TOKYO INSTITUTE OF TECHNOLOGY

MASTER THESIS

Detection and Localization of Gravitational Waves Using Convolutional Neural Networks

---

Author:
Yuting LIU

Supervisor:
Kentaro SOMIYA

A thesis submitted for the degree of Master of Science

in the

Department of Physics
School of Science

Abstract

The gravitational wave is a ripple in spacetime travelling at the speed of light. This phenomenon was predicted by A. Einstein in 1916 and first directly detected by LIGO in 2015. In this thesis, we constructed Convolutional Neural Networks (CNNs) to analyze the gravitational wave data from compact binary coalescence, which was generated by PyCBC software package. We used the CNN to detect gravitational waves in a noisy background. Then we analyzed the data from LIGO, Virgo, and KAGRA to localize the gravitational wave source, by applying a four- channel CNN. As a result, we confirmed that the CNN algorithm significantly reduces the analysis time cost and also has a strong noise resistance. The localization result also shows the importance of adding KAGRA to the present detector network. Adopting the CNN algorithm and a four-detector network at the same time will contribute to the realization of the multi-messenger astronomy.

Contents

List of Figures

List of Abbreviations

Astrophysics:	
GW	**G**rivitational **W**ave
BBH	**B**inary **B**lack **H**ole
BNS	**B**inary **N**eutron **S**tar
CBC	**C**ompact **B**inary **C**oalescence
SNR	**S**ignal-to-**N**oise **R**atio
PSD	**P**ower **S**pectral **D**ensity
Machine Learning:	
ANN	**A**rtificial **N**eural **N**etwork
CNN	**C**onvolutional **N**eural **N**etwork
MLP	**M**ulti**L**ayer **P**erceptron
Detectors:	
LIGO	**L**aser **I**nterferometric **G**ravitational wave **O**bservatory
H, H1, h1	LIGO **H**anford detector
L, L1, l1	LIGO **L**ivingston detector
V, V1, v1	**V**irgo detector
K, K1, k1	**K**AGRA detector

Chapter 1

Introduction

Gravitational Waves(GW) are generated from accelerating objectives and extending or shrinking the space. GW can be detected by GW detectors distributed in ground and space. The first direct observation of GW was achieved in 2015. LIGO observatories confirmed this GW signal was emitted from a binary black hole. The observation of GW150914 started the study of GW astronomy. In 2017, LIGO&Virgo detected the GW signal from a binary neutron star merger event named GW170817. EM follow-up was also observed due to the alerts from LIGO and GLAST. The joint observation of one binary coalescence started the study of multi-messenger astronomy.

In the study of multi-messenger astronomy, the observation of GW and EM signal is especially important. From the point of view of EM observation, a precise and efficient of GW source localization is demanded. Low latency GW alert would promote the EM telescopes to obtain more signals.

Currently, a GW detector network has been constructing around the world, consists of LIGO in America, Virgo in Europe, KAGRA in Japan, and LIGO-India in India. The simultaneous observation of multiple detectors will benefit the accurate GW localization.

Due to the detectors are located differently, GW detection time of detectors varies. Time lags among the network are depending on the incoming direction of GW. Thus the direction of GW can be measured using time lags. Theoretically, more detectors can locate GW more accurately.

The current data analysis algorithm of the GW signal is based on Matched Filtering. This algorithm compares the detected signal with a GW waveform template library, then selects the most similar template as the optimal waveform. However, the matched filtering requires a lot of time and computing resources to find the GW signal

in the massive data.

In this thesis, we introduce a data analysis algorithm based on Convolutional Neural Network(CNN) to analyze the generated GW waveforms. This algorithm can detect and localize the GW signal in several ten nanoseconds, which significantly saved the computing time.

This thesis is organized as follows: Chapter 2 introduces the theory of GW and noise signals in GW detectors. Chapter 3 talks about the Matched Filtering algorithm and the data analysis methods used by LIGO&Virgo, sky localization pipelines are also included. Chapter 4 talks about the principle and function of CNN. How to implement a GPU-accelerated CNN on a computer is also introduced.

Chapter 5 summarized the previous research of GW data analysis based on neural networks, then talked about a CNN model suitable for the detection of GW. A GW data analysis library, PyCBC, was used to generate waveforms and noises. After repeated training and testing, the CNN was able to distinguish GW waveforms from noises. Chapter 6 talked about the localization of GW using another CNN model. The pros and cons of introducing KAGRA to the detector network were investigated by conducting multiple simulation experiments.

Chapter 7 tested and verified the CNN models by conducting a real data injection experiment. Real data provided by LIGO&Virgo data center was imported then analyzed by CNN. The generalization ability and robustness of CNN were verified. Chapter 8 concluded this thesis, and give future prospects of the research.

In conclusion, this thesis described the superiority of CNN algorithm in the GW data analysis, and the necessity of introducing KAGRA to the detector network. Applying the CNN algorithm to the four-detector network will serve multi-messenger astronomy better.

Chapter 2

Gravitational Wave

2.1 Einstein Equation

According to Einstein's general relativity[1], the line element ds between two points in the space-time is defined as:

$$ds^2 = g_{\mu\nu}dx^\mu dx^\nu \tag{2.1}$$

Here the $g_{\mu\nu}$ is the metric tensor, representing spacetime's structure distorted by the gravity field. The $g_{\mu\nu}$ obeys the Einstein equation

$$G_{\mu\nu} = \frac{8\pi G}{c^4}T_{\mu\nu} \tag{2.2}$$

On one hand, G on the right is the gravitational constant, and $T_{\mu\nu}$is the energy momentum tensor. $T_{\mu\nu}$ is a tensor given by the material in space-time. $T_{\mu\nu} = 0$ when the spacetime is vaccum. On the other hand, the $G_{\mu\nu}$ on the left is the Einstein tensor. It can be written like this:

$$G_{\mu\nu} = R_{\mu\nu} - \frac{1}{2}g_{\mu\nu}R \tag{2.3}$$

In this equation, $R_{\mu\nu}$ is Ricci tensor and R is Ricci scalar. They come from Riemann curvature tensor $R^\alpha_{\mu\alpha\nu}$:

$$R_{\mu\nu} = R^\alpha_{\mu\alpha\nu} \tag{2.4}$$

$$R = g_{\mu\nu}R^\mu_\mu \tag{2.5}$$

Where the Riemann curvature tensor $R^{\alpha}_{\mu\alpha\nu}$ is

$$R^{\gamma}_{\mu\rho\nu} = \frac{\partial\Gamma^{\gamma}_{\mu\nu}}{\partial x^{\rho}} - \frac{\partial\Gamma^{\gamma}_{\mu\rho}}{\partial x^{\nu}} + \Gamma^{\gamma}_{\alpha\rho}\Gamma^{\alpha}_{\mu\nu} - \Gamma^{\gamma}_{\beta\nu}\Gamma^{\beta}_{\mu\rho} \tag{2.6}$$

The Γ item is Christoffel symbol. It can be defined using spacetime metric tensor:

$$\Gamma^{\epsilon}_{\mu v} = \frac{1}{2}g^{\epsilon\sigma}\left(g_{\sigma v,\mu} + g_{\sigma\mu,v} - g_{\mu v,\sigma}\right) \tag{2.7}$$

To simplify the notation, the partial derivative is written in this way:

$$A_{\mu,v} := \frac{\partial}{\partial x^{v}}A_{\mu} \tag{2.8}$$

2.2 Linearization of the Einstein Equation

The spacetime is known as Minkowski space if no matter exists. The Energy momentum tensor $T_{\mu\nu}$ is zero. So the solution of Einstein equation is

$$G_{\mu\nu} = 0 \tag{2.9}$$

In other words, the spacetime is flat. When the matter exists, a perturbation $h_{\mu\nu}$ is added to the flat space. And the metric tensor becomes:

$$g_{\mu\nu} = \eta_{\mu\nu} + h_{\mu\nu} \tag{2.10}$$

Then the Christoffel symbol, Riemann curvature tensor, Ricci curvature tensor, and Ricci scalar becomes:

$$\Gamma^{\lambda}_{\mu\nu} \sim \frac{1}{2}\eta^{\gamma\alpha}\left(\frac{\partial h_{\alpha\mu}}{\partial x^{\nu}} + \frac{\partial h_{\alpha\nu}}{\partial x^{\mu}} - \frac{\partial h_{\mu\nu}}{\partial x^{\alpha}}\right) \tag{2.11}$$

$$R^{\mu}_{\nu\delta\lambda} = \frac{\eta^{\mu\sigma}}{2}\left(h_{\sigma\lambda,\nu\delta} + h_{\nu\delta,\sigma\lambda} - h_{\nu\lambda,\sigma\delta} - h_{\sigma\delta,\nu\lambda}\right) \tag{2.12}$$

$$R_{\nu\lambda} = \frac{\eta^{\delta\sigma}}{2}\left(h_{\sigma\lambda,\nu\delta} + h_{\nu\delta,\sigma\lambda} - h_{\nu\lambda,\sigma\delta} - h_{\sigma\delta,\nu\lambda}\right) \tag{2.13}$$

$$R = \frac{\eta^{\nu\lambda}\eta^{\delta\sigma}}{2}\left(h_{\sigma\lambda,\nu\delta} + h_{\nu\delta,\sigma\lambda} - h_{\nu\lambda,\sigma\delta} - h_{\sigma\delta,\nu\lambda}\right) \tag{2.14}$$

In this situation, the Einstein equation is written as

$$G_{\nu\lambda} = \frac{1}{2}\left[h^{\delta}_{\lambda,\nu\delta} + h^{\delta}_{\nu,\lambda\delta} - \Box h_{\nu\lambda} - h_{,\nu\lambda} - \eta_{\nu\lambda}\left(h^{\delta\sigma}_{,\delta\sigma} - \Box h\right)\right]$$

The h and $\Box$ in this equation are

$$h = \eta^{\mu\nu}h_{\mu\nu} \tag{2.15}$$

$$\Box = \eta^{\mu\nu}\partial_{\mu}\partial_{\nu} \tag{2.16}$$

Define the trace deverce tensor $\widetilde{h}_{\mu\nu}$:

$$\widetilde{h}_{\mu\nu} = h_{\mu\nu} - \frac{1}{2}\eta_{\mu\nu}h \tag{2.17}$$

Because the $h_{\mu\nu}$ can be also expressed as

$$\widetilde{h}_{\mu\nu} = \eta^{\mu\nu}\widetilde{h}_{\mu\nu} = -h \tag{2.18}$$

Replace the $-h$ in the Eintsein equation with *widetildeh*:

$$G_{\nu\lambda} = \frac{1}{2}\left(\widetilde{h}^{\delta}_{\lambda,\nu\delta} + \widetilde{h}^{\delta}_{\nu,\lambda\delta} - \Box\widetilde{h}_{\nu\lambda} - \eta_{\nu\lambda}\widetilde{h}^{\delta\sigma}_{,\delta\sigma}\right) \tag{2.19}$$

