

with $\mathbf{P}$ being the cumulative distribution function of our binary classes and Y as our class denominator, the predictions can be defined as the sigmoid function $\hat{\mathbf{P}}$:

$$\hat{\mathbf{P}} = \begin{cases} \hat{p} = \frac{1}{1+e^{-x}} & \hat{Y} = 0 \\ 1 - \hat{p} & \hat{Y} = 1 \end{cases}$$

with x being the input and $\hat{Y}$ as the predicted class. Then, the Cross Entropy of p and $\hat{p}$ can be defined as

$$(p-1)log(1-\hat{p}) - plog(\hat{p}).$$

This logistic definition of Cross Entropy has been widely used as loss function for a variety of tasks. But in the case of unbalanced data, it leads to biased classifiers(98, 99). Our experiments began by using this definition of loss and we indeed had a very biased classifier, which would leave the RFI-contaminated pixels undetected.

4.1.2.2 Balanced Cross Entropy Loss Function

There are ways to compensate for the skewness of the class distribution in the data. One way would be to use balanced Cross Entropy(98) by adding a weight factor γ for positive classes and $1-\gamma$ for negative classes to our sigmoidal Cross Entropy:

$$(p-1)(1-\gamma)log(1-\hat{p}) - \gamma plog(\hat{p}).$$

The downside of balanced Cross Entropy is the fact that the weight vector has to be set manually and although several strategies have been proposed, finding the optimum value for that factor is another challenge in and of itself. We tried to replace the Cross Entropy with balanced Cross Entropy and tried to optimize the weight factor, but still in different experiments with random data selection for input, test and validation the performance still fluctuated more than expected.

4.1.2.3 Focal Loss Function

Our next experiment was to use Focal Loss(99) as our loss function. The focal loss function will set the weight for each sample in a way that more common samples would get less weight and the more rare samples would get more weight. This strategy will punish the model more severely for misclassification of uncommon samples, e.g. RFI-contaminated pixels. Focal loss is defined as an extension of balanced Cross Entropy:

$$\alpha(\hat{p}-1)^{\gamma} plog(\hat{p}) + (\alpha-1)\hat{p}^{\gamma}(1-p)log(1-\hat{p})$$

We had with focal loss the same problem we had with balanced Cross Entropy, albeit with less severity.

4.1.2.4 Dice Loss Function

The problems manifesting with previously mentioned loss functions showed that to have an effective semantic segmentation method for RFI detection, a loss measure is needed that can take into account the performance at *object level* and not just pixel level. Therefore, after careful review of the literature we tested Dice Loss(100) and chose it as the basis of our loss function. Because of its performance advantages for imbalanced classes and its ability to distinguish between objects in the image, using Dice loss lead to better segmentation. We define our logistic Dice loss function as

$$\frac{p + \hat{p} - 2p\hat{p}}{p + \hat{p} + 1}$$

4.1.2.5 Hybrid Loss Function Based on Dice Similarity and Cross Entropy

Using Dice loss lead to better overall segmentation performance and a more stable output in different experiments. But using Dice loss, we were losing pixel level accuracy and there were lots of small pixelated misclassifications in the smooth areas of data. Therefore, since we were losing pixel-level accuracy of Cross Entropy, we combined these two loss measures to take advantage of both. In out implementation, for each minibatch we add the scalar value of the Dice loss to the tensor returned from Cross Entropy which will regulate the Cross Entropy on minibatch level. Using this formula lead to a high level of accuracy and a stable performance throughout different experiments.

4.2 Analysis of Results

The performance of the proposed method is measured using usual metrics of accuracy, F-1 and F-2 scores, and Dice index. Dice index let's us look at the model's performance not only on pixel level - measures like F-scores do - but also on the object level too, measuring apart from the accuracy on pixel level, how well can we detect connected bodies of RFI artifacts in our data.

4.2.1 Model Loss

Figures 4.1 and 4.2 show the Dice loss and Cross Entropy loss of the model during the training phase, both on training and validation sets. As can be seen, the model converges at epoch 50 based on Dice loss training error. The model reaches the Dice index of 82% on our point of convergence.

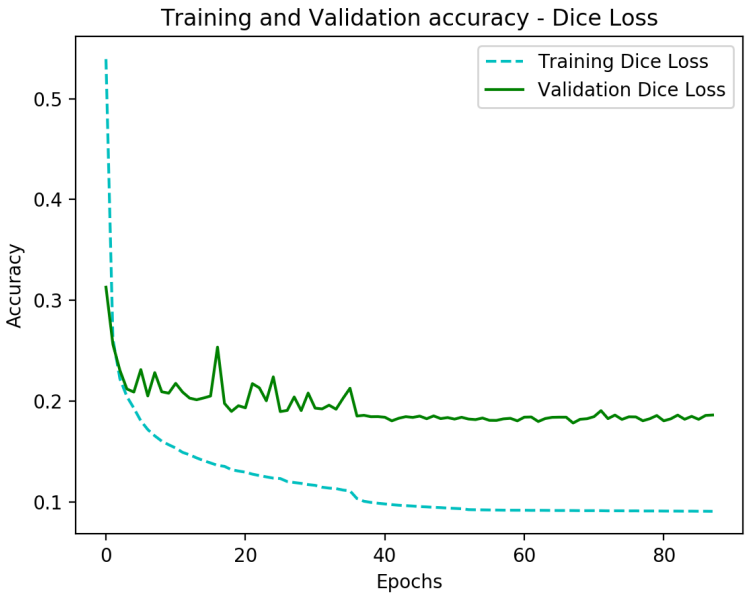

Figure 4.1: Model training and validation Dice loss diagram. The model converges at epoch 50.

