Universiteit Utrecht

Opleiding Natuur- en Sterrenkunde

Sparse, deep neural networks for the early detection of gravitational waves from binary neutron stars

Bachelor Thesis

Koen van Lieshout

Supervisors:

Dr. Sarah Caudill
Nikhef, GRASP

Grégory Baltus
STAR Institute ULiège

Justin Janquart
Nikhef, GRASP

Melissa Lopez
Nikhef, GRASP

June 16, 2021

Abstract

In the field of gravitational-wave detection, machine learning has become a part of the landscape. In this work, we build upon a previous work [1], which gave a proof of concept of the use of convolutional neural network in the detection of the early phases of an inspiraling binary neutron star. Obtaining such early detection would allow one to send out alerts to astronomers, so that other phases of the merger can be observed using other messengers In this work, we adapt two of the latest techniques from other neural network fields, called *inception modules*, adapted from Google Inception-Resnet-v2 and *dilation*, as implemented in Wavenet. Here, for the first time, we revise those tools to be used for one dimensional data. We show they can lead to improvements in the early detection of binary neutron stars.

Contents

1 Introduction **1**

2 Gravitational waves **2**
2.1 What is a gravitational wave? 2
2.2 Detectors 5
2.3 Compact binary coalescence 7
2.4 Matched filtering for gravitational wave detection 9
2.5 Multi-messenger astronomy 12

3 Methodology **12**
3.1 Machine learning 13
3.1.1 Loss function 15
3.1.2 Gradient descent 17
3.1.3 Optimizers 19
3.1.4 Convolutional neural networks 19
3.2 Pooling layers 22
3.3 Dilation 23
3.4 Inception modules 24
3.5 Architecture of the network 26
3.6 Training of the network 29
3.7 Obtaining results 32

4 Results **33**

5 Conclusion and Discussion **39**

A Appendix **45**
A.1 Optimising the network without dilation 45
A.2 Optimising the dilation 47

1 Introduction

When publishing his theory on general relativity in 1915 [2], Einstein predicted the existence of gravitational waves. 100 years later, in September 2015, the first observations of gravitational waves were made by the LIGO-Virgo collaboration [3]. A collaboration revolving around three large interferometers in North America and Europe. Soon after, in 2017, the first gravitational waves coming from a binary neutron star merger were detected. This detection allowed astronomers to observe a binary neutron star merger in the electromagnetic spectrum for the first time [4]. Since then, with the advances in technology, over 50 detections of gravitational waves have been made [5, 6] Currently, the detection of gravitational waves is done by using a process called matched filtering. It uses a library of templates and tries to match these templates with incoming gravitational-wave signals [7]. However, matched filtering is relatively slow and with the ever increasing amount of detected pairs, due to improvements in the detectors, faster techniques will be necessary to do the initial detection [8–10].

Over the last years, neural networks have been developed to address this. The goal of neural networks is to detect gravitational waves faster and recently, an additional goal is to detect waves in earlier stages of merger (early warning) [1, 11]. These neural networks do not directly match the aforementioned templates to the data, but are instead trained on templates or real gravitational wave data in order to be able to recognise gravitational waves [1, 12]. The most common neural networks were convolutional neural networks, which had previously been used for image classification [13].

This work builds upon the work presented in [1], where a proof of concept was given for the early warning detection of gravitational-waves. In [1], convolutional neural networks were used for the detection of binary neutron stars. They have shown that it is possible to detect waves from realistically simulated binary neutron stars in the inspiral phase using neural networks. They were one of the first two papers that focused on early warning using neural networks and we will continue their work.

Currently, there is a lot of research being done on how to improve the accuracy of these networks [14]. Two techniques which resulted in performance increases in image- and audio-recognition respectively were selected [15–17]. The first technique is the use of inception modules, as previously introduced in *Google Resnet v2* [18]. Inception modules have resulted in significantly improved results in the field of image recognition, by allowing networks to

have more layers. The second technique selected for this work is known as dilation. Dilation introduces more contextual information to the network. It was used in *Wavenet* [16], a network used in text-to-speech generation and led to significant improvements here.

In this work, we will implement the previously mentioned techniques in combination with the neural networks from [1]. The Resnet blocks will be adapted to 1D for the first time. We will use the same training techniques and data set in order to be able to compare the results in the end. Through this, the research question of this work becomes

Research question: How can the adaptation of inception modules and dilation into convolutional neural networks lead to increased sensitivity in the detection of the early inspiral phase of a binary neutron star?

2 Gravitational waves

2.1 What is a gravitational wave?

In 1905 Einstein published his theory on special relativity [19]. In this theory, he proved that, in a vacuum, nothing can travel faster than light. This theory was revolutionary, but it also caused several problems with the physics as we knew it up until that point. One of the big problems came from the law of gravity. Newton's law of gravity said that gravity was an instantaneous force. This meant that information traveled between to objects faster than the speed of light, which, according to special relativity, is not possible. In 1915 Einstein found a solution through his theory on general relativity [2]. Here, Einstein said that the effect of gravity is due to a deformation of space-times. Any object with mass curves space-time. Other objects in the vicinity are not pulled by an instantaneous gravitational force, but instead travel through a curved space-time, causing their trajectory to follow a geodesic, a straight line in curved space-time. An illustration can be seen in Fig 1. The denser an object is, the more curvature it causes and the more the trajectory of objects in the vicinity will be altered. When a dense object suddenly accelerates, it will cause a wave in space-time, which travels at the speed of light. These waves are called gravitational waves.

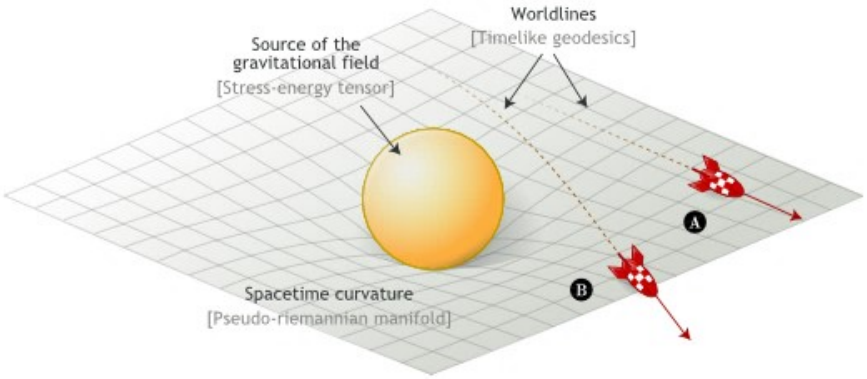

Figure 1: A rocket being influenced by gravity, causing the path to curve following the curvature of space-time from [20].

As space-time is hard to deform, only a few objects are dense enough for us to measure the gravitational waves in the current detectors. Only neutron stars and black holes are dense enough to be detected. We need a system where two dense objects rotate around each other. Such systems are called binary neutron stars (BNS), binary black holes (BBH) or neutron star black hole binaries (NSBH).

We now want to find the effect of gravitational waves on space-time when they reach us. Because these objects are so rare, we can assume that the distance between us and the binary system is much larger than the characteristic size of the system. When they reach us, the gravitational fields are therefore weak. Using this assumption, we can define the space-time metric $g_{\mu\nu}$ as the Minkowski metric for flat space (η, plus a small perturbation $h_{\mu\nu}$ caused by the gravitational wave:

$$g_{\mu\nu} = \eta_{\mu\nu} + h_{\mu\nu}\,, \tag{1}$$

where $|h_{\mu\nu}| << 1$

Outside of the source, because the energy and flux is zero here, $\Box\bar{h}_{\mu\nu} = 0$ This equation has plane-wave solutions of the form

$$h_{ij} = A_{ij}(\overrightarrow{k})cos(\omega t - \overrightarrow{k} \cdot \overrightarrow{x}))\,, \tag{2}$$

where ω is the wave frequency and k is the unit vector in the direction of propagation. A is the amplitude of the wave at time t and position x.

If we look at a gravitational wave as a plane wave propagating in the z-direction, caused by two object orbiting each other circularly in the xy-plane, $\overrightarrow{k} \cdot \overrightarrow{x} = \omega z/c$ and

$$h_{ij} = \begin{pmatrix} h_+ & h_\times & 0 \\ h_\times & h_+ & 0 \\ 0 & 0 & 0 \end{pmatrix} cos(\omega(t - z/c)), \tag{3}$$

where h_+ and $h_\times$ are the amplitudes of the "plus" and "cross polarisations of the wave. Unlike electromagnetic waves, gravitational wave polarisations are at a 45 degree angle. The plus polarisation is the polarisation in the horizontal and vertical direction [21].

When the gravitational wave reaches a line segment $ds^2 = g_{\mu\nu}dx^\mu dx^\nu$. It influences that segment as [22]:

$$ds^2 = -c^2dt^2 + (1 + h_+cos[\omega(t - z/c)])dx^2 + (1 - h_+cos[\omega(t - z/c)])dy^2 \\ +2h_\times cos[\omega(t - z/c)]dxdy + dz^2$$

If we set $h_\times = 0$, but $h_+ \neq 0$ in Eq.(2.1), we get

$$ds^2 = -c^2dt^2+(1+h_+cos[\omega(t-z/c)])dx^2+(1-h_+cos[\omega(t-z/c)]dy^2+dz^2\,. \tag{4}$$

It becomes clear that, with time, only the cosine terms vary. The x- and y-distance change periodically, but with opposite sign. When space stretches in the x-direction, it compresses in the y direction. This is shown in Fig.2.

Similarly, when $h_+ = 0$, but $h_\times \neq 0$,

$$ds^2 = dx^2 + dy^2 + 2h_xcos[\omega(t - z/c)]dxdy + dz^2 + dz^2\,. \tag{5}$$

This equation becomes clearer when rotating the x and y axes by 45°. Noting the rotated axes as x' and y' [21],

$$ds^2 = -c^2dt^2 + (1 + h_xcos[\omega(t - z/c)])dx'^2 + (1 - h_xcos[\omega(t - z/c)])dy'^2 + dz^2. \tag{6}$$

If we rotate the axes by 45 degrees, the end result is the same as we saw in Eq.(6), giving the same effect in the rotated axes. The system will experience the same behaviour in the diagonal direction, as seen in Fig. 2 [22]

As can be seen in Fig. 2, the gravitational wave periodically stretches and compresses space-time as it passes through. In the next section, we will explain how we this characteristic can be uses to our advantage when trying to detect gravitational waves [22].

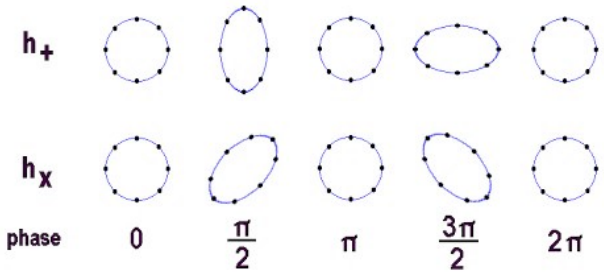

Figure 2: Influence of a gravitational wave on a ring of particles. We see that there is a periodical stretching of the circle with a rotation of 45° between the plus and cross polarizations.