Consider the following Gauge conversion:

$$x'\mu = x^{\mu} + \xi^{\mu}(x) \tag{2.20}$$

So the metric tensor is

$$\begin{aligned} g'_{\mu v} &= \frac{\partial x^{\alpha}}{\partial x'^{\mu}}\frac{\partial x^{\beta}}{\partial x'^{\nu}}g_{\alpha\beta} \\ &= \left(\delta^{\alpha}_{\mu} - \xi^{\alpha}_{\mu}\right)\left(\delta^{\beta}_{v} - \xi^{\beta}_{,v}\right)\left(\eta_{\alpha\beta} + h_{\alpha\beta}\right) \\ &= \eta_{\mu v} + h_{\mu\nu} - \xi_{\mu,v} - \xi_{v,\mu} \end{aligned} \tag{2.21}$$

The perturbation is also transformed:

$$h'_{\mu\nu} = g'_{\mu\nu} - \eta_{\mu\nu} = h_{\mu\nu} - \xi_{\mu,\nu} - \xi_{\nu,\mu} \tag{2.22}$$

$$h' = h - 2\xi^{\sigma}_{,\sigma} \tag{2.23}$$

In the meantime, the trace deverce tensor $\widetilde{h}_{\mu\nu}$ is

$$\tilde{h}'_{\mu v} = h'_{\mu v} - \frac{\eta_{\mu v}}{2}h' = \tilde{h}_{\mu v} - \xi_{\mu,v} - \xi_{v,\mu} + \eta_{\mu\nu}\xi^{\sigma}_{,\sigma} \tag{2.24}$$

Note that the Riemann curvature tensor is invariant to this gauge transformation. Because we have

$$\tilde{h}'^{\mu}_{v,\mu} = \tilde{h}^{\mu}_{v,\mu} - \Box\xi_v \tag{2.25}$$

If we choose one ξ^{μ} that satisfies wave equation, $\tilde{h}'^{\mu}_{v,\mu} = 0$ can be satisfied. This condition is named a harmonic condition. From now on, we omit all the $'$ in the equations, which means the harmonic condition is always satisfied. In this harmonic condition, Einstein tensor can be written as:

$$G_{\mu v} = -\frac{1}{2}\Box\tilde{h}_{\mu v} \tag{2.26}$$

Bring the 2.2 to 2.26 can get the following equation:

$$\Box\widetilde{h}_{\mu\nu} = -\frac{16\pi G}{c^4}T_{\mu\nu} \tag{2.27}$$

2.3 Solution of Einstein euqation

Linearized Einstein equation solution is 2.27. In a flat spacetime, the energy momentum tensor $T_{\mu\nu}$ is zero. Thus, we can transform the 2.27 further:

$$\Box\widetilde{h}_{\mu\nu} = 0 \tag{2.28}$$

Here we consider the $\widetilde{h}_{\mu\nu}$ a plane wave:

$$\widetilde{h}_{\mu\nu} = a_{\mu\nu}\exp\left(ik_{\lambda}x^{\lambda}\right) \tag{2.29}$$

Combine 2.28 and 2.29 we can get

$$\begin{aligned} \Box a_{\mu\nu} \exp\left(ik_\lambda x^\lambda\right) &= a_{\mu\nu}\eta^{\lambda\sigma}\partial_\lambda\partial_\sigma \exp\left(ik_\lambda x^\lambda\right) \\ &= \eta^{\lambda\sigma}k_\lambda k_\sigma a_{\mu\nu} \exp\left(ik_\lambda x^\lambda\right) \\ &= 0 \end{aligned} \tag{2.30}$$

To satisfy the equation

$$\eta^{\lambda\sigma}k_\lambda k_\sigma a_{\mu\nu} = 0 \tag{2.31}$$

We have to let

$$\eta^{\lambda\sigma}k_\lambda k_\sigma = 0 \tag{2.32}$$

Similarly, bringing 2.29 to the harmonic condition equation(1.28) will get this result:

$$\eta^{\lambda v}k_v a_{\mu\lambda} = 0 \tag{2.33}$$

Therefore, a plane wave represents by 2.29 will have such solution:

$$A_{\mu\nu}k^\nu = 0 \tag{2.34}$$

$$k_\mu k^\mu = 0 \tag{2.35}$$

Here, 2.34 represents the amplitude of the plane wave is vertical to wave travelling direction. This indicates that the gravitational wave is a transverse wave. The 2.35 represents gravitational wave travels at the speed of light.

Here we take the transverse-traceless gauge(TT Gauge). From 2.28, to satisfy the harmonic condition, ξ_v should be the solution of the wave function. Use B_μ to describe the wave function, we have

$$\xi_\mu = B_\mu \exp\left(ik_\lambda x^\lambda\right) \tag{2.36}$$

Bring this equation to 2.22 and 2.23:

$$h'_{\mu\nu} = h_{\mu\nu} - i\left(B\mu k_\nu + B_\nu k_\mu\right) \exp\left(ik_\lambda x^\lambda\right) \tag{2.37}$$

$$\begin{aligned} h' &= h - i\eta^{\mu\nu}\left(B\mu k_\nu + B_\nu k_\mu\right)\exp\left(ik_\lambda x^\lambda\right) \\ &= -\left[\eta^{\mu\nu}A_{\mu\nu} + i\eta^{\mu\nu}\left(B_\mu k_\nu + B_\nu k_\mu\right)\right]\exp\left(ik_\lambda x^\lambda\right) \end{aligned} \tag{2.38}$$

There always exists a B_μ that let $h' = 0$. This solution would satisfy traceless condition, which means $h_{\mu v} = \tilde{h}_{\mu v}$. Now we suppose the gravitational wave comes from z direction. Then we have $-k_0 = k_3 = k, k_1 = k_2 = 0$. The harmonic condition and traceless condition are:

$$k\left(A_{\mu 0} + A_{\mu 3}\right) = 0 \tag{2.39}$$

$$-A_{00} + A_{11} + A_{22} + A_{33} = 0 \tag{2.40}$$

In this situation, we can let $h'_{0\mu} = 0$. Let:

$$h'_{\mu\nu} = A'_{\mu\nu}\exp\left(ik_\lambda x^\lambda\right) \tag{2.41}$$

Then the equation(1.37) can be expressed as:

$$A'_{00} = A_{00} + 2iB_0 k \tag{2.42}$$

$$A'_{01} = A_{01} + iB_1 k \tag{2.43}$$

$$A'_{02} = A_{02} + iB_2 k \tag{2.44}$$

$$A'_{03} = A_{03} - i\left(B_0 k - B_3 k\right) = A_{03} - 2iB_0 k \tag{2.45}$$

Summarize the conditions to the solutions coefficients:

$$\begin{aligned} &\bullet\ -a_{00} + a_{11} + a_{22} + a_{33} = 0((\text{Traceless condition})) \\ &\bullet\ a_{\mu 0} + a_{\mu 3} = 0(\text{Harmonic condition}) \\ &\bullet\ a_{\mu 0} = 0 \\ &\bullet\ h_{\mu v} = h_{v\mu}((\text{Opposition condition})) \end{aligned} \tag{2.46}$$

From harmonic condition and symmetry of the condition, $h_{\mu v}$ is:

$$h_{\mu v} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & a_{11} & a_{12} & a_{13} \\ 0 & a_{21} & a_{22} & a_{23} \\ 0 & a_{31} & a_{32} & a_{33} \end{pmatrix} \exp i(-\omega t + kz) \tag{2.47}$$

Combine harmonic condition and opposition condition:

$$h_{\mu v} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & a_{11} & a_{12} & 0 \\ 0 & a_{21} & a_{22} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \exp i(-\omega t + kz) \tag{2.48}$$

Combine the traceless condition and opposition condition:

$$h_{\mu v} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & a_{11} & a_{12} & 0 \\ 0 & a_{12} & -a_{11} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \exp i(-\omega t + kz) \tag{2.49}$$

Usually we use the following coefficients to describe gravitational wave:

$$h_{\mu v} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & h_+ & h_\times & 0 \\ 0 & h_\times & -h_+ & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \exp i(-\omega t + kz) \tag{2.50}$$

The h_+ and $h_\times$ are polarizations of gravitational wave. h_+ is 'plus mode' and $h_\times$ is 'cross mode'.

$$h_+ = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & h_+ & 0 & 0 \\ 0 & 0 & -h_+ & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, h_\times = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & h_\times & 0 \\ 0 & h_\times & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.51}$$

By taking an appropriate gauge, the wave can satisfy the traceless condition as a transverse wave. Such gauge used here is called Transverse Traceless Gauge(TT Gauge).

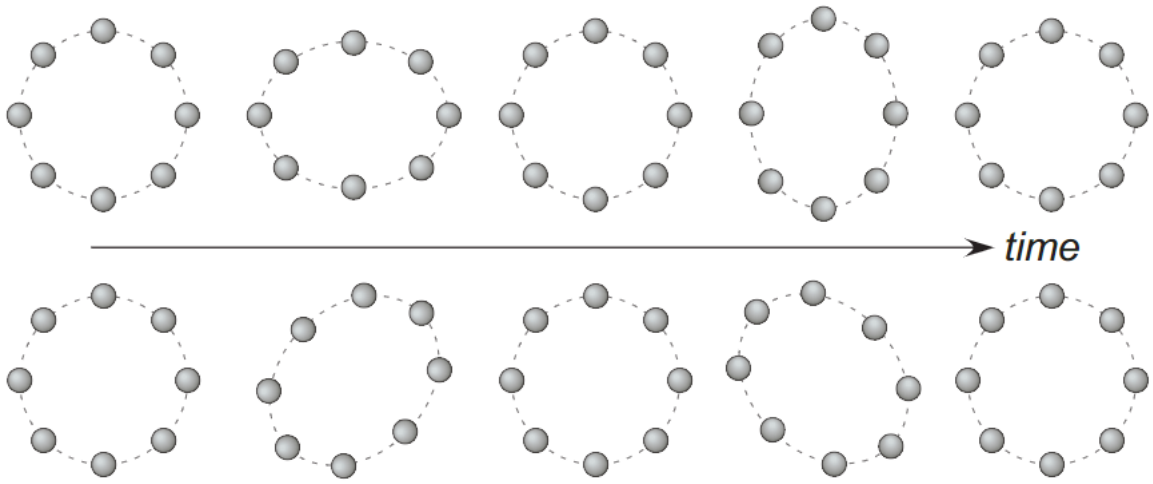

FIGURE 2.1: Distance between free particles rings when gravitational wave go though. The gravitational wave's propagation direction is perpendicular to the paper. The ring above represents the infection of plus mode, the ring below represents the effect of cross mode.

2.4 Effect on free particles

Consider the effect of gravitational wave on two free particles whose positions are (0,0,0) and (ϵ,0,0) in TT Gauge. Assume a gravitational wave pass though the particles when they are at rest, then the distance Δl between the particles should be:

$$\begin{aligned}\Delta l &\equiv \int \left|ds^2\right|^{\frac{1}{2}} = \int \left|g_{\alpha\beta}dx^\alpha dx^\beta\right|^{\frac{1}{2}} \\ &= \int_0^\epsilon |g_{xx}|^{\frac{1}{2}}\, dx \simeq |g_{xx}(x=0)|^{\frac{1}{2}}\,\epsilon \\ &\simeq \left[1+\frac{1}{2}h_{xx}(x=0)\right]\epsilon\end{aligned} \tag{2.52}$$

This equation indicates that the distance between the points is changing with time. As Figure 2.1 shows, the gravitational wave can be detected by measuring the displacement of the particles.