4.2.2 Performance Metrics Analysis

Table 4.1 shows the performance metrics of the model while predicting on hold-out dataset. Average values of performance metrics after 10 runs are reported as our final result. In each run, the network was trained with a randomized training, validation and hold-out set out of our collected measurement sets. The results are from the model's prediction on the hold-out sets for each run.

Performance Metric	Value
Precision	0.931
Recall	0.825
Accuracy	0.995
F1-Score	0.874
F2-Score	0.844
Dice Index	0.89

Table 4.1: Average values of performance metrics after 10 runs, each run with a randomized, different training, validation and hold-out set. The results are from the model's prediction on the hold-out sets for each run

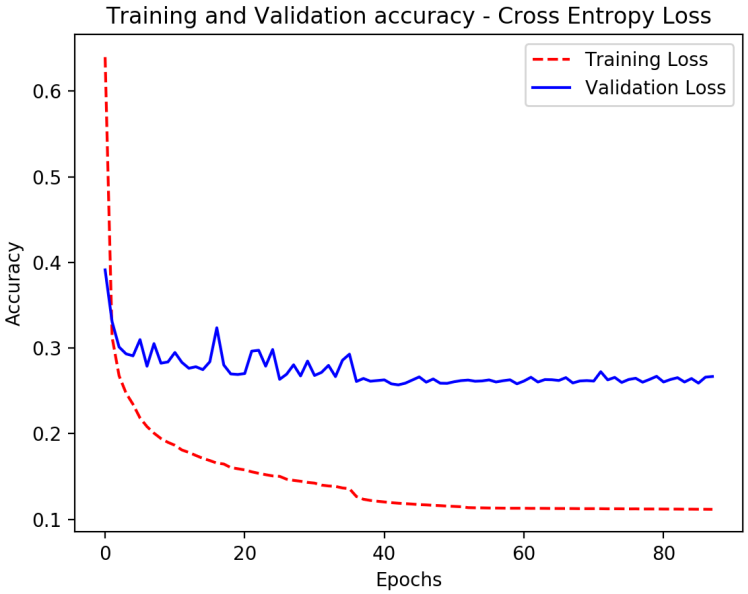

Figure 4.2: Model training and validation Cross Entropy loss diagram. The model converges at epoch 50.

4.2.2.1 Analysis of Precision and Recall

The high precision rate shows the accuracy of the model in predicting RFI areas, which is very high. It is expected, given the powerful model architecture used, the tuning performed on the network and the loss function, and the skewed nature of the dataset which pushed the model towards a maximum precision-zero recall state.

The recall is lower than precision, but that is expected behaviour; Because of the skewness of our dataset, the loss function is designed to heavily penalize the False Positives. But still, the recall rate indicates a rather low rate of false negatives.

4.2.2.2 Analysis of Accuracy, F-1 and F-2 Scores

The accuracy of the model is very high, but it can be a bit misleading since our dataset is skewed with most of the pixels being marked and not having RFI contamination and therefore a model with heavy bias would still achieve a high level of accuracy.

The F-1 and F-2 scores are a more reliable measure of the model's performance. In our case F-2 score is an even more balanced performance measure that F-1 score, since it gives a higher weight to recall compared to precision which will reveal more about the segmentation performance of our model given the class imbalance in our data.

As can be seen, both reported F1- and F2-scores are high which indicate good performance of the model.

4.2.2.3 Analysis of Dice Similarity

Last, we report a Dice similarity index of the model, which reports on the overlap between the RFI contaminated areas in ground truth and predicted segments. As mentioned in section 4.3.2.2, we see a high level of similarity between the hold-out RFI masks and the predicted masks. The reported Dice loss values from the model training phase show an increase in the model's performance and then a steady performance which leads to model convergence.

4.2.3 Comparison to Other Methods

Table 4.3 presents F1-scores reported in the literature for similar DL-based methods. The most similar method, (5) reports an F1-score of 0.85 on synthetic data, with the performance degrading rapidly when they shift to real world data - their actual F-1 score is not reported for real data, but they provide ROC graphs which show a massive degradation in performance. Another similar method, proposed by (14) reports an F1-score of only 0.17. Finally, (6) report F1-scores of 0.90, 0.70 and 0.28 for *narrowband*, *narrowband burst* and *brodband burst* RFI types in synthetic data, which overall can be considered inferior to the proposed method, both in terms of the use of synthetic data and the performance.

Method	F1-Score
(5)	0.85 (synthetic)
(14)	0.17
(6)	0.90, 0.70, 0.28
(88)	0.93
This research	0.841

Table 4.2: Comparison of the F1-Score values of similar works reported in the literature. (5) report a considerable degradation in performance when moving to real world data. The results reported by (6) are on synthetic data and belong to *narrowband*, *narrowband burst* and *brodband burst* RFI types.

It should be *strongly emphasized* that although the F-1 score numbers could provide a rough basis to compare the methods, the results are essentially un-comparable unless all the models are trained on exactly the same data under the same conditions. In most of the reviewed methods, the training data is either synthetic or from a very specific source(Belien

Observatory), while all of the data used for this project is real data coming from LOFAR. On the other hand, due to the flexibility of synthetic data generation synthetic data could potentially contain a broader range of RFI types; In this work, on the other hand, due to computational and practical limitations we only use data from one observation. The fact that this observation has been done during a specific time period and might not contain certain kinds of RFI - which could be present at other times not recorded by our selected observation - could be both a limiting factor and a barrier against comparison of this method with other similar ones. Even among methods that use synthetic data from the same source (the HIDE package), some sources (for example, (88)) do not mention the specifics of their data generation and therefore we still can not compare their results with other similar methods or even claim the superiority of their method.