2.2 Detectors

The detection of gravitational waves is done by using a laser interferometer [23], which uses the stretching of the space-time by the gravitational wave when it passes. In a laser interferometer, a laser beam is split into two perpendicular tubes. At the end of each tube is a mirror. After hitting the mirror, the light travels back through the tube, where it hits a light detector. We configure the length of the tubes, such that there is destructive interference when no gravitational wave travels through the detector. As a simplified explanation: When a gravitational wave passes, assuming the polarization considered is aligned on the arms of the detector, one of the tubes will stretch, while the other compresses. As a consequence, the proper path of the light is different and the resulting time needed to get to the detectors is different. This causes a phase shift and there is no longer destructive interference. Based on the phase difference, we can measure the difference in arm-length and therefore the amplitude of the gravitational wave. So there is a non-null signal at our light detector and it returns a signal. In theory, this would be enough to detect gravitational waves, however, we would need arms that are at least half the gravitational wavelength, in the order of 1000km. In practice, we therefore need some measures to improve the precision of the detector. Two techniques were implemented to enhance the accuracy. First,

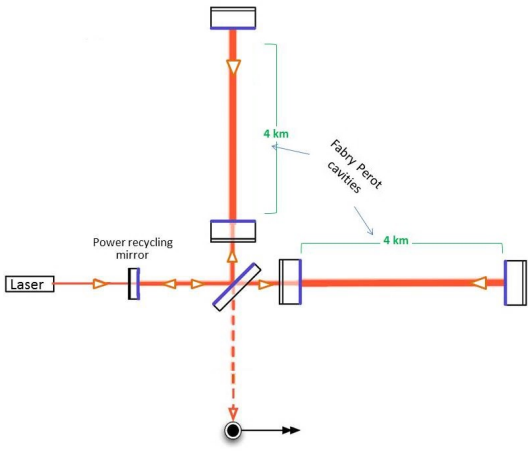

Figure 3: An interferometer with Fabry-Perot cavities. Additional mirrors are inserted near the beam splitter to facilitate multiple reflections of the laser, containing it within the interferometer and increasing the distance traveled by the beams. This greatly increases the sensitivity to the smallest changes in arm length, from [23].

additional mirrors were added to both arms, to construct *Fabry Perot cavities*. After entering the arm, the light is reflected 300 times. This amplifies the effect of a passing gravitational wave, as change in length in the proper path of the light is amplified.

Second, in order to measure gravitational waves, the detectors have to operate at 750kW. It is not viable to use a single laser of that wattage. Instead, a 40W laser is used and the output gets amplified by a power recycling mirror. Half of the light that travels back to the laser gets reflected back into the detector [23].

The output, called gravitational-wave strain, from the detector is [22]:

$$h(t) = \frac{\Delta t_x - \Delta t_y}{t_0} \tag{7}$$

Here, t_x and t_y are the time it takes light to travel in the x- and y-direction of the interferometer respectively. t_0 the time it would take when there is no gravitational wave.

Depending on the frequency, the detector has a strain sensitivity in the order of $10^{-21}/\sqrt{Hz}$ up until 10^{-23} [24]. As can be seen in Fig.4, the output

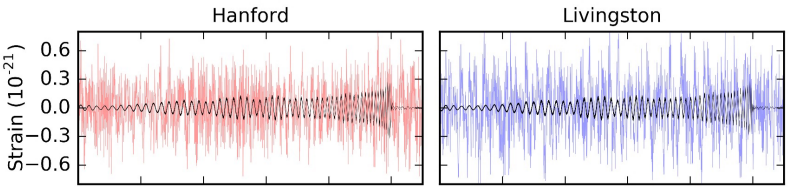

Figure 4: The detector output and the best matching waveform for the binary black hole detection, event GW151226 [5].

of a detector includes a lot of noise. Because the detectors are so sensitive, they pick up noise from seismic activity, thermal noise, and even noise from the fluctuation of photon pressure by the laser [23]. If we want to detect gravitational waves, we therefore need some way to filter out the noise. In section 2.4, we will see a way to filter the waves from the data.

2.3 Compact binary coalescence

Every object that accelerates and has a non-zero quadruple moment produces gravitational waves. With the current detectors, only black holes and neutron stars are dense enough to emit measurable gravitational waves. With developments in the detectors, other sources such as boson stars, white dwarfs and supernova explosions are expected to be observed, but they are beyond the scope of this work [25]. Of course, neither a neutron star nor a black hole will start accelerating on randomly. Extreme amounts of energy are needed. One process that could lead to acceleration of a dense object is the presence in its vicinity of another massive object. For example, another neutron star or black hole. When two neutron stars or black holes get close enough together, they will start orbiting around each other. The gravitational waves produced by the final stages of the orbit are large enough that we can measure them here on Earth from a distance of several Gpc [26].

The coalescence process starts when two celestial bodies meet each other and has three phases. The first phase is called inspiral, during which the two

compact objects orbit around each other. We are still in the weak regime of gravitational waves, meaning that we still have small curvature of space time and can use our approximations. When approximating to the first order, the emission of gravitational waves takes energy away from the system, following the equation

$$\frac{dE_{GW}}{dt} = \frac{32}{5}\frac{c^5}{G}(\frac{G\mathcal{M}_c\omega}{c^3})^{10/3}\,, [21] \tag{8}$$

where $\mathcal{M}_c = \frac{(m_1 m_2)^{3/5}}{(m_1+m_2)^{1/5}}$ is the chirp mass. It is used to make the analysis of binary systems easier as the orbital evolution depends on this combination of both masses. G is the gravitational constant. The orbital frequency ω is related to the gravitational wave frequency $f_{GW} = \frac{\omega}{\pi}$ [27].

For an isolated system, the energy that is radiated away is taken from the orbital energy:

$$\frac{dE_{GW}}{dt} = -\frac{dE_{orb}}{dt}\,. \tag{9}$$

The orbital energy is [21]

$$E_{orb} = \frac{1}{2}(G^2\mathcal{M}_c^5\omega^2)^{1/3} \tag{10}$$

Combining Equation 8, 9 and 10, we get

$$\dot{f_{GW}}(t) = \frac{96}{5}\pi^{8/3}\left(\frac{G\mathcal{M}_c}{c^3}\right)^{5/3} f_{GW}^{11/3}(t) \tag{11}$$

Solving Eq.(11), we end up with an equation for the gravitational wave frequency as a function of time as [21]

$$f_{GW}(t) = \frac{1}{\pi}\left(\frac{G\mathcal{M}_c}{c^3}\right)^{-5/8}(\frac{5}{256}\frac{1}{t_c - t})\,. \tag{12}$$

Here t_c is the coalescence time. In the first order approximation, t_c is the time where the frequency diverges. As the radius of the orbit decreases, the frequency will keep increasing. Without approximating however, the frequency does not converge at coalescence time, instead the radius of the inspiral will become small enough for the two objects to touch. This marks the start of the next phase, called merger. We will no longer be in the weak-field regime of gravity and can no longer make use of our approximations and now need to look at the full Einstein equations. Where during inspiral we

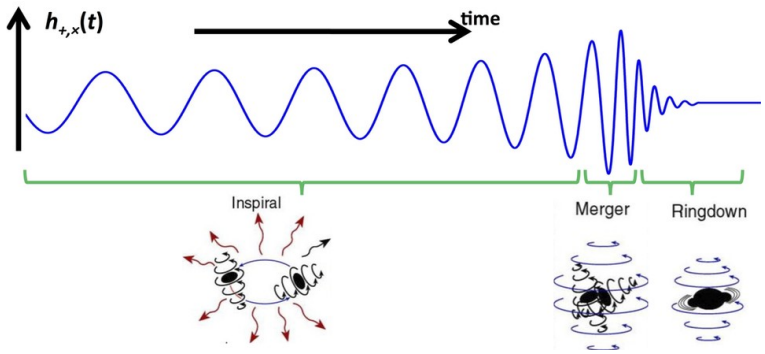

Figure 5: The waveform during the inspiral of a black hole with the corresponding phases.

could find the waveform analytically, we know have to model the waveforms numerically.

Depending on the masses, when two neutron stars merge, they will form either a larger neutron star or they will form a black hole. A black hole merging with either a black hole or a neutron star will always form a larger black hole [28]. The final phase is called ringdown. After merging, the created object will keep orbital momentum, while not being entirely symmetrical. The final object emits the excess of energy as GW in order to get to his energy ground state. During this time phase, binary neutron stars will emit so called afterglow [29]. Afterglow is a form of electromagnetic radiation, caused by a rapidly expanding cloud of material. From its spectrum, we can induce a lot of information about the system. For example, in binary neutron stars, it gives us an indication of the elements within the stars [29].

2.4 Matched filtering for gravitational wave detection

When we receive the signal from the detectors, it still contains a large amount of noise. Currently, the preferred method to assess the presence of a gravitational wave in data is called matched filtering. When a gravitational wave signal enters the detector, its output is a combination of noise and

gravitational-wave signal

$$s(t) = n(t) + h(t)\,, \tag{13}$$

where $s(t)$ is the measured strain, $h(t)$ is the gravitational wave signal and $n(t)$ is the noise. If we know what shape the signal has, we can compare this signal against the data to determine if there is a signal in the detector. We therefore make use of a template bank. It is a collection of waveform models, which have been precomputed. These waveforms are then whitened, which normalises the power at all frequencies. We can use these whitened templates and filter them against the data. When one of the templates corresponds with the signal found in the data, the signal is cross matched with the other detectors. So, one requires to have the same template matched in different interferometers, with the appropriate time delay between them. One detector signalling a gravitational wave is not enough to notify astronomers, as the detectors are susceptible to glitches [30]. They occasionally recognise a waveform that isn't present in the data. If multiple detectors found the same signal, they can be sure a gravitational wave passed.

The output of matched filtering is the *signal-to-noise ratio*. [1] The value of the signal-to-noise ratio (noted SNR and denoted ρ) is the loudness of a signal in the detector.

$$\rho = \left[4\Re \int_{f_{min}}^{f_{max}} \frac{\tilde{s}(f)\tilde{h}^{\star}(f)}{P(f)} df \right]^{1/2} \tag{14}$$

where $\tilde{h}^{\star}(f)$ is the complex conjugate of the Fourier transform of the template. $P(f)$ is the noise power spectral density, which is a measure of the detector noise for each frequency. f_{min} is the lowest frequency the detector is sensitive to and f_{max} is the highest frequency reached by the binary of the innermost stable orbit [1].

The optimal SNR is when the template waveform corresponds to the gravitational-wave signal present in the data [1, 31]

$$\rho = \left[4\Re \int_{f_{min}}^{f_{max}} \frac{|\tilde{h}^{\star}(f)^2|}{P(f)} df \right]^{1/2} \tag{15}$$

This represents the loudness of the signal in the detection for a given noise and is therefore related to the difficulty of detection.