2.5 Emission of Gravitational Wave

Here we consider the emission of a linearized gravitational wave. Use Green's function to transfer 2.28 with a d'Alembert operator $G\left(x^{\sigma}-y^{\sigma}\right)$:

$$G\left(x^{\sigma}-y^{\sigma}\right)=-\frac{1}{4\pi|\boldsymbol{x}-\boldsymbol{y}|}\delta\left(|\boldsymbol{x}-\boldsymbol{y}|-\left(x^{0}-y^{0}\right)\right)\theta\left(x^{0}-y^{0}\right) \tag{2.53}$$

$$\Box G\left(x^{\sigma}-y^{\sigma}\right)=\delta^{(4)}\left(x^{\sigma}-y^{\sigma}\right) \tag{2.54}$$

where $\sigma=0,1,2,3$, $\delta(x)$ and $\theta(x)$ are the Dirac's δ function and the step function, respectively. Also assume that x and y are the position of the obervototy and the gravitational wave source. So the distance between x and y is given by:

$$|\boldsymbol{x}-\boldsymbol{y}|=\left\{\delta_{ij}\left(x^{i}-y^{i}\right)\left(x^{j}-y^{j}\right)\right\}^{1/2} \tag{2.55}$$

Assume the source is far enough to observe, which means $r=|x|\gg|y|$, then $\bar{h}_{\mu\nu}$ can be written is this way:

$$\begin{aligned}\bar{h}_{\mu\nu}(t,\boldsymbol{x})&=-\frac{16\pi G}{c^{4}}\int G\left(x^{\sigma}-y^{\sigma}\right)T_{\mu\nu}\left(y^{\sigma}\right)d^{4}y\\&=\frac{4G}{c^{4}}\int\frac{T_{\mu\nu}(ct-|\boldsymbol{x}-\boldsymbol{y}|,\boldsymbol{y})}{|\boldsymbol{x}-\boldsymbol{y}|}d^{3}y\\&\simeq\frac{4G}{rc^{4}}\int T_{\mu\nu}(ct-r,\boldsymbol{y})d^{3}y\end{aligned} \tag{2.56}$$

The integration in this equation is given by:

$$\int T^{ij}d^{3}y=\frac{1}{2c^{2}}\frac{d^{2}}{dt^{2}}\int T^{00}y^{i}y^{j}d^{3}y=\frac{1}{2}\frac{d^{2}}{dt^{2}}\int\rho y^{i}y^{j}d^{3}y \tag{2.57}$$

The last term is called quadrupole mass distribution I_{ij}:

$$I_{ij}\equiv\int\rho y^{i}y^{j}d^{3}y \tag{2.58}$$

So combining 2.57 and 2.58, the $\bar{h}_{ij}$ can be written like:

$$\bar{h}_{ij}(t,\boldsymbol{x})=\frac{2G}{rc^{4}}\ddot{I}_{ij}(ct-r) \tag{2.59}$$

We can find that the amplitude of gravitational wave is proportional to the second derivative of the quadrupole, and inversely proportional to the distance between observatory and the source.

2.6 Michelson Interferometer

To detect the gravitational wave, the most commonly used detector is the Michelson Interferometer[2]. As Figure 2.2 shows, a laser emits from the left, it can be defined

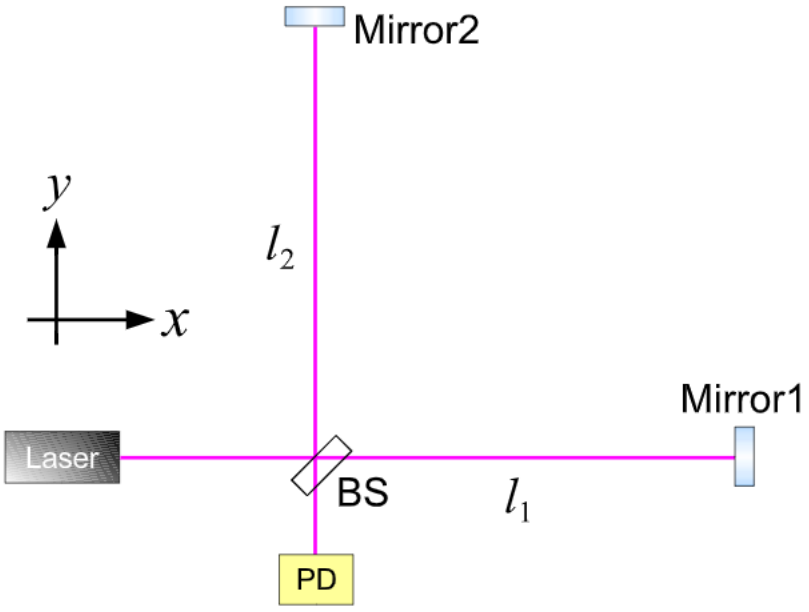

FIGURE 2.2: A Michelson Interferometer

as:

$$E_{\text{in}} = E_0 e^{i\Omega t} \tag{2.60}$$

Then the Beam Splitter(BS) divides the laser to two beams. Then these beams are reflected by the two mirrors, then combined together at the position of BS. We assume that during this process the phase changes of the two beams are ϕ_1 and ϕ_2, then the combined beam at the BS can be written as:

$$E_{\text{PD}} = E_1 e^{i(\Omega t - \phi_1)} - E_2 e^{i(\Omega t - \phi_2)} \tag{2.61}$$

Here the E_1 and E_2 are the amplitudes of the beams. Theoretically, $E_1 = E_2 = \frac{1}{2}E_0$, but in a real interferometer their value are not the same, duo to the different reflectivities of the mirrors. Then the laser are detected by Photo Detector(PD). The laser power detected by PD is:

$$\begin{aligned} P_{\mathrm{PD}} = |E_{\mathrm{PD}}|^2 &= E_1^2 + E_2^2 + 2E_1E_2\cos(\phi_1 - \phi_2) \\ &= \frac{P_{\max} + P_{\min}}{2} + \frac{P_{\max} - P_{\min}}{2}\cos(\phi_1 - \phi_2) \end{aligned} \tag{2.62}$$

Here,

$$\begin{aligned} P_{\max} &\equiv (E_1 + E_2)^2 \\ P_{\min} &\equiv (E_1 - E_2)^2 \end{aligned} \tag{2.63}$$

The power of the laser is a function of the cosine of phase difference $\phi_1 - \phi_2$. When $\phi_1 - \phi_{2=0}$ the power has a maximum value, and when $\phi_1 - \phi_2 = \pi$ the power has the minimum value. To let the laser visibility as strong as possible, keep the phase difference $\phi_1 - \phi_2$ small is essential.

2.7 Response of a Michelson Interferometer to the Gravitational wave

Consider in a Cartesian coordinate, two arms of the interferometer are parallel to the x and y axes. A plus mode gravitational wave h_+ comes from $+z$ direction. Under TT Gauge, the line element is written as:

$$ds^2 = -c^2dt^2 + (1+h)dx^2 + (1-h)dy^2 + dz^2 \tag{2.64}$$

Because the laser in our world line satisfies $ds^2 = 0$, so this equation can be transformed like:

$$\frac{dx}{dt} = \pm\frac{c}{\sqrt{1+h}} \simeq \pm\left(1 - \frac{1}{2}h\right)c \tag{2.65}$$

where the $+$ sign represents the laser traveling in the positive x direction and the $+$ sign represents traveling in the negative x direction. Assume the one interferometer's

arm length is l_1, so the beam in this arm traveled

$$\int dx = 2l_1 = c\int_{t-\tau_1}^{t}\left\{1-\frac{1}{2}h\left(t'\right)\right\}dt' \tag{2.66}$$

long in total. Because the beam propagates in the speed of light, the time spend is:

$$\tau_1 = \frac{2l_1}{c}+\frac{1}{2}\int_{t-\tau_1}^{t}h\left(t'\right)dt' \approx \frac{2l_1}{c}+\frac{1}{2}\int_{t-2l_1/c}^{t}h\left(t'\right)dt' \tag{2.67}$$

In the last integral $\tau_1 \approx 2l_1/c$ because $h \ll 1$. Thus, the phase change during the propagating is:

$$\phi_1 = \Omega\tau_1 = \frac{2l_1\Omega}{c}+\frac{\Omega}{2}\int_{t-2l_1/c}^{t}h\left(t'\right)dt' \tag{2.68}$$

Similarly, the phase change of another beam traveling in another arm should be:

$$\phi_2 = \frac{2l_2\Omega}{c}-\frac{\Omega}{2}\int_{t-2l_2/c}^{t}h\left(t'\right)dt' \tag{2.69}$$

Let:

$$l_1 \simeq l_2 \simeq l \tag{2.70}$$

$$l_- = l_1 - l_2 \tag{2.71}$$

We will have:

$$\phi_1-\phi_2 = \frac{2l_-\Omega}{c}+\delta\phi_{\mathrm{GW}} \tag{2.72}$$

$$\delta\phi_{\mathrm{GW}} = \Omega\int_{t-2l/c}^{t}h\left(t'\right)dt' \tag{2.73}$$

The 2.73 indicates the phase difference of two beams. If the gravitational wave comes to the interferometer, the phase difference will change the interference pattern on the BS. This is how does interferometer detect the gravitational wave.

2.8 Frequency respond of Interferometer

Now we talk about the respond of the interferometer to different frequency gravitational waves. In frequency domain, the Fourier transformation of $h(t)$ is:

$$h(t) = \int_{-\infty}^{\infty} h(\omega) e^{i\omega t} dt \tag{2.74}$$

Bring 2.74 to 2.73:

$$\begin{aligned} \delta\phi_{\mathrm{GW}} &= \Omega \int_{t-2l/c}^{t} \int_{-\infty}^{\infty} h(\omega) e^{i\omega t'} d\omega dt' \\ &= \int_{-\infty}^{\infty} \frac{2\Omega}{\omega} \sin\left(\frac{l\omega}{c}\right) e^{-il\omega/c} h(\omega) e^{i\omega t} d\omega \end{aligned} \tag{2.75}$$

Here we introduce H_{MI}, representing the frequency response of the interferometer to a gravitational wave:

$$H_{\mathrm{MI}}(\omega) = \frac{2\Omega}{\omega} \sin\left(\frac{l\omega}{c}\right) e^{-il\omega/c} \tag{2.76}$$

Then the 2.75 will be:

$$\delta\phi_{\mathrm{GW}} = \int_{-\infty}^{\infty} H_{\mathrm{MI}}(\omega) h(\omega) e^{i\omega t} d\omega \tag{2.77}$$

As we can see, the phase change of the interferometer is depending on the arm length. The interferometer is most sensitive when the phase difference between the two beams is π. From this point, we can choose an optimal baseline length l that can maximize the effect of a fixed frequency gravitational wave. To a $1kHz$ gravitational wave, the optimum base length is about $75km$. It's nearly impossible to construct such a long interferometer on earth. To save the construction cost people developed Fabry-Perot Interferometer which contains Fabry-Perot cavity that can reflect beams for many times to enlarge the phase difference. This technology has been applied to existing gravitational wave detectors, but also brought stronger noise to the interferometer.