4.2.4 The Effect of Data Imbalance and the Impact of Loss Function

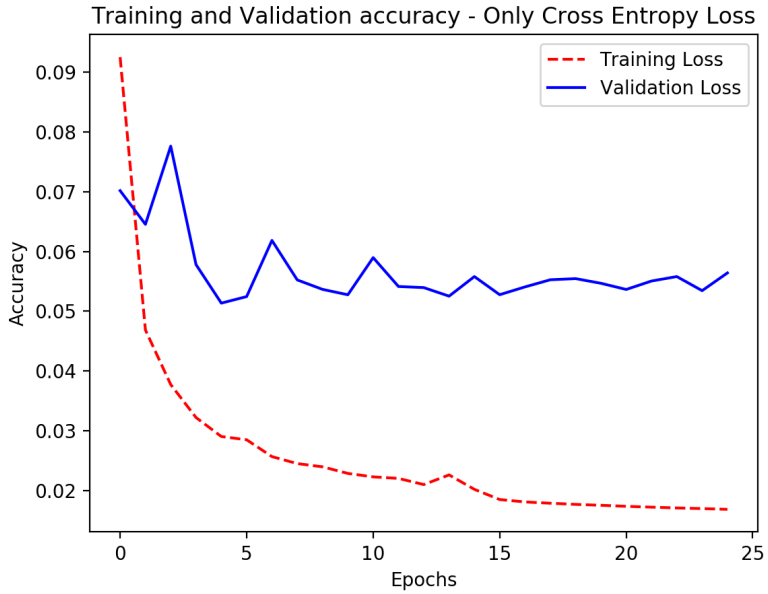

Figure 4.3: Model training and validation Cross Entropy loss diagram. The model converges at epoch 50.

In this section we present the analysis of our best performing model using only Cross Entropy loss function and compare the results to our final model which uses the Cross Entropy - Dice Loss combination loss metric. Figure 4.3 shows the training and validation loss of the model. As can be seen, unlike the final model which converged after 90 epochs, the model using the Cross Entropy loss function converges after 25 epochs. The validation

error starts very low and its curve flattens out around epoch 5 and stabilizes after epoch 15. This shows the effect of dataset imbalance, in which although the performance of the model based on our metrics is excellent, the model is unable to mark RFI in the input data accurately.

Table 4.3 compares the F-1 and F-2 score of the final model and the model with Cross Entropy loss. As can be observed, although the loss value of the Cross Entropy-based model is quite lower than what has been reported by the final model, in terms of F-1 and F-2 scores the performance of final model is superior since the model with Cross Entropy loss function fails to detect many of the RFI areas in the data.

Performance Metric	Final Model	Model with CE Loss
F1-Score	0.874	0.749
F2-Score	0.844	0.723

Table 4.3: Average values of performance metrics after 10 runs, each run with a randomized, different training, validation and hold-out set for two models: The final model and the model trained using only Cross Entropy as loss function. The results are from the model's prediction on the hold-out sets for each run

4.3 Analysis of Model Results

Figure 4.8 shows a sample output of the model from the hold-out dataset. The data is depicted in Figure 4.9. Next, the original RFI mask provided by the measurement set package from LOFAR LTA is plotted in Figure 4.10. The light pixels denote presence of RFI. *Predicted Mask* band in Figure 4.11 visualizes the actual values predicted for each pixel. The Figure 4.18 shows the classification error for this sample by plotting the absolute value of the subtraction of original values and predicted RFI mask values.

Numerous model outputs were analyzed to find out where the model's weakness in regards to detection of RFI lies which leads to lower recall values compared to our high precision level. We tried to measure the model's performance in presence of *narrow-band RFI*, *wide-band RFI* and *sporadic RFI*. One advantage of using synthetic data would have been that we would know exactly which kind of RFI we'd have in each data sample and measure the model's performance accurately. But since we don't have pixel-level classification of RFI types and there is a lot of overlap between different types of RFI, we resort to visual inspection.

Figure 4.19 shows several masks with different RFI types and the proposed system's classification error in presence of each. The samples have been selected from the lowest

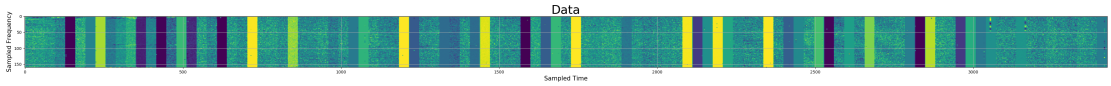

Figure 4.4: Data

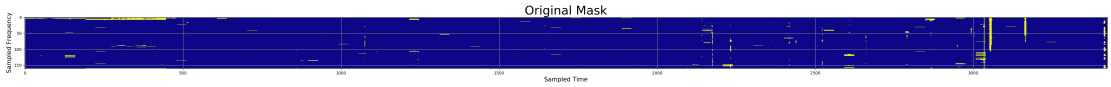

Figure 4.5: Original Mask

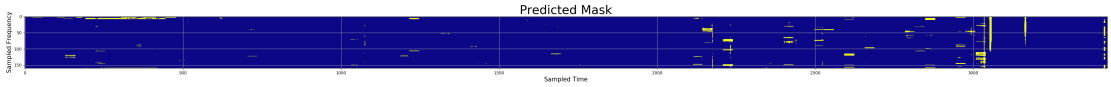

Figure 4.6: Predicted Mask

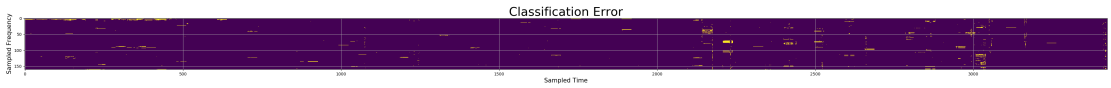

Figure 4.7: Classification Error

Figure 4.8: Sample network output selected out of the bottom 25% of validation set ordered by F-2 score