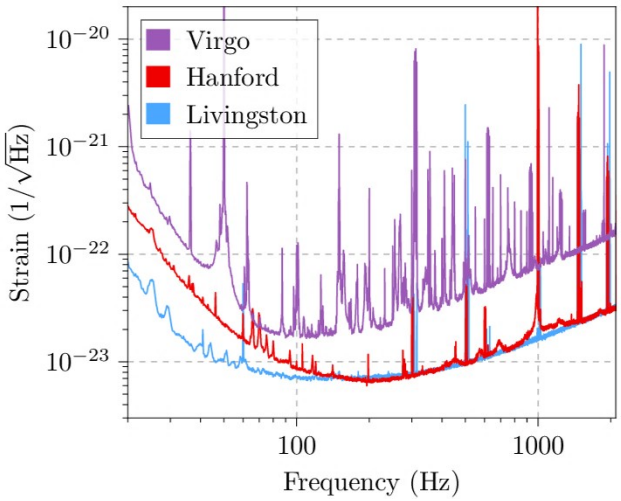

Figure 6: The amplitude spectral density of the different large detectors. The amplitude spectral density is the square root of the power spectral density. This is a measure for the noise for each frequency. Adapted from [3].

Approximating to the first lower order the SNR becomes

$$\rho \simeq \frac{1}{2}\sqrt{\frac{5}{6}}\frac{1}{\pi^{2/3}}\frac{c}{D}\left(\frac{G\mathcal{M}_c}{c^3}^{5/6}\right)\sqrt{I}g(\theta,\phi,\psi,\iota) \tag{16}$$

This gives us a better understanding of what parameters influence the SNR. First D, the luminosity distance. Systems that are further away are harder to detect as they are fainter. Second I, the frequency integral.

$$I = \int_{f_{min}}^{f_{max}} \frac{f'^{-7/3}}{P(f')} df' \tag{17}$$

Finally g, the sky position. Objects at different sky positions might be detected less accurately due to the shape and orientation of the detector. In this work, we look only at the inspiral part of the signal. We can approximate the maximum frequency using Eq(12), using this and combining it with the lowest sensitivity for the detector, we can calculate the SNR at the frequency covered by the inspiral. This is known as the *partial inspiral SNR*, noted PI SNR. The higher the PI SNR, the easier a signal is to detect.

2.5 Multi-messenger astronomy

If we can detect the gravitational waves generated by a binary neutron star in at least two different detectors, we can triangulate the position of the binary neutron star in the sky. We can then observe the behaviour of the binary neutron stars across the electromagnetic spectrum and neutrino domain. Currently, the only observation with a detected counterpart is known as GW170817 [3]. In August of 2017, a binary neutron star coalescence candidate was observed by the LIGO and Virgo detectors. Within an hour, multiple teams had detected an electromagnetic counterpart. They could combine the electromagnetic detection with a gamma-ray burst detection moments earlier [32], confirming a longstanding theory that binary neutron stars create gamma ray bursts.

Then, through optical and infrared detections, they detected a redwards shift over 10 days. The optical and near-infrared spectrum showed similar characteristics to models of a kilonova [33]. A kilonova is a phenomena where two BNS or a NSBH merge and result in an object with 1000 times the brightness of a "normal star". Besides gamma ray bursts, kilonovas were also thought to produce r-process nuclei. The r-process is a set of reactions where neurons are captured. The resulting nuclei were also detected in later stages of the optical detection, confirming that the binary neutron star had become a kilonova [34]. Finally, radio and x-ray emission was detected after 9 and 16 days respectively [4]. From these results, the value of the gravitational wave detections becomes clear. It allows us to look deeper into the mechanics of the detected objects.

The objective of this work is to improve the systems enabling an earlier detection of the binary neutron star. We build upon the work presented in [1], where neural networks are used to produce triggers for merging neutron star up to $100\,s$ before the merger. We want to use two techniques, called *Inception modules* and *dilation* to improve on the networks developed in [1]. In the next section, we introduce the concepts used to build the neural network and detail the new techniques with their implementation.

3 Methodology

As mentioned in the last Chapter, we are currently working towards using neural networks to detect gravitational waves. In this Chapter, we explain

how neural networks work and can be used to detect gravitational waves.

3.1 Machine learning

Machine learning is a field within computer science, where the goal is to design an algorithm that can learn from examples. The basic idea behind current technology is to imitate the human brain. Simplified, the human brain is a group of neurons, all connected by connections called synapses. When a neuron receives an electric signal from another neuron, it modifies the signal and sends it to a different group of neurons. Depending on how strong the signals are, different effects in the body will occur [35]. This is of course an extreme oversimplification, but it will turn out to be enough for our problem.

Neural networks can be easily explained by first looking at linear regression. In linear regression, we have a data set consisting of points. From this data set, we want to obtain a model, such that when we know a variable x, we can predict some other variable y. We assume that there is a linear relation between such that $y = ax + b$ and we want to find the best values for a and b. To get the best values, we first determine a function to represent the error for every data point. In Fig.7, this is the minimum square function in the blue box. We then minimise this error function and receive values for a and b fitted to the data.

Neural networks work similar to linear regression, but instead of fitting a simple line function, a complicated function is fitted. This complicated function is build from the neuron functions:

$$f(x) = \phi(b + \sum_{j=0}^{m} w_j x_j) \tag{18}$$

In this equation, x is the input, $f(x)$ is the output, b is the bias and ϕ represents the activation function, which modifies the output of the neuron. Finally, in Eq.(18), the w_j are the weights that need to be optimized. Common activation functions are the *ReLU* (Rectified Linear Unit) and *Sigmoid* functions, defined as

$$ReLU(x) = max(0, x) \tag{19}$$

$$Sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{20}$$

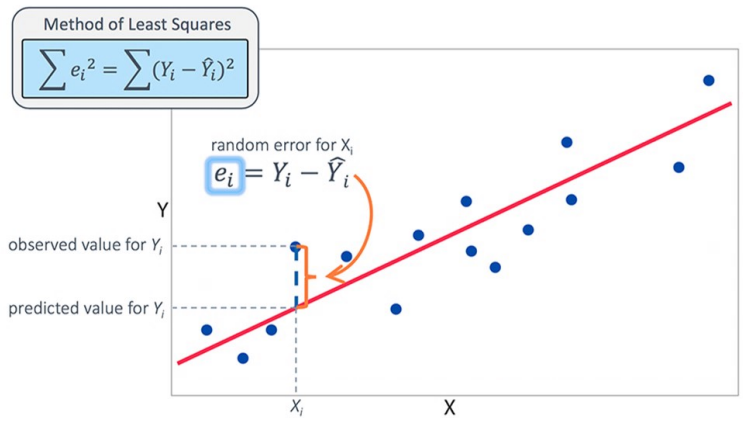

Figure 7: An example of linear regression, where the data points (blue) are regressed to find the best possible the fit (red line).

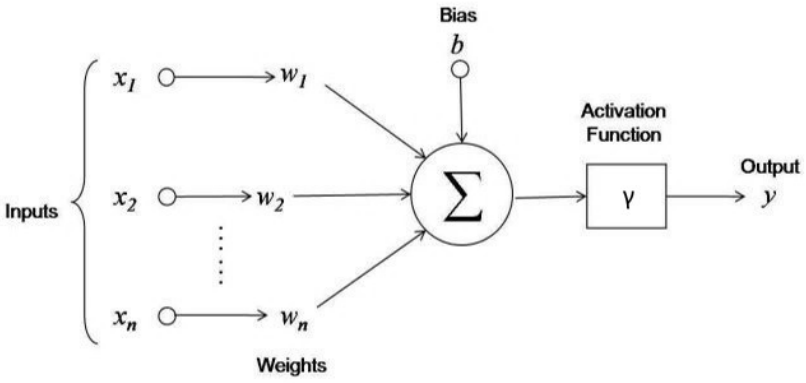

Figure 8: Representation of a neuron in a neural network.

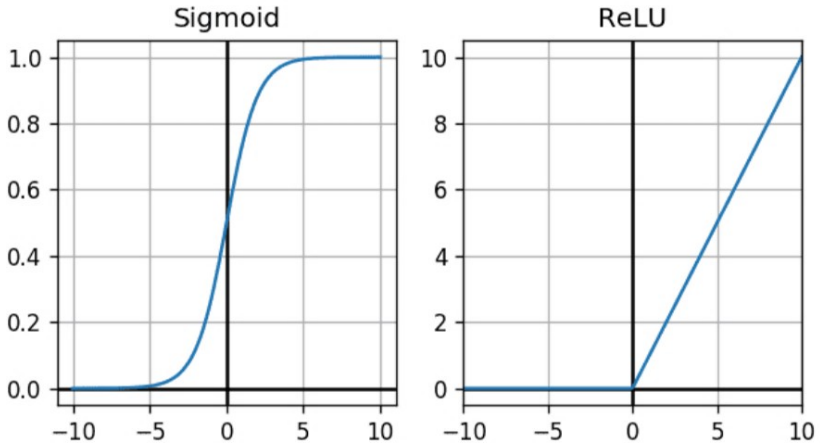

Figure 9: The *ReLU* and *Sigmoid* activation function.

ReLU modifies the output such that it is always positive. Sigmoid binds the output between 0 and 1. From this it is easy to see why neural networks are not linear functions. A representation of both functions can be found in Fig. 9.

In current neural networks, neurons are laid out in layers. For an example, see Fig.10, where every neuron receives input from several neurons in a previous layer and sends its output to neurons in another layer. Instead of the linear function seen in linear regression, the activation function and bias in the neurons allow us to obtain complicated functions that better mimic the data. We again want to optimise the weights in each of the layers such that the output of the network accurately predicts the input. Just like in linear regression, this is done by first defining a function representing the error. This function is called a *loss function* and is minimised so that the error is reduced.

3.1.1 Loss function

As explained above, the loss function is a measure of the error in the output of a network. An easy example of such a function is the *mean squared error*.

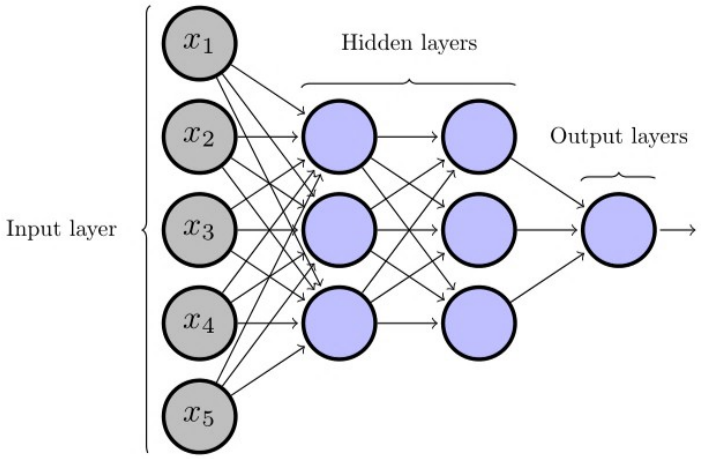

Figure 10: Example of layers of neurons in a neural network architecture [36].

When calculating this function, we give the network a large amount of samples and for each sample, we compute the square of the difference between the output of the network and the value attributed to the data.

$$MSE = \frac{1}{N}\sum_{i}(Y_i - \hat{Y}_i)^2 \tag{21}$$

With Y_i the predicted value and $\hat{Y}_i$ the actual value.

Which function accurately displays the error in the network is dependent on the task the network has to fulfill. Amongst other tasks, there are networks for regression (i.e. predicting a numerical value), clustering (i.e. grouping several items), classification (i.e. determining which class an item belongs to). In this work, neural networks are used as *classifiers*.

A classifier takes an input and determines to which class it belongs. In this work, the network receives a data time series and determines whether it contains a gravitational wave or if it contains only noise. This type of classifier should return 1 when the input contains a gravitational wave and 0 if it contains only noise. When we want to determine the performance of our network, we give it labelled data for which we know if it has a gravitational wave or not. If it has a gravitational wave, the network should output a value as close to 1 as possible.