2.9 Noise in the Michelson Interferometer

The noise is restricting the sensitivity of interferometers[3]. How to restrain the noise level is always a primary study to the gravitational wave researchers. Here I

introduce the main noise exist in interferometers briefly.

2.9.1 Shot noise

Shot noise come from the quantum fluctuation of photon number in the laser beams. Error of photon counting from the PD will cause a pseudo mirror displacement. Assume the photo current injects into the PD is i_P. According to Poisson photon-statistics, the power spectrum of i_P is:

$$i_{shot} = \sqrt{2ei_P} \tag{2.78}$$

where e is the elementary charge. Assume the phase difference between the beams is:

$$\phi_1 - \phi_2 = \Phi_0 + \delta\phi \tag{2.79}$$

From 2.62, the power of photon vibration detected by PD is:

$$\delta I_P = -\frac{I_{max} - I_{min}}{2} \sin(\Phi_0)\, \delta\phi \tag{2.80}$$

The I_{max} and I_{min} are the maximam and minimum current. Consider the minimum phase difference that could be detected by PD, the $\delta\phi_{min}$ should be:

$$\delta\phi_{min} = \frac{2\sqrt{2eI_P}}{I_{max} - I_{min}} \frac{1}{\sin\Phi_0} \tag{2.81}$$

Here the photon current I_P is defined as:

$$I_P = \frac{I_{max} + I_{min}}{2} + \frac{I_{max} - I_{min}}{2} \cos\Phi_0 \tag{2.82}$$

Assume the visibility is optimal, which means $I_{min} = 0$, the 2.81 becomes:

$$\delta\phi_{min} = \sqrt{\frac{2e}{I_{max}}} \frac{1}{\sin(\Phi_0/2)} \tag{2.83}$$

When $\Phi_0 = \pi$, the $\delta\phi_{min}$ has a minimum value.

$$\delta\phi_{min} = \sqrt{\frac{2e}{I_{max}}} \tag{2.84}$$

That is to say, when the PD is at a dark fringe, the shot noise is most restrained. As we can see from 2.83, the shot noise is inversely proportional to the square root of $I_{\max}$. Here we use PD's efficiency η and laser power P to indicate $I_{\max}$:

$$I_{\max} = e\frac{\eta P}{\hbar\Omega} \tag{2.85}$$

Combine 2.84 and 2.85, we can have the shot noise at dark fringe:

$$\delta\phi_{\min} = \sqrt{\frac{2\hbar\Omega}{\eta P}} \tag{2.86}$$

As we can see, the shot noise is inversely proportional to the laser power P. Thus, increasing laser power can reduce the level of shot noise. On the other hand, similarly with the purpose and theory of the Fabry-Perot cavity, introducing a 'Power Recycling Mirror' to the interferometer can increase the effective beam power[4]. The mirror locates between laser and beam splitter can reflect the beam come back from the beam splitter, so the beam is less easy to leak from the interferometer.

2.9.2 Radiation Pressure Noise

When a photon is reflected by a mirror, it gives the mirror a force that vibrates the mirror. The force is fluctuating due to the fluctuation of the photon number. This force is called 'radiation pressure noise.' In a Fabry-Perot Michelson Interferometer, the displacement of mirror induced by radiation pressure noise is given by

$$\delta x_{\text{radi}} = \frac{4\mathcal{F}}{\pi m\omega^2}\left[\frac{2hP_0}{\lambda c\left[1+(\frac{\omega}{\omega_c})^2\right]}\right]^{\frac{1}{2}} \tag{2.87}$$

Where

$$\omega_{\text{c}} = \frac{\pi c}{2L\mathcal{F}} \tag{2.88}$$

The unit of δx_{radi} is $\text{m}/\sqrt{\text{Hz}}$. ω_c is the angular cut off frequency of arm cavities. The $\mathcal{F}$ is the finesse of cavities, meaning how many times can the laser transfer between the mirrors. λ is the laser wavelength, P_0 is the laser power, m is the mass of mirrors, L are the arm lengths. As we can see from the 2.87, the displacement is inversely

proportional to the frequency. Thus in the low-frequency range, the interferometer's sensitivity is more seriously restricted by radiation pressure noise. In the meantime, we notice that the radiation pressure noise is also proportional to the square root of laser power P_0, indicating there is a trade-off between shot noise and radiation pressure noise. Thus, the limitation of the sensitivity of interferometer is called the standard quantum limit(SQL):

$$h_{\mathrm{SQL}} = \frac{1}{L\omega}\sqrt{\frac{8\hbar}{m}} \tag{2.89}$$

This limitation indicates the mechanical limit of the measurement of the laser. People developed many methods to overcome the limit, such as quantum squeezing.[5][6]

2.9.3 Thermal Noise

The detector's sensitivity is also restrained by temperature fluctuation of the mirrors and its suspension systems. KAGRA has adopted a cryogenic interferometry to lower the thermal noise, but the effect of temperature fluctuation is still not solved yet. According to the fluctuation dissipation theorem (FDT)[7], the system in a heat bath fluctuates. The thermal fluctuation is proportional to the mechanical loss in the body. The thermal noise due to the incident of laser, and heat exchange between mirror and suspension system is called mirror thermal noise. The vibration of the mirror body as a pendulum is called suspended thermal noise. According to the FDT, the thermal noise can be analyzed by seeing the mirror and suspension system like a pendulum. The thermal noise of an oscillator can be written as:

$$\left\langle x(\omega)^2 \right\rangle = \frac{4k_{\mathrm{B}}T}{mQ}\frac{\omega_0}{\left(\omega^2-\omega_0^2\right)^2+\omega_0^2\omega^2/Q^2} \tag{2.90}$$

Here the k_{B} is the Boltzmann constant, T is the temperature, m is the mirror mass, ω_0 is the angular resonant frequency of the system. Introducing the loss angel ϕ:

$$\phi(\omega) = \frac{\omega}{\omega_0 Q} \quad : \text{ Viscous damping model} \tag{2.91}$$

$$\phi = \frac{1}{Q} \quad : \text{Structure damping model} \tag{2.92}$$

The loss angle ϕ is related to the argument of the mechanical admittance Y of the system:

$$\arg(Y(\omega)) = \arctan(\phi(\omega)) \tag{2.93}$$

We can infer that decreasing the temperature or increasing the quality factor can reduce the thermal noise.

2.9.4 Seismic Noise

To every ground-based interferometer, the seismic noise is an inevitable problem. Although the mirrors are suspended by the suspension system for avoiding the effect of ground, the suspension system is not isolated from the ground totally. Generally speaking, the seismic noise is a function of frequency and varied from the specific location of the interferometer. The seismic noise can be roughly expressed by:

$$\delta x_{\text{gnd}} \sim 10^{-7} \frac{1}{f^2} \tag{2.94}$$

Obviously, it is also restricting the sensitivity mainly in the low-frequency range. To expand the frequency detection range, developing a suspension system having strong resistance to low-frequency seismic noise is essential.

2.10 Gravitational Wave Resource

Gravitational waves can be emitted from any item that involves their quadrupole moment changing. But since usual items contain small mass can only emit a small amplitude gravitational wave, a detectable gravitational wave resource is mainly an astronomical origin. In this section, we will discuss the common types of gravitational wave sources.

2.10.1 Coalescence of Compact Binary Stars

A coalescence of compact binary stars means a pair of stars such as a binary black hole or binary neutron star approach to each other because of gravity. The start of their merge process is called 'Inspiral'. Gravitational waves are emitted in the last few minutes at the inspiral period. Then when their orbit coincides, the strongest gravitational

wave emits. We call this period 'Merger'. Finally, the binaries become one black hole, and the gravitational wave is decayed at a rapid speed. Because the waveform looks like the ring of a bell, we call this period 'Ringdown'.

2.10.2 Supernova

During the collapse of a massive star, if the gravity center of the star is asymmetric, a gravitational wave would be emitted. The waveform of supernova's gravitational wave is hard to predict, we can only estimate its amplitude as about 10^{-21} to 10^{-20}.[8]

2.10.3 Pulsars

A spinning neutron star will emit gravitational waves if it is asymmetrical to its rotation axis. Pulsars are rotating in a frequency as few hundred Hertz. Such neutron stars are called pulsars. Pulsars would also emit radio beams, and the arrival time of radio beams is not simultaneous with gravitational wave's arrival time. This is due to the gravitational wave is stretching and squeezing the space. We have already observed the time delay by using different observatories. The observation of one source using combined observatories is called 'Multi-messenger astronomy', this study will help us understand the characteristics of neutron stars and black holes more comprehensively.[9]

2.10.4 Cosmological Sources

Various solutions about the gravitational waves generated during the inflation in the early universe have been posted. The amplitudes and frequencies in these theories vary. The estimated frequency is from $10^{9}Hz$ to $10^{-16}Hz$, and the estimated amplitude is 10^{-17} at $10^{-8}Hz$[10]. If the gravitational wave generated from the inflation is detected, we can obtain the information of the universe that is only 10^{-22} second old, which is impossible to electromagnetic observation.

Since the cosmological gravitational wave is considered random, a coincidence test between the independent observatories is essential for detecting the cosmological source.

Chapter 3

Data Analysis of Gravitational Wave by Matched Filtering

This chapter briefly describes the background of gravitational wave source localization by GW detectors. Localization is figuring out the detection of the GW source. So far, this research is based on an algorithm named 'Matched filtering', which was applied during LIGO and Virgo's observation[11, 12, 13, 14].

3.1 Matched Filtering

3.1.1 Optimal Filter

Matched filtering is a common analysis method for the searching of GW signals from the data contain noises. This algorithm was studied in the field of signal-processing, and also known as the optimal filter. Assume the measured waveform from an observatory is:

$$s(t) = h(t) + n(t) \tag{3.1}$$

Here, the $h(t)$ is GW signal and $n(t)$ is noise in the detector.