25% of the validation set, sorted by F-2 score. The analysis shows that the system is detecting narrow-band RFI and wide-band RFI much more accurately than sporadic RFI that happen in bursts. Also, the system is sensitive to both the duration of RFI and its frequency bandwidth. For example, in *Sample 1* the wide-band burst of RFI (seen as a horizontal line is not detected by the system). All the samples show the difficulties the system is having in detecting both burst RFI of either narrow or wide-band. They also show the system struggling with very narrow, sporadic RFI even if its time window is rather wide. The RFI that happens in narrow bands and in very short time windows is the hardest for the network to detect and visual inspection shows that many instances of classification error from our network belong to the sporadic types of RFI. It should be noted that the ground truth provided here is itself the result of statistical analysis on the reading data done using AOFlagger; although it needs further analysis both by an expert and by using ground truth data labeled and hand-curated by domain experts, we could also entertain the possibility of our proposed system detecting fluctuations in the input data which are actually RFI and have been overlooked by the statistical methods employed. Either way, this will be an interesting subject for future works.

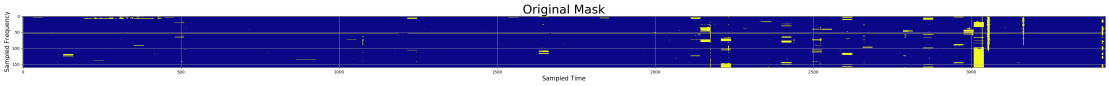

Figure 4.9: Sample 1 - Original Mask

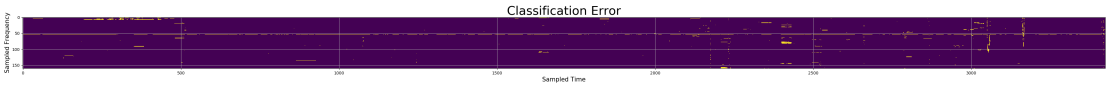

Figure 4.10: Sample 1 - Classification Error

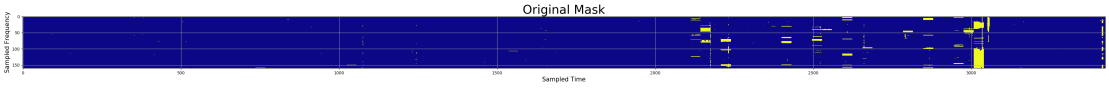

Figure 4.11: Sample 2 - Original Mask

Figure 4.12: Sample 2 - Classification Error

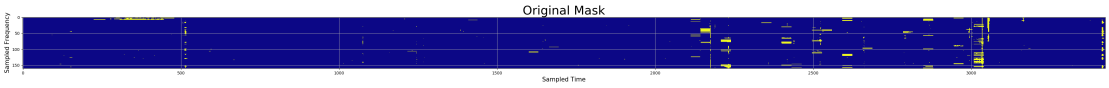

Figure 4.13: Sample 3 - Original Mask

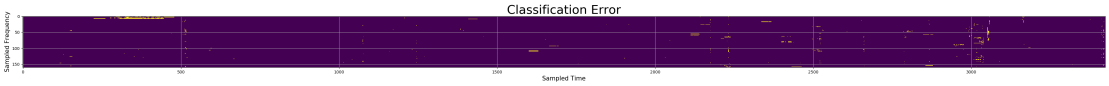

Figure 4.14: Sample 3 - Classification Error

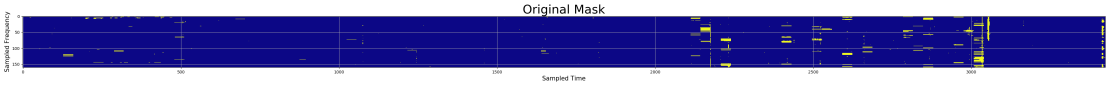

Figure 4.15: Sample 4 - Original Mask

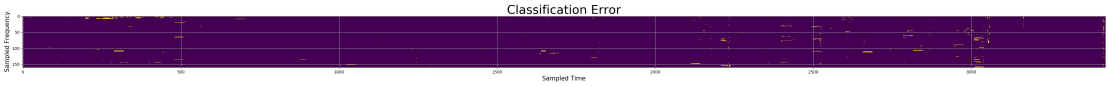

Figure 4.16: Sample 4 - Classification Error

Figure 4.17: Sample 5 - Original Mask

Figure 4.18: Sample 5 - Classification Error

Figure 4.19: 5 hand-picked sample network outputs selected out of the bottom 25% of validation set ordered by F-2 score

4.4 Future Work

In this research, we showcased a deep-learning based approach to identify RFI in radioastronomy. We used real-world data from LOFAR LTA to train and validate the network's performance and we show here highly accurate results, comparable and/or superior to available methods.

One obvious improvement over this work would be to test newer state of the art semantic segmentation methods and compare the results to see which model is more accurate. Intuitively, more refined approaches to this family of problems are under development and would perform better at predicting RFI in radioastronomy data.

Another extension of this work would be in the area of performance. As performant as the model is, if it can not perform the segmentation in a timely fashion - under circumstances dictated by the LOFAR online data processing pipeline - it would lose part of its applicability. We couldn't compare the performance of the model to the methods performed in ASTRON since going over the details of those methods and replicating the data processing environment was outside of the scope of this research. Therefore one avenue of extension on this work would be to research the methods currently in use to detect RFI at ASTRON and define metrics that would accurately measure the performance of those methods. This potential research would comprehensively compare the results of these methods with our proposed model to see how this DL-based model fares against non DL-based methods in terms of performance. This would include checking on the performance advantages of different python interpreters and Linux kernels, DL frameworks and possible improvements to the existing pipeline.