In this paper we use a more elaborate loss function called the *cross entropy*

loss function, where we calculate the loss as

$$\text{Loss}(x_i) = -x_i + log(\sum_j e^{x_j}), \tag{22}$$

Where x_i is the actual probability for a given class and x_j are the probabilities for the classes as returned by the network. Cross entropy is a way to measure the distances between probability distributions. In neural networks, the distance between the probability returned by the network and correct class is measured. This is an indication of the error on the predicted values. The smaller the error in the prediction, the closer the probability returned by the network will be to the right class and the smaller the loss will be [37].

3.1.2 Gradient descent

As previously mentioned, we want the lowest possible error rate, which is translated by a minimization of the loss function. For a single signal, it is possible to calculate the correct weights analytically. However, when we want to correctly classify many different signals, it is not possible to compute the correct weights either numerically or analytically. Therefore, we iteratively modify the weights during a process called *training*.

First, we start off with random weights. Every weight is set to a value between 0 and 1. Then we apply a technique called *gradient descent*. It works as follows: we let the network classify a number of different samples. Then, we calculate the loss over these samples. Recalling that the loss function is a function on weights, input and a bias, we can visualise the loss as a multidimensional hyperplane. This way, one sees that if we want to find a minimum in loss, we need to descend along the gradient of this plane. This is represented in Fig.11. In order to minimise the loss, we therefore calculate its gradient. Based on the latter, we adapt the weights.

In this work, stochastic gradient descent is used. For each iteration, a small batch of samples is fed through the network and the weights are adapted as:

$$w_{i+1} = w_i - \frac{\alpha}{n} \sum_{j=1}^{n} \nabla L_j(w_i). \tag{23}$$

In this equation, ∇L_i is the gradient of the loss calculated on n samples. A group of n samples is called a batch.

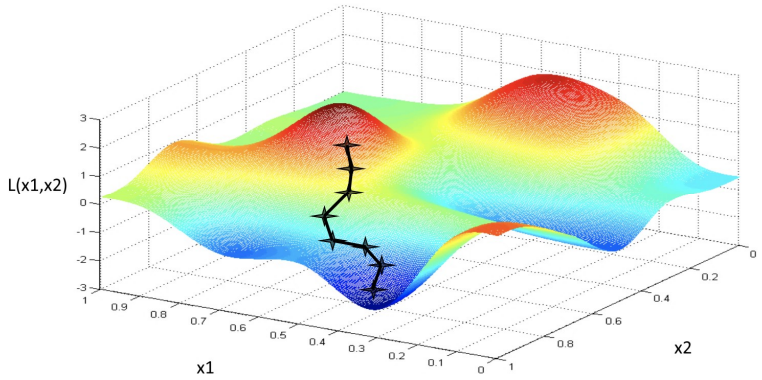

Figure 11: A visualisation of gradient descent in a loss function on 2 variables, adapted from [38].

After obtaining the average gradient of this loss, it is then multiplied by a factor α, known as the learning rate [39]. The learning rate is a measure for the amount by which the weights are changed when trying to descend towards the minimum in the loss-plane. The learning rate determines the size of the step we take towards the minimum. If the learning rate is too small, we take very small steps towards the minimum. This means that it takes more time to reach the minimum and the training time becomes larger. It can also mean that we get stuck in a local minimum or saddle point, as the steps we take are not large enough to get out of it.

When using a learning rate that is too large, we risk having a network that does not converge towards the minimum at all, meaning that the network does not increase performance when training. In this work we use a learning rate of $8e-5$. Using a larger learning rate results in worse performance of the network, while using a smaller one leads to a longer training time without improving the performances of the network.

3.1.3 Optimizers

When training a neural network, we are adapting the weights until we get to a minimum. A loss function is often a complicated function with many local minima and maxima. If we pick a random minimum, there is a large chance that we pick a local minimum, instead of a global minimum. To increase our chances of getting a minimum close to the global minimum, we use so called *optimizers*. They control the descent towards a minima, by modifying Eq.(23). The idea is to use a larger learning rate at first, which slowly decays as we get further in the training. The optimizer used in this work is called Adamax. It uses a property called momentum. This can be interpreted like the physical property momentum. When a ball rolls down a hill, it has a certain momentum. When the ball reaches a tiny hole in the hill (a bad local minima), the momentum forces it to roll out of the hole and continue down the slope towards a better minimum. Adamax splits the momentum in two different variables v and m. v is defined as the largest of the current gradient and the gradient of the last iteration, multiplied by a dampening constant,β_2. The dampening constant can be seen as the friction on the ball from the example. In [40], the authors suggest a value of $\beta_2 = 0.999$. Every iteration, the contribution of past gradients is multiplied by 0.999, in order to not let the highest gradient dominate the momentum. Adamax controls the learning rate by dividing it by the value v_i, defined as

$$v_i = max(\beta_2 v_{i-1}, \nabla L). \tag{24}$$

∇L is the current gradient of the loss. m is the moving average of gradients. Adamax controls the gradient by incorporating past gradients into the current one with a constant factor β_1, with a proposed value of 0.9 [40].

$$m_i = \beta_1 m_{i-1} + (1 - \beta_1)\nabla L \tag{25}$$

Adamax finally adapts the Eq.(23) such that

$$w_{i+1} = w_i - \frac{\alpha}{v_i}\frac{m_i}{1 - \beta_1^i} \tag{26}$$

3.1.4 Convolutional neural networks

The most commonly used neural networks, which are the ones used in this work, are called convolutional neural networks (CNN's). In the simplest

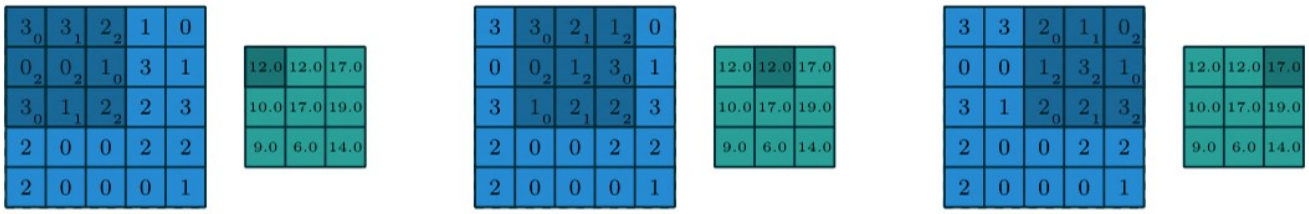

Figure 12: Example of a convolution. This convolution uses a 4x4 filter. While shifting the filter over the input, four values are combined with the weights to produce one output value [43].

terms, a CNN is a series of convolutions. The goal is to feed input data into the neural network and to get as output the probability that the input contains a gravitational wave. To do this, we first need to know what a convolution is. A convolution is a way to combine two functions f and g, such that the shape of g is modified by f. The convolutional operator $*$ is defined as [41]

$$f_{i+1} = \sum_{0}^{C-1} \sum_{0}^{M} g_{m,c} \times f_{m+i,c} \tag{27}$$

In neural networks, the function f is the input of the network and g is a series of weights. Every convolutional layer consists of a set of neurons. When we apply the convolutional operator to the input and the weights, we are actually shifting the weights over the input. By doing so, we are calculating a form of weighted average [42]. More intuitively, we take a matrix of weights, called the filter. We then shift this filter past the input, multiplying every input value with this matrix. See Fig. 12 for an illustration. Finally, C stands for the number of channels. This represents the number of filters we shift over the input. For an image for example, one might have separate filters for the red, the green and the blue values, but we can have as many filters as we want. More filters means more weights to calibrate.

One convolution returns an array of outputs, which then get used as input for the next layer.

To discuss neural networks on a more abstract level, some more definitions are needed. First we introduce the kernel size. The kernel size is a term for the size of the filter. In Fig.12 the kernel size is 3×3. This means that every neuron is connected to a square of 3×3 neurons in the previous layer. The

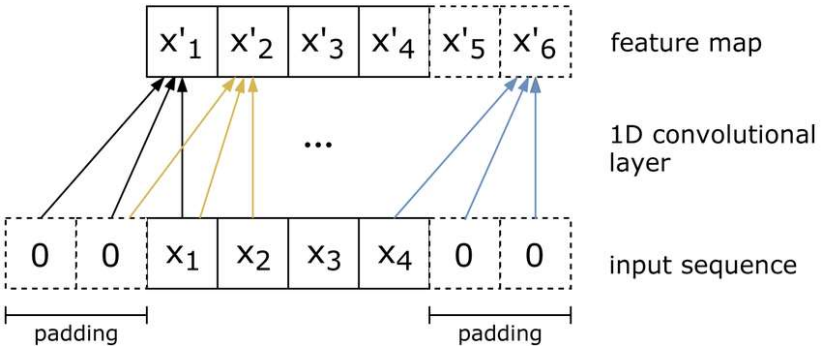

Figure 13: An example of padding in a 1D convolutional layer. Zeroes are added to the input, resulting in a larger output [45].

amount of nodes a neuron is connected to is called the *receptive field*. When training neural networks, it has been shown that contextual information is very important. For example, when a network tries to recognise a picture, it is important that a node has access to not just one pixel, but also the pixels around it [44]. A larger receptive field is therefore a good way to improve network performance.

However, a larger kernel size also means that the output size for a single layer becomes smaller. If we do not want the input size to become smaller, we can implement padding (see Fig.13). Using padding amounts to adding additional values to the input size. These values are only there to moderate the size of the output and do not contain any information about the input. Besides increasing the input size, padding can also be implemented to conserve information on the edge of the input. Without padding, this information is only used to produce 1 output, while other values are used equal to the kernel size. In this work, padding is always done by symmetrically adding zeroes, meaning that we append the same amount of zeros to the end of the input as at the start of the input as done in Fig.13. By doing this, the output size as a function of the input size becomes [46]

$$|x_{i+1}| = |x_i| + \frac{2 \cdot \text{padding} - (\text{dilation} \cdot \text{kernel size} - 1)}{\text{stride}} + 1, \qquad (28)$$

where the *stride* represents the amount by which we shift the filter every iteration. The stride is usually 1 in convolutional layers. Dilation is a way

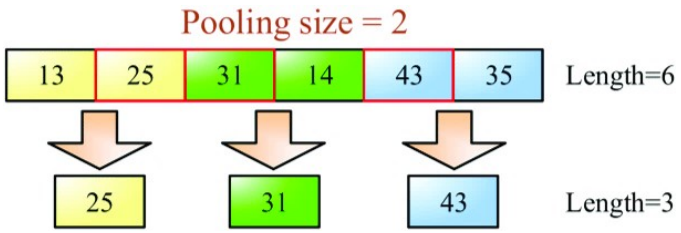

Figure 14: A representation of a pooling layer with a kernel size of 2 and a stride of 2.

to increase the receptive field without picking a larger kernel size and will be explained in Section 3.3.