The two-sided noise power spectral sensitivity(PSD) is $S_n(f)$, which is defined by

$$\left\langle \tilde{n}(f)\tilde{n}^*\left(f'\right)\right\rangle = \delta\left(f - f'\right) S_n(f) \tag{3.2}$$

Define a real-valued filter $F(t)$ as:

$$A = \int_{-\infty}^{\infty} F(t)a(t)dt = \int_{-\infty}^{\infty} \tilde{F}^*(f)\tilde{a}(f)df \tag{3.3}$$

In 3.3, A is the filtered value of $a(t)$. $\tilde{a}(f)$ comes from the Fourier transform of $a(t)$:

$$\tilde{a}(f) = \int_{-\infty}^{\infty} a(t) e^{-2\pi i f t} dt \tag{3.4}$$

Combine 3.2, 3.3 and 3.4, we have:

$$\begin{aligned} \left\langle N^2 \right\rangle &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \tilde{F}^*(f) \tilde{F}^*\left(f'\right) \left\langle \tilde{n}(f) \tilde{n}\left(f'\right) \right\rangle df' df \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \tilde{F}^*(f) \tilde{F}^*\left(f'\right) \delta\left(f - f'\right) S_n(f) df' df \\ &= \int_{-\infty}^{\infty} |\tilde{F}(f)|^2 S_n(f) df \end{aligned} \tag{3.5}$$

[1]

Now we try to find a optimal filter, which can extract the GW signal from the noise best. We can get this filter by maximizing the ratio of filtered values:

$$\begin{aligned} \frac{H^2}{\langle N^2 \rangle} &= \frac{\left| \int_{-\infty}^{\infty} \tilde{F}^*(f) \tilde{h}(f) df \right|^2}{\int_{-\infty}^{\infty} |\tilde{F}(f)|^2 S_n(f) df} \\ &= \frac{\left| \int_{-\infty}^{\infty} \tilde{F}^*(f) \sqrt{S_n(f)} \tilde{h}(f) / \sqrt{S_n(f)} df \right|^2}{\int_{-\infty}^{\infty} |\tilde{F}(f)|^2 S_n(f) df} \end{aligned} \tag{3.6}$$

Using Cauchy-Schwarz inequality,

$$\left| \int_{-\infty}^{\infty} A(f) B(f) df \right|^2 \leq \int_{-\infty}^{\infty} |A(f)|^2 df \int_{-\infty}^{\infty} |B(f)|^2 df \tag{3.7}$$

we can claim that in order to let 3.6 have a maximum value, $\tilde{F}^*(f)\sqrt{S_n(f)}$ and $\tilde{h}(f)/\sqrt{S_n(f)}$ must be equal. If so, 3.6 is now written by:

$$\begin{aligned} \frac{H^2}{\langle N^2 \rangle} &= \frac{C \left(\int_{-\infty}^{\infty} |\tilde{F}(f)|^2 S_n(f) df \right) \left(\int_{-\infty}^{\infty} |\tilde{h}(f)|^2 / S_n(f) df \right)}{\int_{-\infty}^{\infty} |\tilde{F}(f)|^2 S_n(f) df} \\ &= C \int_{-\infty}^{\infty} \frac{\tilde{h}^*(f) \tilde{h}(f)}{S_n(f)} df \end{aligned} \tag{3.8}$$

Where C is a constant number. From this discussion we find that the optimal filter for $h(t)$ is

$$\tilde{F}^*(f) = C \frac{\tilde{h}(f)}{S_n(f)} \tag{3.9}$$

The inner product of the detected data s and the template h is defined by:

$$(h \mid s) = \int_{-\infty}^{\infty} \frac{\tilde{h}^*(f)\tilde{s}(f)}{S_n(f)} df \tag{3.10}$$

We can divide the denominator to form the amplitude spectral densitities(ASDs).

$$(h \mid s) = \int_{-\infty}^{\infty} \frac{\tilde{h}^*(f)}{\sqrt{S_n(f)}} \frac{\tilde{s}(f)}{\sqrt{S_n(f)}} df \tag{3.11}$$

Then we difine $\bar{h}(f) = \frac{\tilde{h}^*(f)}{\sqrt{S_n(f)}}$, $\bar{s}(f) = \frac{\tilde{s}^*(f)}{\sqrt{S_n(f)}}$. Then 3.11 becomes:

$$(h \mid s) = \int_{-\infty}^{\infty} \bar{h}^*(f)\bar{s}(f) df \tag{3.12}$$

3.1.2 Waveform Overlap

So far, we have obtained the optimal filter to extract the GW waveform from the signal. Define a overlap M of two vectors A and B:

$$\mathrm{M} = \left(a' \mid b'\right) \tag{3.13}$$

While the prime make means:

$$\begin{aligned} a' &= \tfrac{a}{\sigma_a} \\ \sigma_a^2 &= (a \mid a) \\ b' &= \tfrac{b}{\sigma_b} \\ \sigma_b^2 &= (b \mid b) \end{aligned} \tag{3.14}$$

The overlap M is going to normalize the signal-to-noise ratio(SNR).

$$\mathrm{SNR} = \frac{1}{\sigma_h}(s \mid h) \tag{3.15}$$

SNR value is proportional to the amplitude of GW signal. In time domain, the SNR given by matched filter is:

$$(s(t) \mid h) = \int_{-\infty}^{\infty} \tilde{s}^* \tilde{h} e^{-2\pi i f t} df \tag{3.16}$$

Only when the measured signal gives an SNR higher than an SNR threshold, the signal is expected to contain a GW. Such signal's SNR is called a 'Trigger.'

3.1.3 Workflow of Matched Filtering Search

Section 3.1.1 and 3.1.2 talked about the algorithm of matched filtering. LIGO's CBC event searching tool [15] based on Matched Filtering can be described as Figure 3.1:

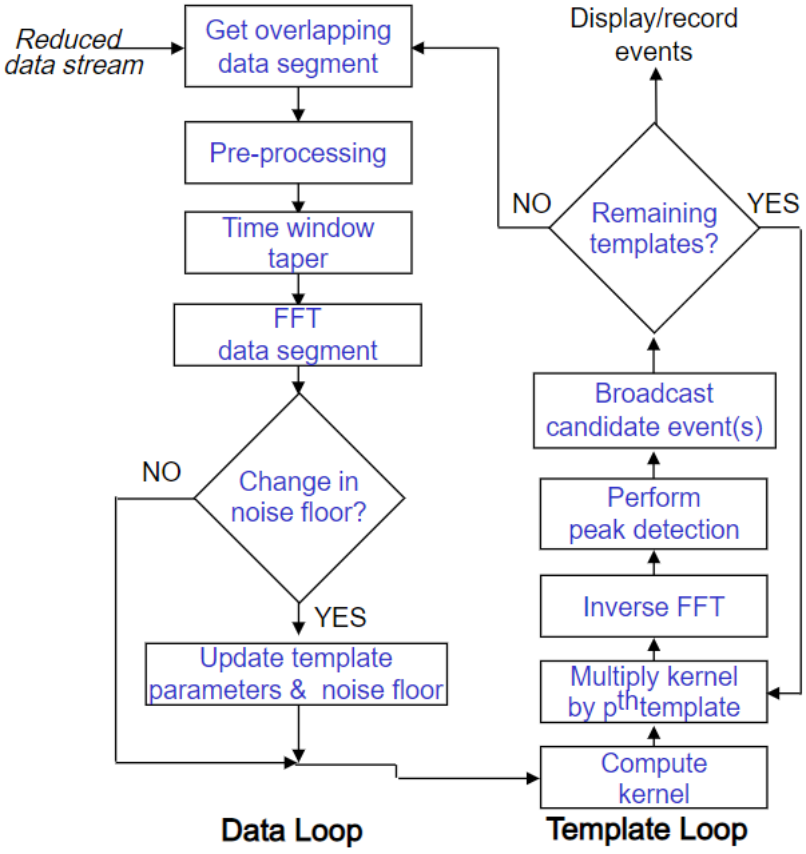

FIGURE 3.1: Data workflow of LIGO for the CBC searching. Representing a signal detector data analysis workflow.

The data stream from detectors go though the Data loop and the Template loop to find the optimal template and then record the event. Due to the huge computational cost of this algorithm, real-time detection and localization of GW are almost impossible. For example, GW170814 is firstly identified by two matched filter pipelines with high confidence about 30 seconds after its arrival[16]. Then the trigger was generated

to alert the observation partners. To observe the EM follow-up, low-latency search tools are necessary. In Chapter 5 and Chapter 6, we will present a new GW data analysis algorithm that can effectively reduce the computing cost.

3.2 SNR and Distance

We talk about the relation between SNR and the GW source distance. Combine 3.16 and 2.59, we can see that the SNR is inversely proportional to the distance between the observation point and the source, but not the square of the distance. To characterize the detector sensitivity, we usually use the range, which is called the detection range. It means the source at this range can be detected with an SNR of 8. Particularly, we often cite the distance of a binary neutron star(BNS) who have the masses of $1.4M_{\odot}$, if the SNR of this BNS event is 8.

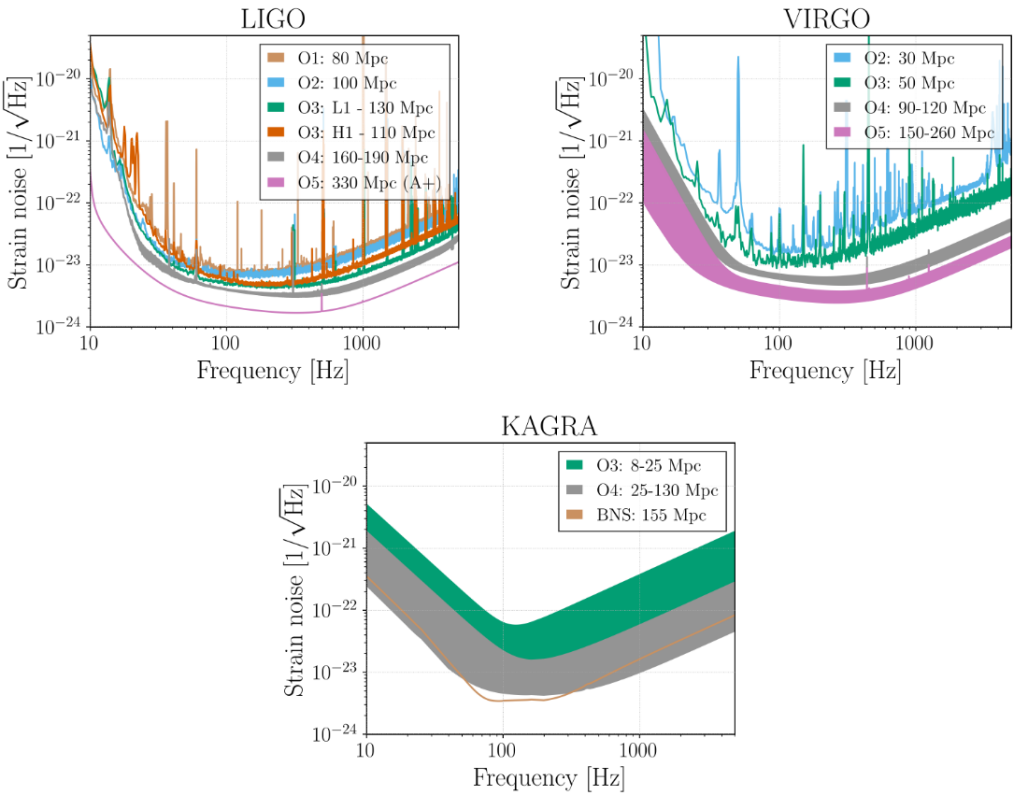

FIGURE 3.2: aLIGO(top left),AdV(top right) and KAGRA (bottom) target strain sensitivities as a function of frequency.