One of the first decisions that we made for this project was to take the absolute values of the complex data from readings provided by LOFAR an. While that provides certain advantages in terms of simplicity and computational costs, and we have proved that even under this assumption that would lead to the loss of information we're still able to accurately detect RFI in radio astronomy data, one area of improvement to this research would be to come up with possible transformations or redesign of the pipeline so that we can use the reading data in their complex form.

On computational costs of the training phase of the model, we have to assume that at some point in time, due to changes in RFI sources, the current model would be deprecated and we need to retrain our model on new data. Figuring out when we should actually retrain the model and accept the costs of this process instead of relying on human expertise or a non-dynamic approach, e.g. a fixed interval, would be beneficial to this research.

Lastly, in this research we're not considering the temporal aspects of the RFI. Intuitively we can argue that some RFI sources would have identifiable temporal characteristics. Furthermore, there is already a body of research on temporal semantic segmentation available in literature. Therefore, it would be worth investigation to see if we can add the time element to this pipeline and make the predictions more accurate. This research topic would include investigating the possibility of forming A Priori knowledge about the RFI-free state of different areas of the sky, and detect changes in the behavior of the RFI sources, which might indicate the deprecation of the trained model itself.

References

[1] Mark Nicol. **Measuring the Universe: A Multiwavelength Perspective, by George H. Rieke: Scope: review. Level: postgraduate, advanced undergraduate, scientist, engineers.**, 2013. vii, 6

[2] Michael P van Haarlem, Michael W Wise, AW Gunst, George Heald, John P McKean, Jason WT Hessels, A Ger de Bruyn, Ronald Nijboer, John Swinbank, Richard Fallows, et al. **LOFAR: The low-frequency array**. *Astronomy & astrophysics*, **556**:A2, 2013. vii, 7, 8

[3] Jonathan Long, Evan Shelhamer, and Trevor Darrell. **Fully convolutional networks for semantic segmentation**. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. vii, 12, 13

[4] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. **U-net: Convolutional networks for biomedical image segmentation**. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. vii, 3, 13, 17, 18, 19, 22

[5] Joel Akeret, Chihway Chang, Aurelien Lucchi, and Alexandre Refregier. **Radio frequency interference mitigation using deep convolutional neural networks**. *Astronomy and computing*, **18**:35–39, 2017. ix, 1, 2, 3, 4, 19, 33

[6] Joshua Kerrigan, Paul La Plante, Saul Kohn, Jonathan C Pober, James Aguirre, Zara Abdurashidova, Paul Alexander, Zaki S Ali, Yanga Balfour, Adam P Beardsley, et al. **Optimizing sparse RFI prediction using deep learning**. *Monthly Notices of the Royal Astronomical Society*, **488**(2):2605–2615, 2019. ix, 2, 19, 33

REFERENCES

[7] AR Offringa, AG De Bruyn, M Biehl, S Zaroubi, G Bernardi, and VN Pandey. **Post-correlation radio frequency interference classification methods**. *Monthly Notices of the Royal Astronomical Society*, **405**(1):155–167, 2010. 2, 8, 9

[8] Juan Zhao, Xiaolei Zou, and Fuzhong Weng. **WindSat radio-frequency interference signature and its identification over Greenland and Antarctic**. *IEEE Transactions on Geoscience and Remote Sensing*, **51**(9):4830–4839, 2013. 2, 9

[9] AR Offringa, JJ Van De Gronde, and JBTM Roerdink. **A morphological algorithm for improving radio-frequency interference detection**. *Astronomy & astrophysics*, **539**:A95, 2012. 2, 8, 9

[10] AR Offringa, RB Wayth, N Hurley-Walker, DL Kaplan, N Barry, AP Beardsley, ME Bell, G Bernardi, JD Bowman, F Briggs, et al. **The low-frequency environment of the Murchison Widefield Array: radio-frequency interference analysis and mitigation**. *Publications of the Astronomical Society of Australia*, **32**, 2015. 2

[11] Joel Akeret, Sebastian Seehars, Chihway Chang, Christian Monstein, Adam Amara, and Alexandre Refregier. **HIDE & SEEK: End-to-end packages to simulate and process radio survey data**. *Astronomy and Computing*, **18**:8–17, 2017. 2, 4, 19, 20

[12] PA Fridman and WA Baan. **RFI mitigation methods in radio astronomy**. *Astronomy & Astrophysics*, **378**(1):327–344, 2001. 2, 8

[13] Licheng Jiao, Fan Zhang, Fang Liu, Shuyuan Yang, Lingling Li, Zhixi Feng, and Rong Qu. **A Survey of Deep Learning-Based Object Detection**. *IEEE Access*, **7**:128837–128868, 2019. 2

[14] Paul Ray Burd, Karl Mannheim, Tobias März, Jonas Ringholz, Alexander Kappes, and Matthias Kadler. **Detecting radio frequency interference in radio-antenna arrays with the recurrent neural network algorithm**. *Astronomische Nachrichten*, **339**(5):358–362, 2018. 2, 19, 33

[15] Rob V van Nieuwpoort. **Towards exascale real-time RFI mitigation**. In *2016 Radio Frequency Interference (RFI)*, pages 69–74. IEEE, 2016. 3

[16] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. **Learning from massive noisy labeled data for image classification**. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2691–2699, 2015. 3