3.2 Pooling layers

Sometimes however, to reduce the computational cost, we want to artificially decrease the size of the input. Randomly removing values would lead to a large loss of information. Instead pool layers are implement. These layers decrease the input size in such a way that the decreased input best represents the actual input. Much like a normal filter, a pool layer takes adjacent values and applies a function to it to get the best representation. The most common pooling layer is max pooling (see Fig.14), where we take the maximum value as the representative value.

$$output(n) = \max_{i=0}^{m}(x_{sn+i}), \tag{29}$$

where m is the filter size and s is the stride. In pooling layers, the stride is rarely less than 2, as a stride of 1 does not significantly decrease the input size.

3.3 Dilation

In section (3.1.4), the concept of a receptive field has already been discussed. Increasing the receptive field can be achieved in many different ways. Originally there were two main methods. The first was to simply increase the kernel size. By doing this, a single neuron is connected with more nodes from the previous layer, leading to more contextual data. However, when using larger networks, this very quickly becomes computationally expensive, as the introduction of a new node in a layer with N nodes leads to $N!$ new connections.

As seen in the previous section, we can use pooling layers to decrease the input size in order to make it computationally cheaper. Although pooling layers are inexpensive, they have one clear downside: with every pool layer, a certain amount of information is lost, making the network perform worse.

To solve the aforementioned issues, the concept of dilation is introduced. When we introduce dilation to a convolutional layer, instead of feeding adjacent data points to a filter, we use every N^{th} data point. This is shown in Fig. 15. With dilation, the convolution equation becomes

$$f_{i+1} = \sum_{0}^{C} \sum_{0}^{M} g_{m,c} \times f_{lm+i,c}, \tag{30}$$

where l is now the dilation. As seen in Fig.15, after dilation, the group of nodes in the first layer are connected to a much larger group of nodes in the second layer. In recent papers, this effect was amplified even further by doubling the dilation in every layer [16].

In this paper we apply a similar dilation to the one implemented in *Wavenet* [16]. There, they start of with a dilation of 1 and then keep doubling the dilation until a dilation of 512 is reached. Then they start at 1 again. In this work, the implementation was similar, but due to a smaller input size, we could only increase the dilation up to a value of 64. When increasing the dilation past this value, this value would have lead to negative input values for some of the layers, which is not possible.

This can be fixed by using padding. Using the simple formula as defined in Eq.(28), we can calculate the padding needed to keep the size of the input consistent. The problem with this, is that the padding increases with the dilation and we want to avoid large padding as the increase in padding increases the loss in information [47]. Indeed, now, nodes on the edge not

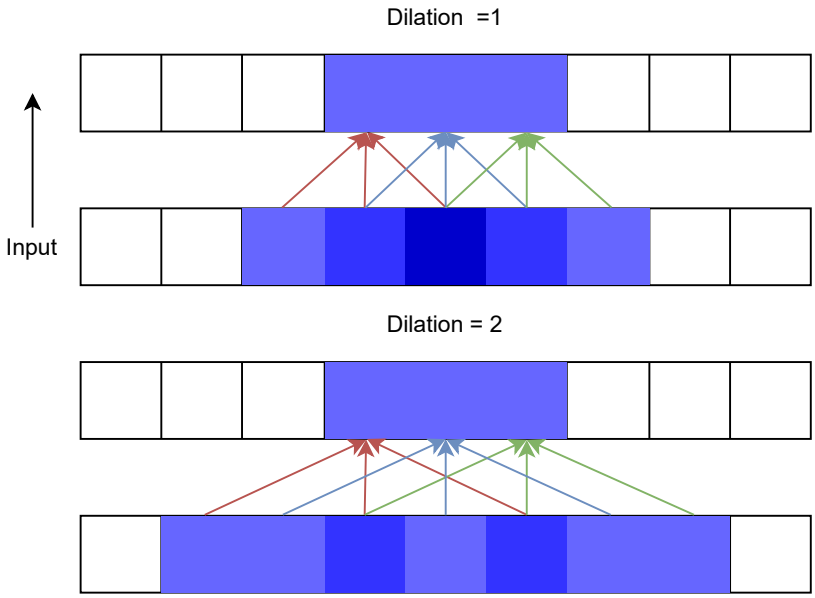

Figure 15: A representation of nodes in a layer, connected to the previous layer. Top: When using a dilation of 1. Bottom: When using a dilation of 2.

only include the input data, but also a large amount of padded zeros that are not representative of the data, degrading the accuracy. We therefore chose not to incorporate any padding in the convolutional layers.

3.4 Inception modules

The second technology in this paper is called inception modules. In this work, we adapt the inception modules from [18]. In general, a way to increase performance in neural networks is to increase the depth of a network. However, adding extra layers comes at great computational cost. It also leads to overfitting of the network. It means that the network's performance on the training data increases while the performance on test data worsens. Instead of learning the patterns in the data, the network begins to recognise the individual samples, leading to a loss in generality for the network. The solution they came up with was to mimic biological structures by moving to more sparse networks, which means that neurons in networks are connected to less neurons. An example of this can be seen in Fig.16. It needs to be noted that we do not just reduce the kernel size, which would also reduce

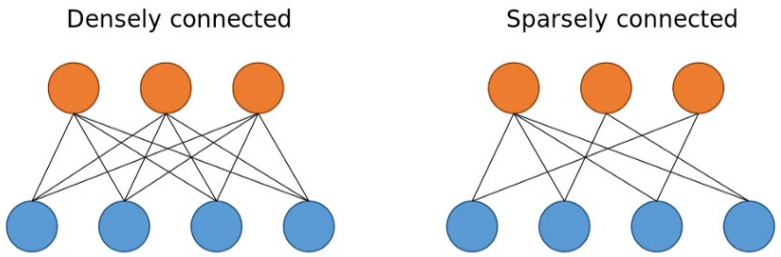

Figure 16: An example of two sparsely connected layers. The connections here are randomly chosen and not representative for those in an inception module.

the amount of connections between layers. Instead the kernel size remains the same, but some values from the kernel are disregarded. By doing this, the receptive field does not decrease in size, while the amount of connections between the layers is reduced.

By reducing the amount of connections between layers, networks can be very deep without overfitting. This better mimics the human brain and would lead to a better performance. For a deeper and more mathematical explanation, one can look at [48], but this goes beyond the scope of this work. Unfortunately, the current hardware available to us is very unoptimized for sparse data structures. Where dense calculations can be fully parallelised on the GPU, sparse calculations use many lookups, making it 100 times slower [48]. This means that switching to sparse networks will not lead to any major performance increases on the current hardware [48]. We can however approximate this sparse structure. By using dense convolutional layers in parallel, the network is still optimized for the architecture of the hardware while achieving the same result as in a sparse layer.

In this paper, we will use *Google Resnet* as a guideline for building blocks.

It is not possible to simply use the *Resnet V4* network for gravitational-wave detection data as the input images for this network are two dimensional while the data we receive from gravitational-wave detectors is one dimensional. This means that the inception modules needed to be reworked to accept one dimensional input. In this work, the *Resnet A* and *Resnet C* block have been adapted to 1D data.

The A- and C-blocks in 2D are shown in Fig.17, and Fig. 18. The A-block consists of a ReLU activation layer, followed by three parallel branches with convolutional layers. The first branch has a single convolutional layer with a kernel size of 1. The other branches start with the same convolutional layer and are followed by 1 or 2 other convolutional layers. The first layer, with a 1×1 kernel size, decreases the amount of parameters and speeds up the training. The output from the three branches is then concatenated and fed through a final 1×1 convolutional layer. It also incorporates a residual connection, which is the connection between the input and the output of the block and does not have any convolutional layers. First introduced in [49], residual connections or skip connections, were meant to improve image recognition in very deep networks, by allowing past gradients to flow through the network directly. They were also used to speed up training. In the Resnet implementation, it was not proven that they improved the results, but they did substantially decrease training time [18].

The C block is very similar to the A block, but there are minor differences. First, it has only two instead of three branches. The first branch is similar to the first branch of the A block. The second branch is similar to the third branch of the A block, but the kernel sizes are different. Instead of having kernel sizes of 3×3, it has one 1×3 convolution, followed by a 3×1 convolution. This is equal to having one 3x3 matrix, but they found it to be 33% cheaper.

3.5 Architecture of the network

The final architecture used in this work can be seen in Fig.21. It is a standard convolutional network, interlaced with two 1D-Resnet A blocks and 1D-Resnet C blocks. After each convolutional or 1D-resnet block, a max pooling and activation layer is used.

The Adaptation of the Resnet A and Resnet C blocks can be seen in Fig.19. The major changes are the following. The layers within the branches are very different. First, we use a different kernel size. In one dimension, the

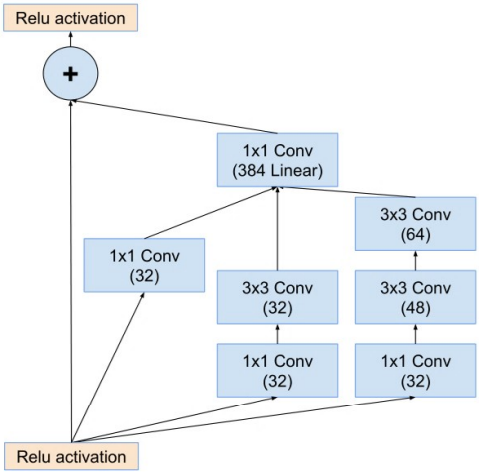

Figure 17: Representation of the *Inception-Resnet-A module* from the *Inception-Resnet-v2* network. It consists of three branches with varying kernel sizes and a skip connection [18].

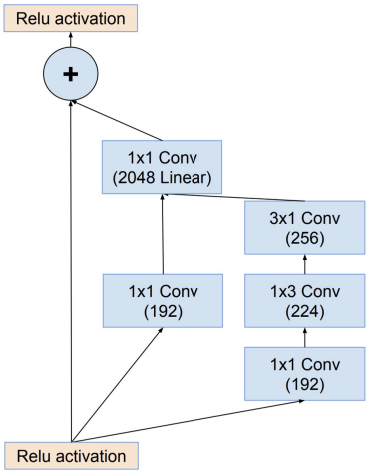

Figure 18: The schema for the *Inception-Resnet-C* module from the *Inception-Resnet-v2* network. It is similar to the A-block, but does not include the branch with the largest kernel size [18].

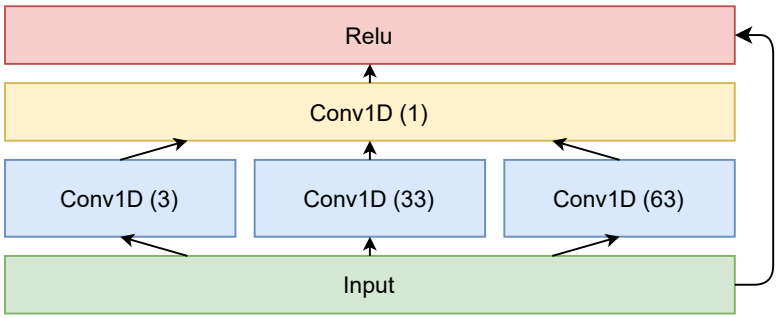

Figure 19: The architecture for the 1D-Resnet block A.