In Figure 3.2[2], the detection range is for a $1.4M_{\odot}$+$1.4M_{\odot}$ BNS merger. The BNS range (in megaparsec) achieved in past observation runs and anticipated for future runs is shown. The O1 LIGO curve is taken from the Hanford detector, the O2 LIGO curve comes from Livingston. In each case, these had better performance for that observing run. The O3 curves for aLIGO and AdV reflect the recent performance of the detectors.

3.3 Antenna Patterns

In this section, we consider the observatories' angular dependence on the GW source. We assume the location of the GW source on the skymap is (θ, ϕ), and the resource is rotated by ψ, which is called polarization angel. The angles can be seen in Figure 3.3. In this figure, two arms of the interferometer are along with x and y axis.

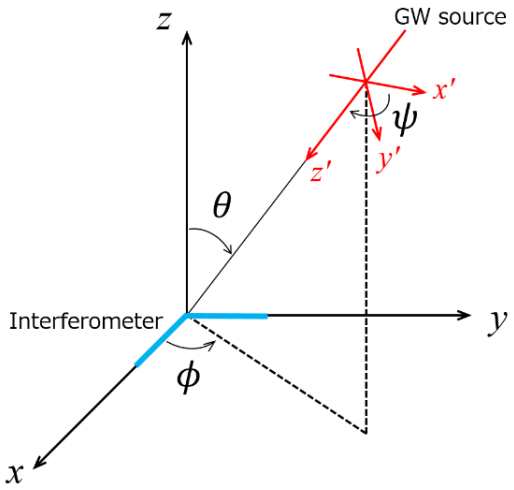

FIGURE 3.3: Defination of the angles describing the location of GW source and detector.

The amplitude of the signal detected by the observotory would be:

$$h(t) = F_+(\theta, \phi, \psi)h_+(t) + F_\times(\theta, \phi, \psi)h_\times(t) \tag{3.17}$$

where the $h_\times$ and h_+ are two polarizations of GW signal. The factors $F_\times$ and F_+ are antenna pattern functions. This equation tells how do the observotories respond to the

[2]You can find the file at https://dcc.ligo.org/public/0161/P1900218/002/SummaryForObservers.pdf

GW signal's angular characteristic. The two antenna pattern functions are defined as:

$$F_+(\theta,\phi,\psi) = \frac{1}{2}\left(1+\cos^2\theta\right)\cos 2\phi\cos 2\psi - \cos\theta\sin 2\phi\sin 2\psi \tag{3.18}$$

$$F_\times(\theta,\phi,\psi) = \frac{1}{2}\left(1+\cos^2\theta\right)\cos 2\phi\sin 2\psi + \cos\theta\sin 2\phi\cos 2\psi \tag{3.19}$$

3.4 Source Localization

The localization of the GW source needs the combination of separated GW detectors. As GW travels in the speed of light, the arriving time of GW recognized by different detectors are varied, depending on the direction of GW and detectors' location. A time lag between two detectors can be observed if they are not symmetric to the GW direction.

Assume using two detectors to localize the GW source. Suppose that the GW source's location is R on a unit sphere, and D is the light second distance between two detectors. The time lag between the arriving time measured by two detectors is given by $(T_1 - T_2) = D \cdot R$.

Assume the two detectors have time accuracies σ_1 and σ_2, the measured time are t_1 and t_2, the distribution of the reconstructed location $\boldsymbol{r}$ and prior distribution $p(\boldsymbol{r})$ are defined by:

$$p(\boldsymbol{r}|\boldsymbol{R}) \propto p(\boldsymbol{r})\exp\left[-\frac{(D\cdot(\boldsymbol{r}-\boldsymbol{R}))^2}{2\left(\sigma_1^2+\sigma_2^2\right)}\right] \tag{3.20}$$

To summarize, the time lag of the detectors is given by $D \cdot R$, and the time accuracy is $\sigma \sim \left(2\pi\rho\sigma_f\right)^{-1}$, where ρ represents SNR, and σ_f is the bandwidth of the signal calculated by

$$\overline{f^n} = 4\int_0^\infty \mathrm{d}f \frac{|\tilde{h}(f)|^2}{S(f)} f^n \tag{3.21}$$

In a three-detector-network occasion, the distribution function is transformed like[17]:

$$p(\boldsymbol{r}|\boldsymbol{R}) \propto p(\boldsymbol{r})\exp\left[-\frac{1}{2}(\boldsymbol{r}-\boldsymbol{R})^T\boldsymbol{M}(\boldsymbol{r}-\boldsymbol{R})\right] \tag{3.22}$$

where the matrix M represents the accuracy:

$$M = \frac{D_{12}D_{12}^T}{\sigma_{12}^2} + \frac{D_{23}D_{23}^T}{\sigma_{23}^2} + \frac{D_{31}D_{31}^T}{\sigma_{31}^2} \tag{3.23}$$

Figure 3.4 shows the localization result of GW170817. As we can see, a two-detector-

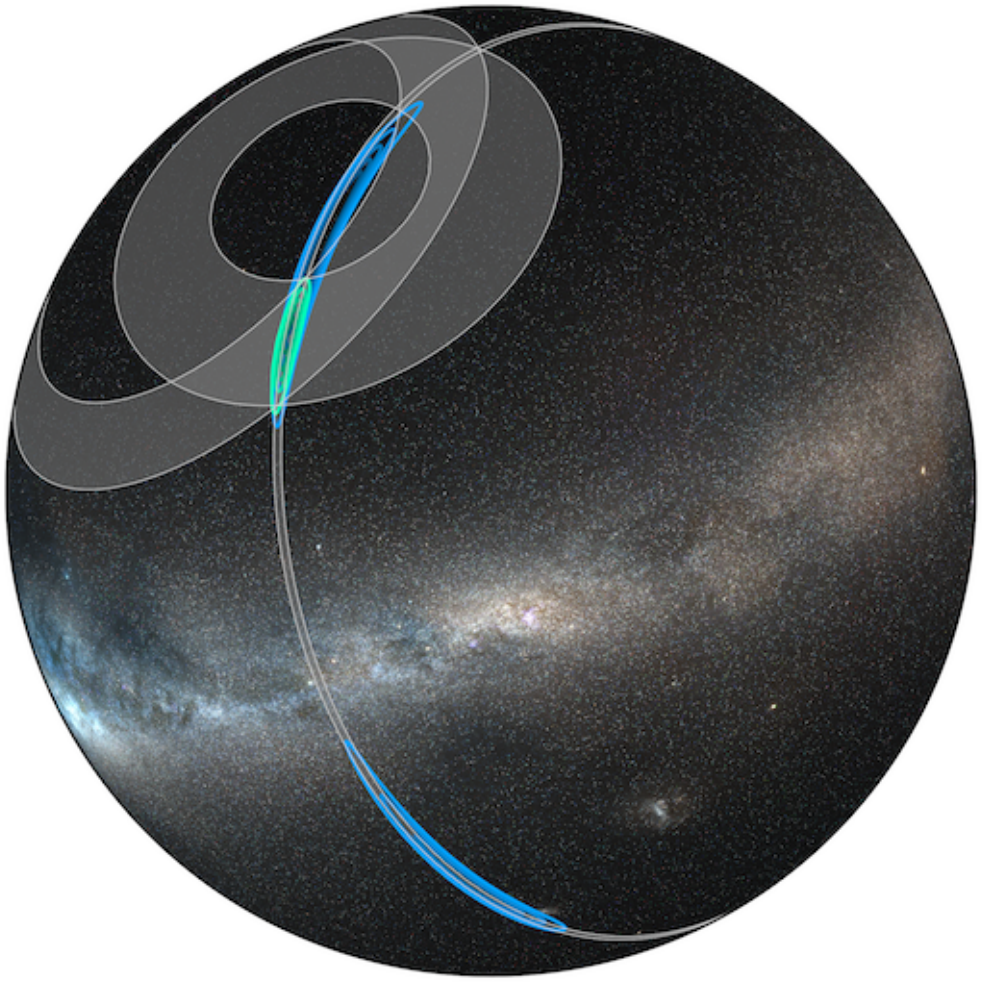

FIGURE 3.4: GW170817 Localization and Triangulation Annuli. The rapid Hanford-Livingston localization is shown in blue, and the final Hanford-Livingston-Virgo localization is in green. The gray rings are one-sigma triangulation constraints from the three detector pairs. [Credit: LIGO/Virgo/NASA/Leo Singer (Milky Way image: Axel Mellinger)]

network localization(blue) can only give a ring shape result. But a three-detector-network can localize the GW source with higher accuracy(green). To conclude, a high precise GW source localization needs a three-detector network at least. As the GW detector in Japan, KAGRA has finished construction and started running, it's very attractive to see if a four-detector network has better performance in the study of localization.

3.5 Analysis Algorithms

LIGO and Virgo have developed several localization methods based on matched filtering. The algorithms are called pipelines. The candidate events are uploaded to GraceDb.

3.5.1 Compact binary search

Here we introduce several pipelines that are used by LIGO and Virgo. The pipelines are aiming at the detection of compact binary coalescence(CBC) events.

PyCBC search

PyCBC[14, 18, 19, 20] is an open-source toolkit developed by GW data analysis researchers. LIGO and Virgo used PyCBC to calculate the SNR of GW events. PyCBC is based on matched filtering algorithm, more details can be found in the document of PyCBC.[3]

GstLAL pipeline

This is also a matched filtering based algorithm used for localization research, developed by LIGO and Virgo[21]. More details can be found in[22].

cWB pipeline

Coherent WaveBurst (cWB)[23] is a data-analysis tool to search for a broad range of gravitational-wave (GW) transients. The pipeline identifies coincident events in the

[3]http://pycbc.org/pycbc/latest/html/

GW data from earth-based interferometric detectors and reconstructs the gravitational wave signal by using a constrained maximum likelihood approach. coherent WaveBurst was the first algorithm to identify the gravitational wave signal GW150914 detected by LIGO.

MBTA pipeline

MBTA[24] constructs its background by making every possible coincidence from a single detector triggers over a few hours of recent data. It then folds in the probability of a pair of triggers passing the time coincidence test.

SPIIR pipeline

SPIIR[25] applies summed parallel infinite impulse response (IIR) filters to approximate matched-filtering results. It selects high-SNR events from each detector and finds coherent responses from other detectors. It constructs a background statistical distribution by time-shifting detector data one hundred times over a week to evaluate foreground candidate significance.

3.5.2 Sky position reconstruction

Full parameter estimation

For reconstructing the GW source's precise position, we use Bayesian inference to estimate all the unknown parameters of the GW signal. LALInference is an algorithm based on Bayesian inference that can estimate the GW parameters. However, this full parameter estimation costs a lot of time. Usually, the GW signal analysis contains 15 parameters' calculation and lasts hours to days to complete.

Bayestar

Bayestar algorithm is used for fast GW position reconstruction. Following Bayesian inference, Batestar calculates a likelihood function using a three-detector network's output, such as arrival time, amplitudes, and phases. It can calculate localization results in several minutes. Comparing with full parameter estimation, the Bayestar is faster but can localize the GW source with similar accuracy.