[17] Yi-zhou Lin, Zhen-hua Nie, and Hong-wei Ma. **Structural damage detection with automatic feature-extraction through deep learning**. *Computer-Aided Civil and Infrastructure Engineering*, **32**(12):1025–1046, 2017. 3

[18] Daniel George and EA Huerta. **Deep Learning for real-time gravitational wave detection and parameter estimation: Results with Advanced LIGO data**. *Physics Letters B*, **778**:64–70, 2018. 3

[19] Hao Ye, Geoffrey Ye Li, and Biing-Hwang Juang. **Power of deep learning for channel estimation and signal detection in OFDM systems**. *IEEE Wireless Communications Letters*, **7**(1):114–117, 2017. 3

[20] Mingfei Liu, Wei Wu, Zhenghui Gu, Zhuliang Yu, FeiFei Qi, and Yuanqing Li. **Deep learning based on Batch Normalization for P300 signal detection**. *Neurocomputing*, **275**:288–297, 2018. 3

[21] Sharada P Mohanty, David P Hughes, and Marcel Salathé. **Using deep learning for image-based plant disease detection**. *Frontiers in plant science*, **7**:1419, 2016. 3

[22] Zhaowei Cai, Mohammad Saberian, and Nuno Vasconcelos. **Learning complexity-aware cascades for deep pedestrian detection**. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3361–3369, 2015. 3

[23] Young-Jin Cha, Wooram Choi, and Oral Büyüköztürk. **Deep learning-based crack damage detection using convolutional neural networks**. *Computer-Aided Civil and Infrastructure Engineering*, **32**(5):361–378, 2017. 3

[24] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. **Deep learning**. *nature*, **521**(7553):436–444, 2015. 3

[25] Grégory Hellbourg. *Radio Frequency Interference spatial processing for modern radio telescopes*. PhD thesis, 2014. 3

[26] George B Field and Eric J Chaisson. **The invisible universe. Probing the frontiers of astrophysics**. 1984. 5

[27] Patrick Moore and Robin Rees. *Patrick Moore's data book of astronomy*. Cambridge University Press, 2014. 5

[28] George H. Rieke. *Interferometry and aperture synthesis*, page 266–292. Cambridge University Press, 2012. 5

[29] **LOFAR System Capabilities: Frequency, subband selection, and RFI**. 8

[30] Xuelei Chen. **The Tianlai project: a 21cm cosmology experiment**. In *International Journal of Modern Physics: Conference Series*, **12**, pages 256–263. World Scientific, 2012. 8

[31] Peter E Dewdney, Peter J Hall, Richard T Schilizzi, and T Joseph LW Lazio. **The square kilometre array**. *Proceedings of the IEEE*, **97**(8):1482–1496, 2009. 8

[32] MI Krauss, AM Soderberg, L Chomiuk, BA Zauderer, A Brunthaler, MF Bietenholz, RA Chevalier, Claes Fransson, and M Rupen. **Expanded Very Large Array observations of the radio evolution of SN 2011dh**. *The Astrophysical Journal Letters*, **750**(2):L40, 2012. 8

[33] Jonathan C Pober, Adrian Liu, Joshua S Dillon, James E Aguirre, Judd D Bowman, Richard F Bradley, Chris L Carilli, David R DeBoer, Jacqueline N Hewitt, Daniel C Jacobs, et al. **What next-generation 21 cm power spectrum measurements can teach us about the epoch of reionization**. *The Astrophysical Journal*, **782**(2):66, 2014. 8, 19

[34] Heino D Falcke, Michiel P van Haarlem, A Ger de Bruyn, Robert Braun, Huub JA Röttgering, Benjamin Stappers, Wilfried HWM Boland, Harvey R Butcher, Eugène J de Geus, Leon V Koopmans, et al. **A very brief description of LOFAR–the Low Frequency Array**. *Proceedings of the International Astronomical Union*, **2**(14):386–387, 2006. 8

[35] MA Brentjens. **Interference due to wind turbines at 30–200 MHz**. In *2016 Radio Frequency Interference (RFI)*, pages 7–10. IEEE, 2016. 8

[36] Matthew Braunstein, James M Ralston, and David A Sparrow. **Signal processing approaches to radio frequency interference (RFI) suppression**. In *Algorithms for Synthetic Aperture Radar Imagery*, **2230**, pages 190–208. International Society for Optics and Photonics, 1994. 8

[37] Andrew G Dempster and Ediz Cetin. **Interference localization for satellite navigation systems**. *Proceedings of the IEEE*, **104**(6):1318–1326, 2016. 8

[38] Cornelis Johannes Wolfaardt. *Machine learning approach to radio frequency interference (RFI) classification in radio astronomy*. PhD thesis, Stellenbosch: Stellenbosch University, 2016. 8, 10

[39] Roslan Umar, Zamri Zainal Abidin, Zainol Abidin Ibrahim, Mohd Saiful Rizal Hassan, Zulfazli Rosli, and Zety Shahrizat Hamidi. **Population density effect on radio frequencies interference (RFI) in radio astronomy**. In *AIP Conference Proceedings*, **1454**, pages 39–42. American Institute of Physics, 2012. 8

[40] AR Offringa, AG de Bruyn, and S Zaroubi. **Post-correlation filtering techniques for off-axis source and RFI removal**. *Monthly Notices of the Royal Astronomical Society*, **422**(1):563–580, 2012. 8

[41] Noppasin Niamsuwan, Joel T Johnson, and Steven W Ellingson. **Examination of a simple pulse-blanking technique for radio frequency interference mitigation**. *Radio Science*, **40**(5), 2005. 8

[42] AR Offringa, AG de Bruyn, S Zaroubi, and M Biehl. **A LOFAR RFI detection pipeline and its first results**. *arXiv preprint arXiv:1007.2089*, 2010. 9