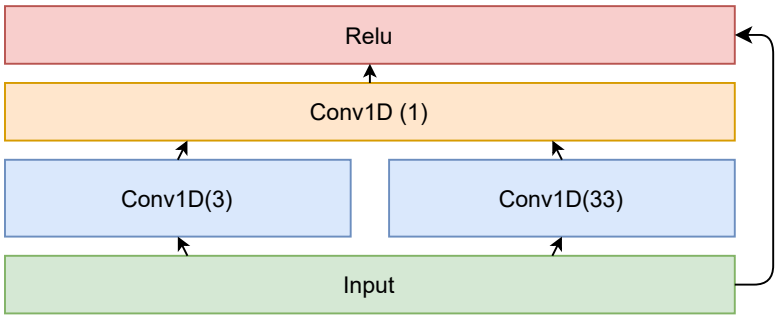

Figure 20: The architecture for the 1D-Resnet block C.

original kernel sizes did not result in good performance of the network. When using the base convolutional network, mixed with the Resnet blocks, the best performance was reached by using kernel sizes of 3, 33 and 63 for block A and 3, 33 for block C. Secondly, note that there are no convolutional layers with a kernel size of 1 before any of the convolutional layers in the branches. These layers were removed because they did not improve learning. In 2 dimensions, the convolutional layers with a kernel size of 1 are included to reduce the amount of parameters, which increases training speed and they also serve as a form of linear rectification. In 1 dimension however, parameters increase at a much lower rate when adding new layers (linear order instead of quadratic). We suspect this could be the reason for the difference in outcome. Both blocks do still implement a skip connection, similar to the original Resnet blocks.

The kernel size as shown in Fig.21 is the kernel size as for the network without dilation. Before implementing dilation, the network was first optimised. When moving on to dilation, we noticed that different kernel sizes worked for different dilation layouts[1]. In the blocks, we also always implement padding when using dilation. This is necessary, because in order to concatenate the output of the different branches, all branches need to have the same output size. Since these branches have different kernel sizes, we can only guarantee consistent output sizes by using padding. To make sure that the influence of the padding in these blocks remains small enough not to degrade the training of the network, we decided to use smaller kernel sizes in blocks with larger dilation. In some networks, it was possible to avoid having large dilation in the 1D-Resnet blocks all together. As can be seen in Section 4, not all layers have dilation. When we had a large dilation in one of the 1D-Resnet blocks, while not all convolutional layers had dilation, we often chose to have a dilation of 1 in the 1D-Resnet block, moving the dilation to the next convolutional layer.

3.6 Training of the network

The network in this work was trained on a data set consisting of data simulating heavy binary neutron stars. In this work, when we talk about heavy BNS, we use the definition from [1]. By this definition, a binary neutron star pair is considered heavy when the chirp mass exceeded $2.09 - 2.61 M_{\odot}$. We

[1] When talking about dilation in the *1D-Resnet* blocks, we only increase dilation in the branches of the blocks. The final convolutional layer does not have any dilation.

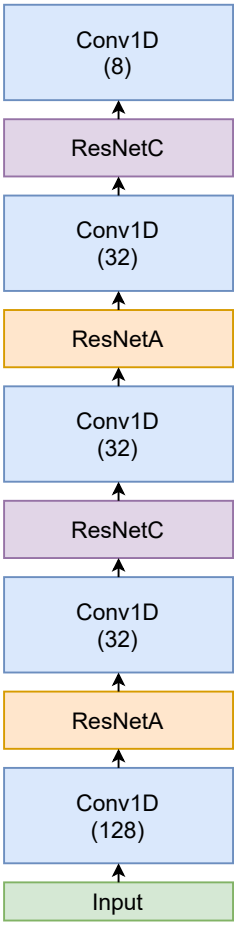

Figure 21: The architecture for the network with inception modules but without dilation as used in this work.

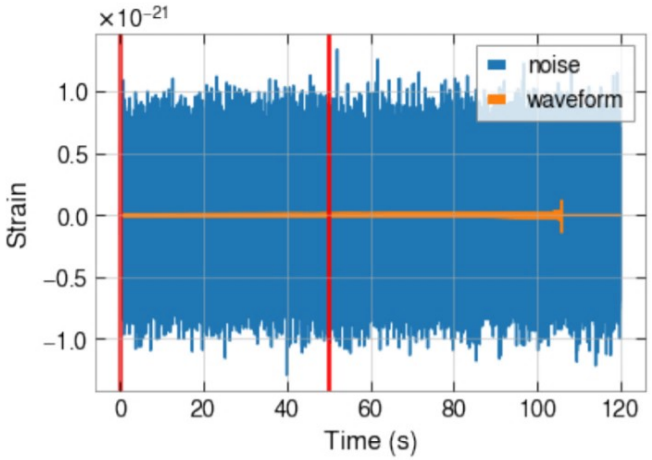

Figure 22: An example of a sample on which the neural networks are trained. This sample contains a gravitational wave. Only the data between the two red line is used to feed the network.

focus on this category alone as the results here can be generalised to other categories as well as shown in [1].

The data we use to train the network consists of noise combined with a template. Every training sample contains 30 seconds of colored Gaussian noise based on the sensitivity power spectral density in the detector. To this noise, we add a simulated non-spinning heavy BNS waveform. We choose the location of the BNS to be the optimal sky location, where the plus polarisation is aligned with the arms of the inferometer. An example of such a waveform in simulated noise can be seen in Fig.22. The waveforms are chosen such that the distances range from 50 to 240 Mpc and the PI SNR range from a value of 7 to a PI SNR of 35. When the waveform is injected into the noise, we select only the first 30 seconds to have only the early inspiral. The dataframes are then whitened and normalised, such that the amplitudes have a value between -1 and 1.

In total, the training set consisted of 8000 data frames, where 4000 dataframes contain noise combined with a gravitational wave signal and the other 4000 consist of just noise.

3.7 Obtaining results

After training the network, we use a test set to determine the accuracy of the network. The test set included 4000 of Heavy BNS samples, similar to the ones in the training set. It also contained 4000 samples filled with noise. When we run the network over a sample, the network will return a value between 0 and 1 which is related to the probability for the presence of a gravitational wave in the data. A higher value means a higher chance for a gravitational wave. Of course, in practice, we do not want a probability, but instead a conclusion about the presence of a wave. The easiest way to accomplish this is by choosing a threshold. When the network returns a value higher than the threshold, we decide that the network classified the sample as a gravitational wave. If the value is lower, it determined that the sample did not contain a wave. Here, the threshold is chosen so that we get a fixed false-alarm rate (FAR) of 0.01 [50].

$$FAR = \frac{FP}{FP + TN}\,, \tag{31}$$

where FP is the number of false positives, meaning the network determined there was a gravitational wave in the input while it only contained noise. TN stands for the number of true negatives, indicating that the network determined correctly that there was only noise in the input. The threshold is determined such that we get a false alarm rate of 1%. Using this threshold, the true positives, false positives, true negatives and false negatives are determined. They can be displayed in a confusion matrix [50]: A confusion

	Event	No event
Event	TP	FN
No event	FP	TN

Table 1: A confusion matrix used to compare the networks. It contains the true positives, false positives, true negatives and false negatives.

matrix is a representation of the results of a network. It displays the number of examples correctly classified as containing a gravitational wave (true positives), the number of examples falsely classified as containing a gravitational wave (false positives), the number of examples correctly classified as containing only noise (true negatives) and finally the number of examples

wrongly classified as containing only noise (false negatives). It is used to easily compare the different networks to each other.

Besides the confusion matrix, we will use the *true alarm probability* as a leading metric to determine the performance of a network. The true alarm probability is defined as [50]

$$TAP = \frac{TP}{TP + FN}\,. \tag{32}$$

The true alarm probability is the probability that a signal containing a gravitational wave is well classified. This is important when using the network for actual detection as detections are broadcasted to a large number of astronomers in the world and a false alarm could result in a lot of wasted research effort. In current matched filtering pipelines, the true alarm probability is related to the SNR. It is 99% [51] for an SNR higher than 10.

4 Results

In this section we will analyse the results. The main criteria by which we analyse these are the TAP as a function of the distance, and the TAP as a function of the PI SNR. Recall from the background information that the PI SNR is inversely related to the distance. Before analyzing our result, we first summarize the results from [1], because we build upon these. Then we will look at the results from this work and compare both.

In [1], they use the network as shown in Fig.23. This network produces 100% TAP up to a distance of $\sim 125\,Mpc$. They obtained a TAP of 88% over the entire set and a TAP of 1 up until a PI SNR of 17. They compared their networks to the current matched filtering techniques, looking at gravitational wave samples with a PI SNR of at least 8. They found that their results are similar to matched filtering, but the detection by the neural network is a factor 10 faster. When using the same network, but trained on a training set consisting of simulated BNS with an intermediate mass $(1.56 - 2.09 M_{\odot})$, they get a TAP of 68%, showing that the techniques applied in their paper work on BNS with a lower mass as well.

In this work, various combinations of dilation structures and kernel sizes, both in the blocks and inception modules have been tested. In this section, we only show the best performing architectures. The details about the other structures tested can be seen in Appendix A. An overview of the architectures

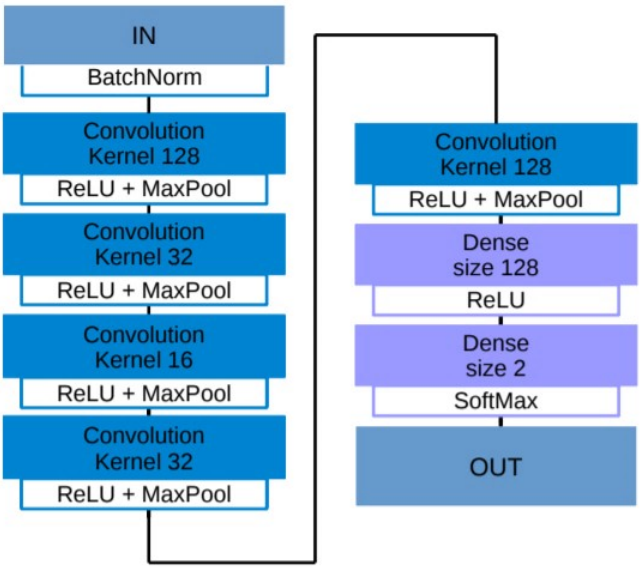

Figure 23: The architecture of the best performing network as described in [1].

that are discussed in this section can be seen in Table 2. For every layer, the dilation size is given and the layer are displayed in their order of appearance in the network (from left to right).

First we have the network with blocks but without any dilation. As seen in Fig. 24, the True alarm probability for the network without dilation is 100% up until a distance of 80 Mpc. This is significantly worse than the network as seen in the network of [1], where the TAP is close to one for distances up to 125 Mpc. This is the best result we have obtained when using Resnet blocks and no dilation. Better results may be obtianed using different architectures, but were not explored in this work. Before getting this result, the kernel size and learning rate for this network have been optimised. Other kernel sizes and learning rates can be seen in Appendix A. Unlike the original Resnet, this network is not very deep. The reason for this is to stay closer to the network from [1] in order to get a better comparison. We suspect that in the future, better results may be reached by implementing a deeper network in combination with the inception modules, although larger training sets may be used to prevent overfitting on the data.