3.6 Multi-messenger astronomy

As a rapid localization of the GW source can be accomplished, the observation of one GW source with EM observatories together is possible. Not only BBH and BNS, but we can also understand EM-emitting supernovas and pulsars more comprehensively. Multi-messenger astronomy will be more important in the field of astronomy in the future. For example, to observe a kilonova event derived from a BNS merger event, the BlackGEM telescope will cost a few minutes to align to an area of 2.7deg^2. A monitoring range smaller than 300deg^2 is necessary for the EM observation. On the one hand, Since the EM signal decays in a few days, a quick response to the merger event will benefit to EM observatories alignment. On the other hand, a multi-detector-network GW localization can localize the GW source in a smaller range, to help EM detectors maintain more information.

The expected localization accuracy is about 100deg^2 if an advanced detector becomes operational this year. The Figure 3.5 shows the summary of localization accuracy using two-detector-network and three-detector-network[4] :

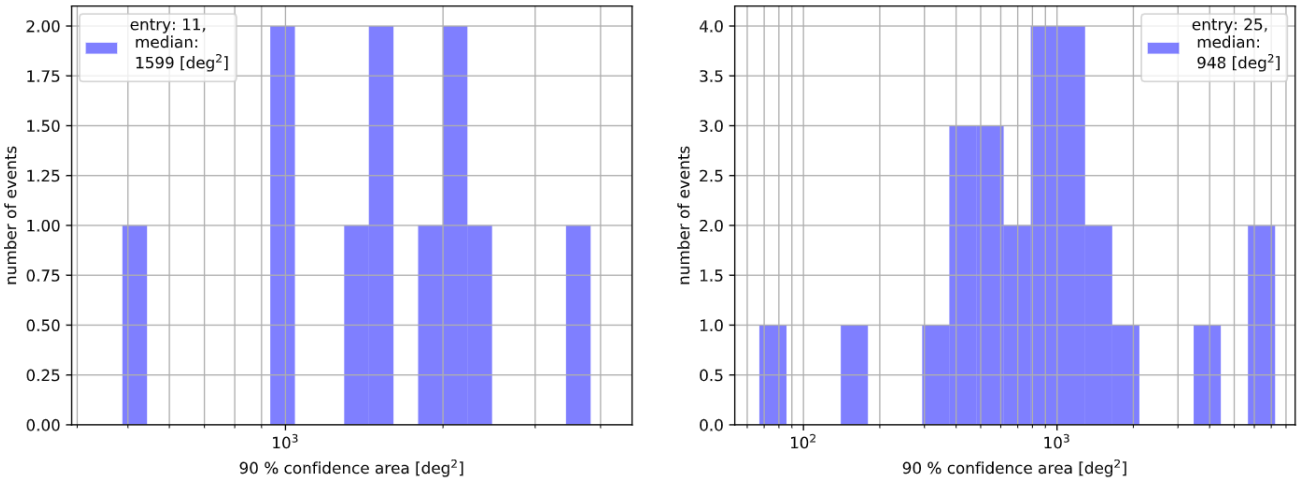

FIGURE 3.5: The size of 90% confidence area which are alerted in the first half of O3, which is called O3a. Events detected by two LIGO detectors are shown in the left and the events detected by LIGO-Virgo three-detector-network are shown in the right

As we can see, the median value of confidence from LIGO two detector's observing data is 1600deg^2, and is 950deg^2 to a three-detector-network. As the sensitivity

[4]Cited from https://summary.ligo.org/detchar/summary/O3a/

situation during the measurement, the detection ranges of Advanced LIGO Hanford, Advanced LIGO Livingston, and Advanced Virgo are 120Mpc, 140Mpc, and 45Mpc, respectively. The quoted ranges are the detection ranges for a BNS merger event.

Hopefully, the fourth GW detector KAGRA will join in the localization network in the future. If all the four detectors can reach their design sensitivity, the localization accuracy would be improved to 10deg^2[26]. However, the actual sensitivities are far lower than we expected. How to use the low-sensitivity GW detectors to localize the GW source is still under learning.

Chapter 4

Convolutional Neural Network

In Chapter 3, we discussed the analysis of the GW signal using the Matched Filtering algorithm. But the matched filtering has shortages like slow reaction speed, and massive template bank necessity, which is significantly unsuitable to multi-messenger astronomy. In this chapter, we will discuss developing a Convolutional Neural Network(CNN) to analyze the GW signal instead. The problems met by Matched Filtering can be solved by CNN's fast analyze speed and strong generalization performance.

4.1 Perceptron

To understand the structure and principle of CNN, we have to figure out its basic unit, the artificial neuron. The theory and principle of an artificial neuron are proposed by Warren McCulloch and Walter Pitts in 1943[27]. In 1957, Frank Rosenblatt developed this idea and designed an algorithm called Perceptron, which can run on a computer firstly[28]. Perceptron solves linear classification problem, and a handwritten letters recognition experiment succeed in 1960. From then on, Machine learning has ushered in a research boom.

Figure 4.1 is the structure of a perceptron: In Figure 4.1, the x_1 and x_2 are two input numbers, b is called bias, w_1 and w_2 are called weights. This perceptron take two inputs number(x_1, x_2) in, and output one number. The output y, has a value of 0 or 1, depending on the input value, bias, and weight. The perceptron can be expressed as the function below:

$$y = \begin{cases} 1 & \text{if } \sum_i x_i w_i + b > 0 \\ 0 & \text{if } \sum_i x_i w_i + b \leq 0 \end{cases} \tag{4.1}$$

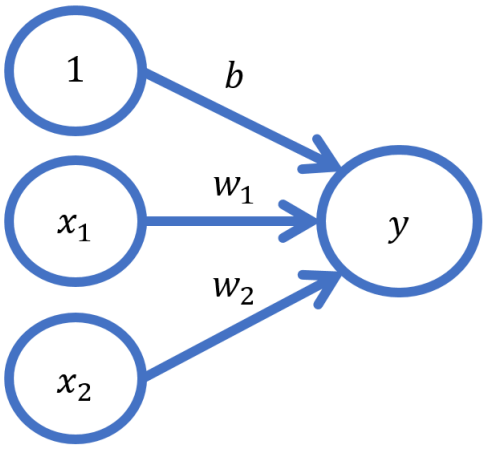

FIGURE 4.1: Percetron diagram

That is to say, this perceptron looks like a gate. When the input number is small, the gate would not open for it, the output value is 0. But if the input has a big value knocking the door, the gate would open, thus the output is 1.

4.2 Liner Classification Using Single Perceptron

As we talked before, a perceptron can solve a liner classification problem. To describe the work simply and clearly, here we compare the perceptron with a AND logic gate. Perceptron and logic gate both have two inputs and one output, and they can both solve a liner classification problem. The truth table of AND gate is shown in Table 4.1. We can see only when $x_1 = x_2 = 1, y = 1$. Otherwise, the output should be 0.

TABLE 4.1: AND gate truth table

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

Next, we try to use a perceptron to accomplish the same classification. Here we adopt the following parameter setup: $w_1 = w_2 = 0.5, b = -0.75$. The parameter set

is not unique, countless parameter sets satisfy our demand. To simplify the equation, such setups were used.

Thus, 4.1 has been transformed as the following equation:

$$y = \begin{cases} 1 & \text{if } \sum_i 0.5x_i - 0.75 > 0 \\ 0 & \text{if } \sum_i 0.5x_i - 0.75 \leq 0 \end{cases} \tag{4.2}$$

Now we draw a picture consists of $(0,0), (0,1), (1,0), (1,1)$ four points and line $y = 0$ in 4.2.

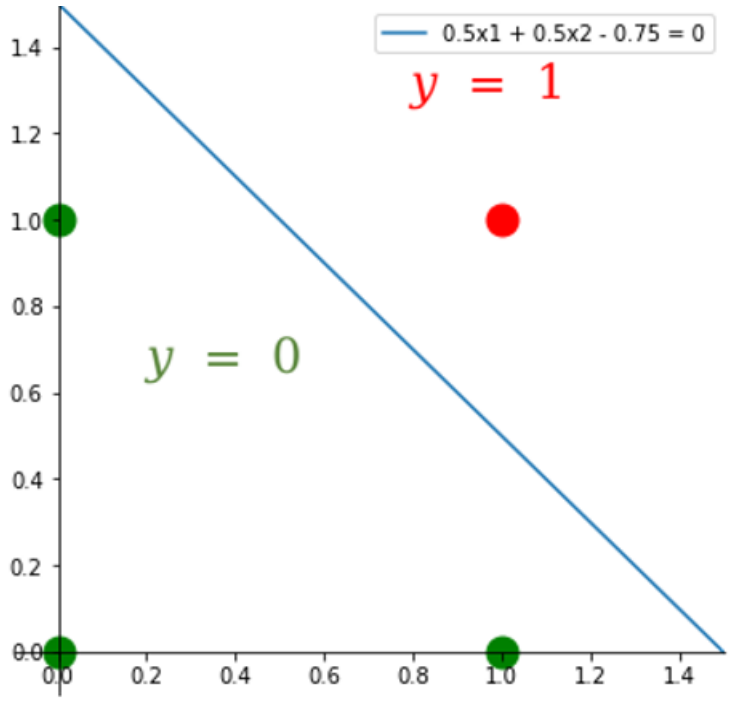

FIGURE 4.2: Perceptron achieves four points classification

In this picture, three blue points($y = 0$) and one red point ($y = 1$) are divided into two groups by the blue line. Any inputs above the line will cause the output to be $y = 1$, and the inputs below the line will lead to $y = 0$. In this way, we prove that a perceptron can achieve a liner classification, just likes AND gate does. Not only AND gate, by handling the parameters properly, but perceptrons can also have the same function with OR gate, NOT gate, and other liner logic gates.

4.3 Non-linear Classification Using a Multilayer Perceptron

So far, we proved the liner classification ability of perception. Now we talk about a non-linear case. Like what we did in the Section 4.2, we consider how to achieve an XOR gate using perceptrons. Its circuit diagram and truth table is shown in Figure 4.3 and Table 4.2. We can see only when the inputs are the same, $y = 1$. Otherwise, $y = 0$.

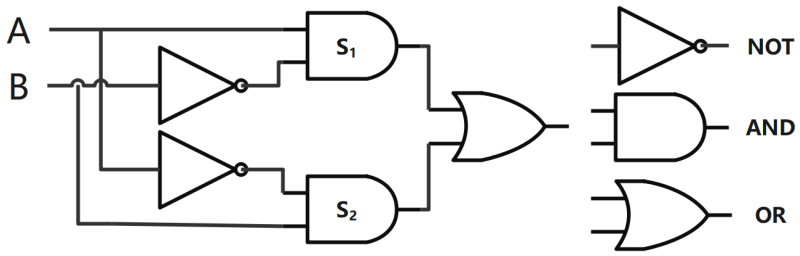

FIGURE 4.3: XOR gate

TABLE 4.2: XOR gate truth table

x_1	x_2	y
0	0	1
0	1	0
1	0	0
1	1	1

The Figure 4.4 shows the positions of four points. But as we can see, a straight line can not precisely divide the points into two groups according to their y value. None of parameter sets can curve the line to satisfy a non-liner classification demand. Thus, we have to consider adopting a multilayer perceptron with a activation function. Multiple layers and non-linear activation functions will bring nonlinearity to the perceptrons.