[43] R Weber, C Faye, F Biraud, and J Dansou. **Spectral detector for interference time blanking using quantized correlator**. *Astronomy and Astrophysics Supplement Series*, **126**(1):161–167, 1997. 9

[44] WA Baan, PA Fridman, and RP Millenaar. **Radio frequency interference mitigation at the westerbork synthesis radio telescope: Algorithms, test observations, and system implementation**. *The Astronomical Journal*, **128**(2):933, 2004. 9

[45] Gelu M Nita, Dale E Gary, Zhiwei Liu, Gordon J Hurford, and Stephen M White. **Radio frequency interference excision using spectral-domain statistics**. *Publications of the Astronomical Society of the Pacific*, **119**(857):805, 2007. 9

[46] Benjamin Winkel, Juergen Kerp, and Stephan Stanko. **RFI detection by automated feature extraction and statistical analysis**. *Astronomische Nachrichten: Astronomical Notes*, **328**(1):68–79, 2007. 9

[47] Olorato Mosiane, Nadeem Oozeer, Arun Aniyan, and Bruce A Bassett. **Radio Frequency Interference Detection using Machine Learning.** In *IOP Conference Series: Materials Science and Engineering*, **198**, page 012012. IOP Publishing, 2017. 10

[48] Cornelis Johannes Wolfaardt, David Davidson, and Thomas Niesler. **Statistical classification of radio frequency interference (RFI) in a radio astronomy environment**. In *2016 Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference (PRASA-RobMech)*, pages 1–5. IEEE, 2016. 10

[49] Olorato Mosiane, Nadeem Oozeer, and Bruce A Bassett. **Machine learning for radio frequency interference mitigation using polarization**. In *2017 IEEE Radio and Antenna Days of the Indian Ocean (RADIO)*, pages 1–2. IEEE, 2017. 10

[50] Gary Doran. **Characterizing interference in radio astronomy observations through active and unsupervised learning**. 2013. 10

[51] Basemah Alshemali and Jugal Kalita. **Improving the Reliability of Deep Neural Networks in NLP: A Review**. *Knowledge-Based Systems*, page 105210, 2019. 10

[52] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. **Deep learning for time series classification: a review**. *Data Mining and Knowledge Discovery*, **33**(4):917–963, 2019. 10

[53] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. **Object detection with deep learning: A review**. *IEEE transactions on neural networks and learning systems*, **30**(11):3212–3232, 2019. 10

[54] Jose Bernal, Kaisar Kushibar, Daniel S Asfaw, Sergi Valverde, Arnau Oliver, Robert Martí, and Xavier Lladó. **Deep convolutional neural networks for brain image analysis on magnetic resonance imaging: a review**. *Artificial intelligence in medicine*, **95**:64–81, 2019. 10

[55] Xin Yi, Ekta Walia, and Paul Babyn. **Generative adversarial network in medical imaging: A review**. *Medical image analysis*, page 101552, 2019. 10

[56] Tongxue Zhou, Su Ruan, and Stéphane Canu. **A review: Deep learning for medical image segmentation using multi-modality fusion**. *Array*, page 100004, 2019. 10

[57] Chen Chen, Chen Qin, Huaqi Qiu, Giacomo Tarroni, Jinming Duan, Wenjia Bai, and Daniel Rueckert. **Deep learning for cardiac image segmentation: A review**. *arXiv preprint arXiv:1911.03723*, 2019. 10

[58] Saed Asgari Taghanaki, Kumar Abhishek, Joseph Paul Cohen, Julien Cohen-Adad, and Ghassan Hamarneh. **Deep Semantic Segmentation of Natural and Medical Images: A Review**. *arXiv preprint arXiv:1910.07655*, 2019. 10

[59] Xiaolong Liu, Zhidong Deng, and Yuhan Yang. **Recent progress in semantic image segmentation**. *Artificial Intelligence Review*, **52**(2):1089–1106, 2019. 10, 11

[60] Michael Treml, José Arjona-Medina, Thomas Unterthiner, Rupesh Durgesh, Felix Friedmann, Peter Schuberth, Andreas Mayr, Martin Heusel, Markus Hofmarcher, Michael Widrich, et al. **Speeding up semantic segmentation for autonomous driving**. In *MLITS, NIPS Workshop*, **2**, page 7, 2016. 11

[61] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. **V-net: Fully convolutional neural networks for volumetric medical image segmentation**. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 565–571. IEEE, 2016. 11

[62] Mohammad D Hossain and Dongmei Chen. **Segmentation for object-based image analysis (obia): A review of algorithms and challenges from remote sensing perspective**. *ISPRS Journal of Photogrammetry and Remote Sensing*, **150**:115–134, 2019. 11

[63] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. **Image Segmentation Using Deep Learning: A Survey**. *arXiv preprint arXiv:2001.05566*, 2020. 11

[64] Dengsheng Lu and Qihao Weng. **A survey of image classification methods and techniques for improving classification performance**. *International journal of Remote sensing*, **28**(5):823–870, 2007. 11

[65] Brian Fulkerson, Andrea Vedaldi, and Stefano Soatto. **Class segmentation and object localization with superpixel neighborhoods**. In *2009 IEEE 12th international conference on computer vision*, pages 670–677. IEEE, 2009. 11

[66] Himani S Parekh, Darshak G Thakore, and Udesang K Jaliya. **A survey on object detection and tracking methods**. *International Journal of Innovative Research in Computer and Communication Engineering*, **2**(2):2970–2978, 2014. 11

[67] Gustavo Carneiro and Nuno Vasconcelos. **Formulating semantic image annotation as a supervised learning problem**. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, **2**, pages 163–168. IEEE, 2005. 12