Using the network described above as a base, we implemented other tech-

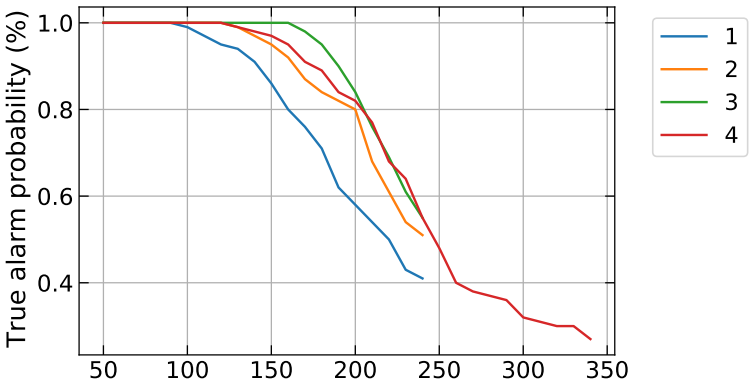

Figure 24: The TAP as a function of the source distance for the different networks considered in this work. The network using blocks and no dilation (1) performs the worst. When adding dilation only in the convolutions (2) the network TAP as a function of distance becomes closer to the one from the network described in [1] (4). Finally, when using a combination of dilation in the convolutional layers and in the blocks (3), we get a TAP that remains closer to 1 for higher distances, with better performances for sources placed up to 200 Mpc, while keeping the same performances for higher distances. This shows that the combination of Resnet-like blocks and dilation can be used to enhance performances on one-dimensional data.

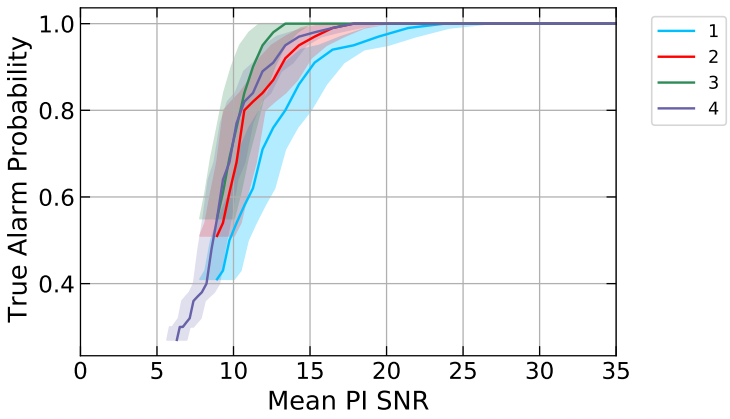

Figure 25: The TAP as a function of the PI SNR for the different networks considered in this work. 1: The architecture from this work without any dilation, performing the worst. 2: The architecture from this work with dilation in the convolutional layers only, performing better 3: The architecture from this paper with dilation in both the convolutional and 1D Resnet layers, performing the best at all PI SNR 4: The architecture from paper [1], outperforming the base networks of this paper, but dropping off at low PI SNR when compared to network 3.

1.1	Architecture with 1D resnet blocks, no dilation								
Layers	Conv	1DResA	Conv	1DResC	Conv	1DResA	Conv	1DResC	Conv
Kernel	128	3,33,63	32	3,33	32	3,33,63	32	3,33	8
Dilation	1	1	1	1	1	1	1	1	1
1.2	Dilation only in the convolutional layers								
Layers	Conv	1DResA	Conv	1DResC	Conv	1DResA	Conv	1DResC	Conv
Kernel	128	3,33,63	32	3,33	32	3,33,63	32	3,33	8
Dilation	1	1	2	1	4	1	8	1	16
1.3	Dilation in all layers								
Layers	Conv	1DResA	Conv	1DResC	Conv	1DResA	Conv	1DResC	Conv
Kernel	128	3,9,12	32	3,9	32	3,9,12	32	3,9	128
Dilation	1	2	4	8	16	32	64	1	1

Table 2: The best performing neural networks when using the 1D-Resnet blocks and no dilation (1.1), when using the dilation only in the convolutional layers(1.2) and when using dilation in both the convolutional and the 1D-Resnet blocks(1.3).

niques in order to improve the performance. First, we have included a network which only has dilation in the convolutional layers. This was in line with [16]. Originally, to keep a consistent input size between the layers, we added padding in these convolutional layers as well. However, as is summarised in the Appendix, this did not result in significant performance increases. When removing this padding, we see significant improvements compared to the first network. The network without padding has a TAP of 100% up until a distance of $125\,Mpc$. This is similar to the network in [1]. The increase in performance with respect to the previous network points to the effectiveness of the dilation for the network when combined with blocks. At higher distances, the TAP drops faster with distance than for the network in [1].

We can see that dilation in the convolutional layers improved the performance of the network significantly. In this work, the best results were obtained when using dilation both in the convolutional layers and in the 1D Resnet blocks. The kernel sizes in the blocks were 3,9 and 12 in the A block and 3 and 9 in the C block. This is a small kernel size compared to the original network. When we use the original kernel sizes of 3, 33 and 63, the

1.1	Standard architecture	
	GW	**No GW**
Gw	1762	238
No GW	173	1827

1.2	Dilation only in the convolutional layers	
	GW	**No GW**
Gw	1833	167
No GW	180	1820

1.3	Dilation in all layers	
	GW	**No GW**
Gw	1906	94
No GW	160	1840

Table 3: The confusion matrices for the standard architecture and the network with dilation in all layers. Showing for every level the true positives (top left), false negatives (top right), true negatives (bottom left) and false positives (bottom right).

network performs worse. We suspect the loss in performance was due to the increase in padding when combining these kernel sizes with dilation. As can be seen in Eq.(28), the amount of padding needed is related to the kernel size by dilation $\times$ kernel size/2. With large dilation and kernel size, this leads to a large number of zeroes, which do not contain any information about the signal and therefore worsen the results.

To exclude the possibility that the smaller kernel size performs better in general, we also tried this kernel size without dilation, but this network performed slightly worse than the original network with blocks but without dilation.

In the best performing network, the dilation got doubled every layer up to a dilation of 64. After this value, no more doubling was possible, as the input size would be smaller than the size needed to output a single value. When looking at the results for this network, we can see that it vastly outperforms the network without dilation. Not only does it have a 100% TAP up to 160 Mpc, it also holds a higher TAP for all higher distances. When comparing this network to the results from [1], we notice that our

network performs better at lower distances. However, at higher distances ($D_L \leq 200Mpc$) the performances are similar. When comparing the PI SNR however, we see significant performance improvements. This network has a TAP of 1 starting at a PI SNR of 14. At lower PI SNR (10-14), it also performs better than the other networks. The network goes back to the performance from [1] at a PI SNR of 10, at which point it only has a TAP of 0.8. This network was not yet trained on other categories of BNS, so we can not compare the networks on this area. In the future, the network from this work should be tested on lighter BNS, as well as on other classes of gravitational wave detection, such as black holes.

5 Conclusion and Discussion

In this work, we have combined various techniques from the machine learning field in order to improve the early warning detection of gravitational waves. For this purpose, we have incrementally added new features to our architectures. The networks have subsequently been trained on simulated heavy binary neutron star data. When using simulated data, we have shown that using a combination of dilation and inception modules can lead to significant improvements in the detection rate of gravitational waves in the early phase of the inspiral of binary neutron stars. The improvements are especially noticeable for samples with a smaller PI SNR. This is important as those samples are the most difficult to detect.

We also want to discuss some aspects of the results as well as propose some new techniques. One can also build upon the results presented in this work, using the newly implemented modules.

- Inception modules

  Regarding the Inception modules, even though significant increase in performance has been shown when using a combination of dilation and inception modules, the network with only inception blocks did not result in good performance. As seen in the results, the network without any dilation is performing worse than an architecture without them for the considered cases. We suspect that other architectures (different combinations of blocs, different number of branches, more layers, ...) might see larger performance improvements when implementing the 1D-Resnet blocks. In future work, it is important to maximise the performance of networks with just inception

modules, as well as the performance of networks with dilation and no inception modules. We suggest implementing a deeper network, as inception modules are usually used in combination with very deep networks.

- Training methods

  Secondly, we believe improvements can be made by using different training methods. As discussed previously, this network is trained in one iteration, on a data set with a PI SNR between 2 and 45. A different training method has already been introduced in [1] and is called *curriculum learning*. When using curriculum learning, the network is trained on increasingly difficult training sets. First, the network is trained on a data set consisting of samples with a high average PI SNR. Then, the network is retrained on sample with a lower average PI SNR. This should increase performance on the samples with a lower PI SNR. Curriculum learning has already shown to result in improvements in neural networks in in [52] and in [1] they have already implemented a test, which pointed to positive results for gravitational wave detection.

- Extension of methods to other signals

  The techniques used for these neural networks are most likely also valid in the detection of other objects in the gravitational-wave field. First, the early inspiral of lighter binary neutron stars could be detected using the same approach. Second, light binary black holes could benefit from the techniques in this work. Finally, 1D data might be used in other detections as well. In some networks, not focused on early warnings, the data is first converted to a spectogram. With the technique from this work, it would not be necessary to do the conversion, which could increase the speed of the networks, while holding a high detection rate [53].

References

[1] G. Baltus, J. Janquart, M. Lopez, A. Reza, S. Caudill, and J.-R. Cudell, The name of the journal **4**, 201 (1993), an optional note.

[2] A. Einstein, Sitzungsberichte der Königlich Preußischen Akademie der Wissenschaften (Berlin pp. 688–696 (1916).

[3] B. Abbott, R. Abbott, T. Abbott, M. Abernathy, F. Acernese, K. Ackley, C. Adams, T. Adams, P. Addesso, R. Adhikari, et al., Physical Review Letters **116** (2016), ISSN 1079-7114, URL http://dx.doi.org/10.1103/PhysRevLett.116.061102.

[4] B. P. Abbott, R. Abbott, T. D. Abbott, F. Acernese, K. Ackley, C. Adams, T. Adams, P. Addesso, R. X. Adhikari, V. B. Adya, et al., The Astrophysical Journal **848**, L12 (2017), ISSN 2041-8213, URL http://dx.doi.org/10.3847/2041-8213/aa91c9.

[5] B. P. Abbott et al. (LIGO Scientific, Virgo), Phys. Rev. Lett. **116**, 241103 (2016), 1606.04855.

[6] *Ligo: all detections*, URL https://dcc.ligo.org/P2000061/public.

[7] B. J. Owen and B. S. Sathyaprakash, Phys. Rev. D **60**, 022002 (1999), URL https://link.aps.org/doi/10.1103/PhysRevD.60.022002.

[8] D. George and E. A. Huerta, Phys. Rev. D **97**, 044039 (2018), 1701.00008.

[9] Classical and Quantum Gravity **32**, UNSP 024001 (2015), ISSN 0264-9381.

[10] J. Miller, L. Barsotti, S. Vitale, P. Fritschel, M. Evans, and D. Sigg, Phys. Rev. D **91**, 062005 (2015), 1410.5882.

[11] H. Yu, R. X. Adhikari, R. Magee, S. Sachdev, and Y. Chen (2021), 2104.09438.

[12] Y.-C. Lin and J.-H. P. Wu, Physical Review D **103** (2021), ISSN 2470-0029, URL http://dx.doi.org/10.1103/PhysRevD.103.063034.

[13] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, Proceedings of the IEEE **86**, 2278 (1998).

[14] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, L. Wang, G. Wang, et al., arXiv e-prints arXiv:1512.07108 (2015), 1512.07108.

[15] X. Lei, H. Pan, and X. Huang, IEEE Access **7**, 124087 (2019).

[16] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, CoRR **abs/1609.03499** (2016), 1609.03499, URL http://arxiv.org/abs/1609.03499.

[17] C. Fang, Y. Shang, and D. Xu, arXiv e-prints arXiv:1709.06165 (2017), 1709.06165.

[18] C. Szegedy, S. Ioffe, and V. Vanhoucke, CoRR **abs/1602.07261** (2016), 1602.07261, URL http://arxiv.org/abs/1602.07261.

[19] A. Einstein, Annalen der Physik **322**, 891 (1905).

[20] F. Bellaiche, *Quantum bits: Basic principles of general relativity*, URL https://www.quantum-bits.org/?p=116.

[21] S. Husa, General Relativity and Gravitation **41**, 1667 (2009).

[22] C. van den Broeck, *Lecture notes on gravitational waves*, Bachelor's Degree Physics, Utrecht University (2018/19).

[23] I. C. . E. Team, *Ligo: Ligo's interferometer*, URL https://www.ligo.caltech.edu/page/ligos-ifo.

[24] D. Martynov, E. Hall, B. Abbott, R. Abbott, T. Abbott, M. Abernathy, K. Ackley, C. Adams, P. Addesso, V. Adya, et al. (2016).

[25] M. Bezares and C. Palenzuela, Class. Quant. Grav. **35**, 234002 (2018), 1808.10732.

[26] K. Chatziioannou et al., Phys. Rev. D **100**, 104015 (2019), 1903.06742.

[27] J. Abadie et al. (LIGO Scientific, VIRGO), Class. Quant. Grav. **27**, 173001 (2010), 1003.2480.

[28] H. Gao, B. Zhang, and H.-J. Lü, Phys. Rev. D **93**, 044065 (2016), 1511.00753.

[29] S. Banagiri, M. W. Coughlin, J. Clark, P. D. Lasky, M. A. Bizouard, C. Talbot, E. Thrane, and V. Mandic, Mon. Not. Roy. Astron. Soc. **492**, 4945 (2020), 1909.01934.

[30] S. Sachdev et al. (2019), 1901.08580.

[31] B. Allen, W. G. Anderson, P. R. Brady, D. A. Brown, and J. D. E. Creighton, Phys. Rev. D **85**, 122006 (2012), URL https://link.aps.org/doi/10.1103/PhysRevD.85.122006.

[32] *Fermi grb detection*, URL https://gcn.gsfc.nasa.gov/other/524666471.fermi.

[33] B. D. Metzger, G. Martínez-Pinedo, S. Darbha, E. Quataert, A. Arcones, D. Kasen, R. Thomas, P. Nugent, I. V. Panov, and N. T. Zinner, **406**, 2650 (2010), 1001.5029.

[34] V. M. Lipunov, E. Gorbovskoy, V. G. Kornilov, N. . Tyurina, P. Balanutsa, A. Kuznetsov, D. Vlasenko, D. Kuvshinov, I. Gorbunov, D. A. H. Buckley, et al., **850**, L1 (2017), 1710.05461.

[35] T. Edition, *Anatomy and Physiology Volume 3 of 3* (Lulu.com, 2014), ISBN 9781304843319, URL https://books.google.nl/books?id=-TuSoAEACAAJ.

[36] J. McGinn, C. Messenger, I. Heng, and M. Williams (2021).

[37] T. Contributers, *Pytorch documentation: Crossentropyloss*, URL https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html.

[38] C. Surai, *Medium, machine learning fundamentals. 2.gradient descent algorithm*, URL http://medium.com/swlh/machine-learning-fundamentals-2-gradient-descent-algorithm-6c8f5204bd9b.

[39] R. Karim, *10 gradient descent optimisation algorithms, towards datascience*, URL https://towardsdatascience.com/10-gradient-descent-optimisation-algorithms-86989510b5e9.

[40] S. Ruder, CoRR **abs/1609.04747** (2016), 1609.04747, URL http://arxiv.org/abs/1609.04747.

[41] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, 2016), http://www.deeplearningbook.org.

[42] J. D. Irwin, *The industrial electronics handbook. Control and mechatronics* (Boca Raton, Florida, 2011), 2nd ed.

[43] V. Dumoulin and F. Visin, *A guide to convolution arithmetic for deep learning* (2018), 1603.07285.

[44] W. Luo, Y. Li, R. Urtasun, and R. Zemel, *Understanding the effective receptive field in deep convolutional neural networks* (2017), 1701.04128.

[45] R. Castro, Y. M.Souto, E. Ogasawara, F. Porto, and E. Bezerra, Neurocomputing **426** (2019).

[46] T. Contributers, *Pytorch documentation: Conv1d*, URL https://pytorch.org/docs/stable/generated/torch.nn.Conv1d.html.

[47] *Convolutional Neural Networks for Visual Recognition*, Department of Computer Science (2020/21), URL http://cs231n.stanford.edu/.

[48] S. Arora, A. Bhaskara, R. Ge, and T. Ma, arXiv e-prints arXiv:1310.6343 (2013), 1310.6343.

[49] K. He, X. Zhang, S. Ren, and J. Sun, CoRR **abs/1512.03385** (2015), 1512.03385, URL http://arxiv.org/abs/1512.03385.

[50] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: with Applications in R* (Springer, 2013), URL https://faculty.marshall.usc.edu/gareth-james/ISL/.

[51] *Ligo: All-sky search for long-duration gravitational wave transients in the first advanced ligo observing run*, URL https://www.ligo.org/science/Publication-O1AllskyLongduration/.

[52] Y. Bengio, J. Louradour, R. Collobert, and J. Weston (Association for Computing Machinery, New York, NY, USA, 2009), ICML '09, p. 41–48, ISBN 9781605585161, URL https://doi.org/10.1145/1553374.1553380.

[53] D. George and E. Huerta, Physics Letters B **778**, 64 (2018), ISSN 0370-2693, URL https://www.sciencedirect.com/science/article/pii/S0370269317310390.

A Appendix

In this section, we compare some of the networks build while optimising the networks presented in the results. Most of these networks have worse performance due to a difference in kernel size or dilation.

A.1 Optimising the network without dilation

First we compare some attempts at optimising the network without dilation. We compare the architectures with different kernel sizes. We do not include the different learning rates, as these either did not let the network learn or they did not influence the performance of the network. We compare 2 different networks: One with smaller kernel sizes, one starting at large kernel sizes, before switching to smaller kernel sizes. The exact values of the kernel sizes can be seen in Table.4 and the comparison of the TAP as a function of the distance of the source can be seen in Fig.26. The TAP of these networks are relatively similar, but we see that the network with the smallest kernel size has a TAP of 1 up until a distance of only 90 Mpc. The reason for this is that this network has a smaller receptive field and less connections between the layers. The second network has slightly larger kernel sizes at the start, increasing the receptive field somewhat. This improves the results. It has a TAP of 1 until a distance of 100 Mpc. The network used in this work has the largest kernel size and performs the best. Increasing the kernel size even further would come with the downside of a smaller input size, making dilation more difficult. We therefore did not implement a larger kernel size.

Layers	Conv	1DResA	Conv	1DResC	Conv	1DResA	Conv	1DResC	Conv
Kernel1	64	3,33,63	16	3,33	16	3,33,63	16	3,33	8
Kernel2	128	3,33,63	64	3,33	16	3,33,63	16	3,33	8

Table 4: The kernel sizes for the tested networks without dilation and with Inception blocks.

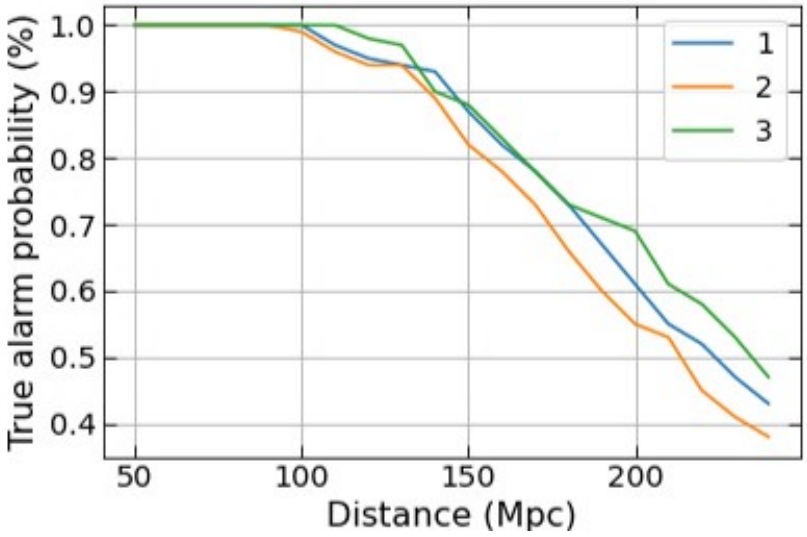

Figure 26: Some of the different architectures tried in the network with inception modules but without dilation. 1: The network with a smaller kernel size in all layers, performing the worst due to a small receptive field. 2: The network with large kernel sizes in the earlier layers, performing better at lower distances. 3: The network as used in the results with the best TAP.

A.2 Optimising the dilation

Several combination of dilation values have been tested. Some different configurations are shown in Tab.5 and Fig.27. We have included a network which has a twice repeating dilation up until a dilation of 8, which is similar to the implementation presented in [16], where they doubled the dilation value up until a value of 512 is reached. After that, they restart with a value of 1 and double the dilation value for each following layer. At these low dilation values, it does not increase performance. We have also included a network which has smaller dilation values. This network performed the worse than the repeating dilation. The suspected reason was that it had the smallest receptive field of the three networks. In fig;27, we represent these two scenarios as well as the best performing architecture for comparison.

Layers	Conv	1DResA	Conv	1DResC	Conv	1DResA	Conv	1DResC	Conv
Dilation1	1	2	4	8	16	1	1	1	1
Dilation2	1	2	4	8	1	2	4	8	1

Table 5: The different configurations of dilation in the network with dilation in both the convolutional and 1D-resnet blocks. Dilation1 is the network with doubling dilation up until a value of 8. Dilation2 is the network with twice repeating dilation up until a value of 8.

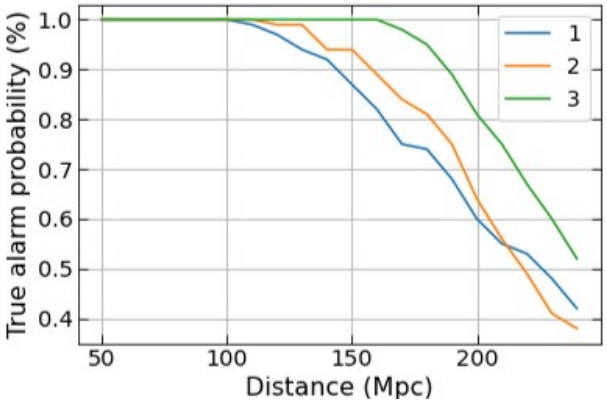

Figure 27: The TAP for different networks with dilation in both the convolutional layers and the *1-D Resnet* blocks layers. 1: the network with doubling dilation up until a value of 16. 2: the network with twice repeating dilation up until a value of 8. This network has a larger receptive field than network 1, resulting in better performance. However it does not perform as good as the best performing networks, as it has smaller dilation in the final layers.