[68] Celine Hudelot, Nicolas Maillot, and Monique Thonnat. **Symbol grounding for semantic image interpretation: from image data to semantics**. In *Tenth IEEE International Conference on Computer Vision Workshops (ICCVW'05)*, pages 1875–1875. IEEE, 2005. 12

[69] Jianping Fan, Yuli Gao, and Hangzai Luo. **Multi-level annotation of natural scenes using dominant image components and semantic concepts**. In *Proceedings of the 12th annual ACM international conference on Multimedia*, pages 540–547, 2004. 12

[70] Jiebo Luo, Andreas E Savakis, and Amit Singhal. **A Bayesian network-based framework for semantic image understanding**. *Pattern recognition*, **38**(6):919–934, 2005. 12

[71] Alvaro Pardo. **Semantic image segmentation using morphological tools**. In *Proceedings. International Conference on Image Processing*, **2**, pages II–II. IEEE, 2002. 12

[72] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. **Imagenet classification with deep convolutional neural networks**. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 12

[73] Karen Simonyan and Andrew Zisserman. **Very deep convolutional networks for large-scale image recognition**. *arXiv preprint arXiv:1409.1556*, 2014. 12

[74] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. **Deep residual learning for image recognition**. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 12

[75] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. **Going deeper with convolutions**. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 12

[76] Jing Huang and Suya You. **Point cloud labeling using 3d convolutional neural network**. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2670–2675. IEEE, 2016. 13

[77] Sergey Zagoruyko, Adam Lerer, Tsung-Yi Lin, Pedro O Pinheiro, Sam Gross, Soumith Chintala, and Piotr Dollár. **A multipath network for object detection**. *arXiv preprint arXiv:1604.02135*, 2016. 13

[78] Pedro HO Pinheiro and Ronan Collobert. **Recurrent convolutional neural networks for scene labeling**. In *31st International Conference on Machine Learning (ICML)*, number CONF, 2014. 13

[79] Evan Shelhamer, Kate Rakelly, Judy Hoffman, and Trevor Darrell. **Clockwork convnets for video semantic segmentation**. In *European Conference on Computer Vision*, pages 852–868. Springer, 2016. 13

[80] Richard HR Hahnloser, Rahul Sarpeshkar, Misha A Mahowald, Rodney J Douglas, and H Sebastian Seung. **Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit**. *Nature*, **405**(6789):947–951, 2000. 14

[81] Prajit Ramachandran, Barret Zoph, and Quoc V Le. **Searching for activation functions**. *arXiv preprint arXiv:1710.05941*, 2017. 14

[82] Sergey Ioffe and Christian Szegedy. **Batch normalization: Accelerating deep network training by reducing internal covariate shift**. *arXiv preprint arXiv:1502.03167*, 2015. 15

[83] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016. 16

[84] Pierre Baldi and Peter J Sadowski. **Understanding dropout**. In *Advances in neural information processing systems*, pages 2814–2822, 2013. 16

[85] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. **Dropout: a simple way to prevent neural networks from overfitting**. *The journal of machine learning research*, **15**(1):1929–1958, 2014. 16

[86] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. **A review of recurrent neural networks: LSTM cells and network architectures**. *Neural computation*, **31**(7):1235–1270, 2019. 19

[87] Sepp Hochreiter and Jürgen Schmidhuber. **Long short-term memory**. *Neural computation*, **9**(8):1735–1780, 1997. 19

[88] Zhicheng Yang, Ce Yu, Jian Xiao, and Bo Zhang. **Deep residual detection of radio frequency interference for FAST**. *Monthly Notices of the Royal Astronomical Society*, **492**(1):1421–1431, 2020. 20, 33, 34

[89] Md Zahangir Alom, Mahmudul Hasan, Chris Yakopcic, Tarek M Taha, and Vijayan K Asari. **Recurrent residual convolutional neural network based on u-net (r2u-net) for medical image segmentation**. *arXiv preprint arXiv:1802.06955*, 2018. 20

[90] Zhengxin Zhang, Qingjie Liu, and Yunhong Wang. **Road extraction by deep residual u-net**. *IEEE Geoscience and Remote Sensing Letters*, **15**(5):749–753, 2018. 20

[91] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. **Proceedings of the IEEE conference on computer vision and pattern recognition**. *Convolutional Pose Machines*, pages 4724–4732, 2016. 20

[92] Rendong Nan. **Five hundred meter aperture spherical radio telescope (FAST)**. *Science in China series G*, **49**(2):129–148, 2006. 20

[93] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. **Tensorflow: A system for large-scale machine learning**. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016. 25

[94] Nikhil Ketkar. **Introduction to keras**. In *Deep learning with Python*, pages 97–111. Springer, 2017. 25

[95] Casacore Team. **casacore: Suite of C++ libraries for radio astronomy data processing**. *ascl*, pages ascl–1912, 2019. 25

[96] Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. **scikit-image: image processing in Python**. *PeerJ*, **2**:e453, 2014. 25

[97] Nils Bjorck, Carla P Gomes, Bart Selman, and Kilian Q Weinberger. **Understanding batch normalization**. In *Advances in Neural Information Processing Systems*, pages 7694–7705, 2018. 27

[98] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. **Class-balanced loss based on effective number of samples**. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9268–9277, 2019. 29

[99] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. **Focal loss for dense object detection**. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 29

[100] Xiaoya Li, Xiaofei Sun, Yuxian Meng, Junjun Liang, Fei Wu, and Jiwei Li. **Dice Loss for Data-imbalanced NLP Tasks**. *arXiv preprint arXiv:1911.02855*, 2019. 30