By applying one perceptron's output as the next layer perceptron's input, we can construct a multilayer perceptron. In this 4-point-classification case, a two-layer perceptron can meet our needs. As Figure 4.5 shows, the first layer (x_1, x_2) consists of single perceptrons, but the second layer (s_1, s_2) applied the activation function $f(x) = \arctan x$. By adjusting the weights and the biases of the perceptrons, the two-layer perceptron in Figure 4.5 can have the same function as an XOR gate.

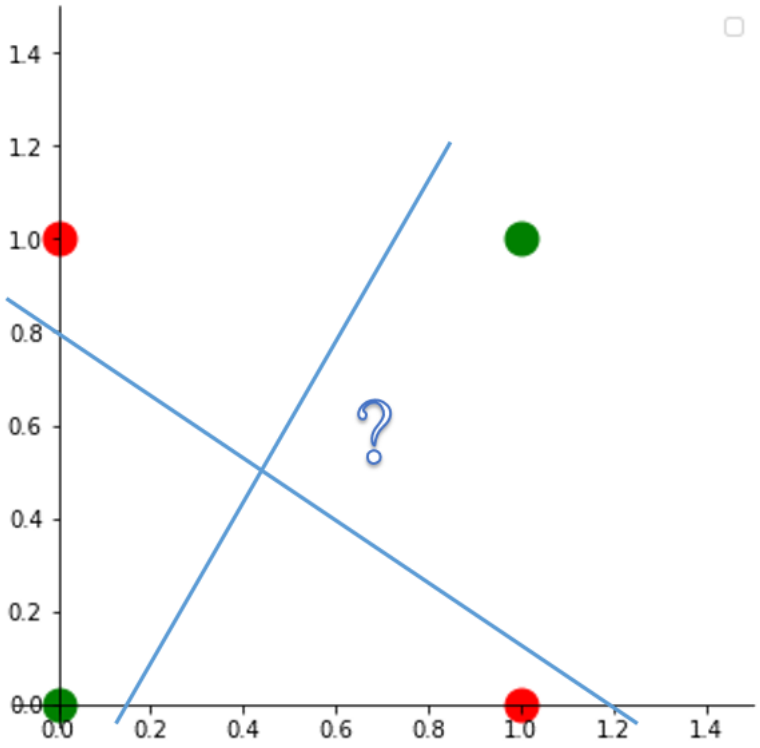

FIGURE 4.4: One line can not achieve non-liner classification

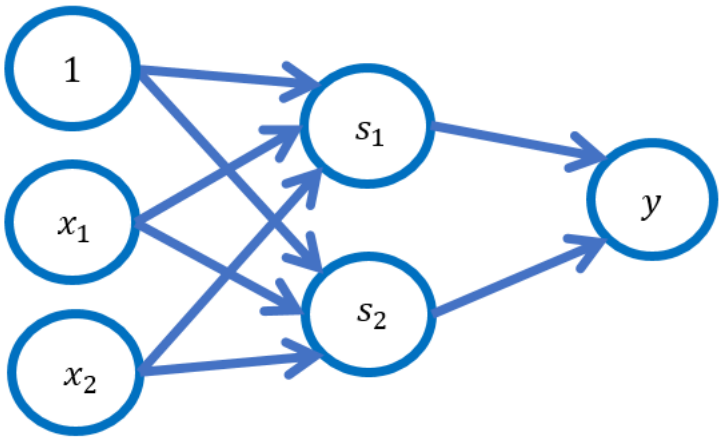

FIGURE 4.5: Use 2-layer-perctron to construct a XOR gate

Table 4.3 is truth table of Figure 4.5 and Figure 4.3.

So far, So a non-linear classification is accomplished by the two-layer perceptron. By adjusting the weights and biases of the perceptron, other non-linear classification problem can also be solved. The perceptrons having multilayers are named multi-layer perceptron(MLP), which is the most simple neural network – artificial neural

TABLE 4.3: Two-layer perceptron truth table

x_1	x_2	s_1	s_2	y
0	0	1	0	0
1	0	1	1	1
0	1	1	1	1
1	1	0	1	0

network(ANN). Researchers developed more complicated but more superior neural networks from ANN, such as the convolutional neural network(CNN), recurrent neural network(RNN), etc.

4.4 Components of Convolutional Neural Network

Convolutional Neural Network is a class of deep neural networks, which is commonly applied in image analyzing. An experiment of cat&dog image classification using CNN[29] gained an accuracy of over 83%. A typical CNN consists of pooling layers, fully connected layers, normalization layers, and at least one Convolutional layer. These layers are called hidden layers, the calculation process in these layers is usually invisible.

4.4.1 Convolutional Layer

The Convolutional layer is the core of the convolutional neural network. Unlike the other neural networks or multilayer perceptrons, CNN does not have fully connecting layers, that is, neurons in CNN only accept limited data but not all input from the last layer. The 'fully-connectedness' ensures that CNN is not disturbed by overfitting. This idea comes from image process machines in real-life – eyes. Neurons of the animal visual system are sensitive to the specific wavelengths, and the information from different perception area merged to process a full image.

The name of Convolutional layer comes from the convolution function, which can filter the input data with learnable kernels.

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix} * \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1n} \\ y_{21} & y_{22} & \cdots & y_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ y_{m1} & y_{m2} & \cdots & y_{mn} \end{bmatrix} = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} x_{(m-i)(n-j)} y_{(1+i)(1+j)} \tag{4.3}$$

In 4.3, the first matrix is input data, and the second matrix is the filter kernel. The equation gives a number as a convolution result. But convolution kernels are usually smaller than the input matrix so that the output is no longer a number but a matrix. The kernel will filter the target matrix from left the right and from top to bottom. Each process gives one outcome. Finally the output matrix is a combination of several convolution outcomes. For example, we calculate the convolution outcome of a 4*4 matrix and a 3*3 matrix:

$$\begin{bmatrix} 1 & 2 & 3 & 0 \\ 4 & 1 & 0 & 3 \\ 3 & 0 & 1 & 0 \\ 0 & 3 & 4 & 1 \end{bmatrix} * \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 7 & 6 \\ 8 & 11 \end{bmatrix} \tag{4.4}$$

The result matrix come from these four processes:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 1 & 0 \\ 3 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix} = 7 \tag{4.5}$$

$$\begin{bmatrix} 2 & 3 & 0 \\ 1 & 0 & 3 \\ 0 & 1 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix} = 6 \tag{4.6}$$

$$\begin{bmatrix} 4 & 1 & 0 \\ 3 & 0 & 1 \\ 0 & 3 & 4 \end{bmatrix} * \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix} = 8 \tag{4.7}$$

$$\begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 0 \\ 3 & 4 & 1 \end{bmatrix} * \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix} = 11 \tag{4.8}$$

In CNN, the filters play the role of weights in neural networks. The same with MLP, convolutional layer also has biases, which act on the every elements in matrices, like 4.9 shows.

$$\begin{bmatrix} 7 & 6 \\ 8 & 11 \end{bmatrix} + \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix} = \begin{bmatrix} 9 & 8 \\ 10 & 13 \end{bmatrix} \tag{4.9}$$

In this thesis, we used Tensorflow to construct the CNNs. The Convolutional layer function of Tensorflow has four parameters: filters, kernel size, strides, and padding. Now we give a brief introduction to the parameters.

Filters

Filters of Convolutional layer mean the number of the filters or the dimensionality of the output space in term. The filters parameter in 4.4 is one, so the matrix size is $[4 * 4 * 1]$. If we let the filters parameter be 3, we will have a $[4 * 4 * 3]$ size output matrix. In the field of image classification, filters are usually set to 3 to detect the information in the RGB tri-color field.

Kernel size

Kernel size specifying the height and width of the filter matrix. It is an integer list contains two numbers. In 4.4 the kernel size is $(3, 3)$.

Stride

An integer list contains 2 integers, specifying the stride of the convolution along with the height and width. In 4.4 the stride is $(1, 1)$, which means the filter slides over the matrix from left to right, from top to bottom, with a step of 1. If the stride is 2 or more, the filter will skip 1 or more elements every time it slides. A large stride parameter will produce a smaller output but loss some information.

Padding

Padding controls the size of the output matrix, by adding columns and lines with 0 to the matrix. It is usually used to even the size of input and output matrices. Padding's value is 'True' when activated.

$$\begin{bmatrix} 1 & 2 & 3 & 0 \\ 4 & 1 & 0 & 3 \\ 3 & 0 & 1 & 0 \\ 0 & 3 & 4 & 1 \end{bmatrix} * \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 7 & 6 \\ 8 & 11 \end{bmatrix} \overset{\text{Padding}}{\rightarrow} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 7 & 6 & 0 \\ 0 & 8 & 11 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.10}$$

4.4.2 Pooling layer

In a CNN structure, pooling layers are inserted between Convolutional layers to reduce the size of output matrices from the previous layers. Thus the data amount and parameters are controlled, computing costs are also efficiently saved. For example, a $(2*2)$ pooling layer can divide the input matrix into several $(2*2)$ small matrices, and extract the maximum values from each matrix.

$$\begin{bmatrix} 4 & 11 & 2 & 1 \\ 5 & 3 & 2 & 2 \\ 9 & 11 & 4 & 0 \\ 4 & 13 & 1 & 2 \end{bmatrix} \overset{\text{Pooling}}{\rightarrow} \begin{bmatrix} 11 & 2 \\ 13 & 4 \end{bmatrix} \tag{4.11}$$

As 4.11 shows, as we activated the pooling function, the data was decreased by 75%. Pooling size larger than $(2, 2)$ is too destructive, which usually causes information loss. Pooling size of $(2, 2)$ is more commonly used.

4.4.3 Fully Connected Layer

Dense layer, like its name, is a kind of fully connected layer that is connecting every neuron from the previous layer. The Dense layer collects all the impulse from the previous layer and outputs N values. Usually, the Dense layer is the final layer in CNN architecture, so that the N numbers given by Dense layer corresponds N possibilities of N classes. For example, in the dog & cat classification, Dense layer was used to give

two possibilities of dog & cat. The large the difference between these two numbers, the better the classification effect.

4.5 Architecture and Workflow of Convolutional Neural Network

4.5.1 CNN Architecture

The Section 4.4 introduced the layers of CNN. Their specific combination and sequence are depending on our needs. Figure 4.6 shows the CNN architecture for binary classification. The architecture can be simplified into three components: Input later, Hidden layer, and output layer. The input layer is in charge of transforming data into operational matrices, then send them to the hidden layer. Hidden layer, which consists of Convolutional layers, Polling layers, Fully connected layers can process the data, summarize the laws among the data, and pass the information to the Output layer. Output layer gives a list of possibilities corresponding to input data classes. In a binary classification case, the output layer gives two possibilities.

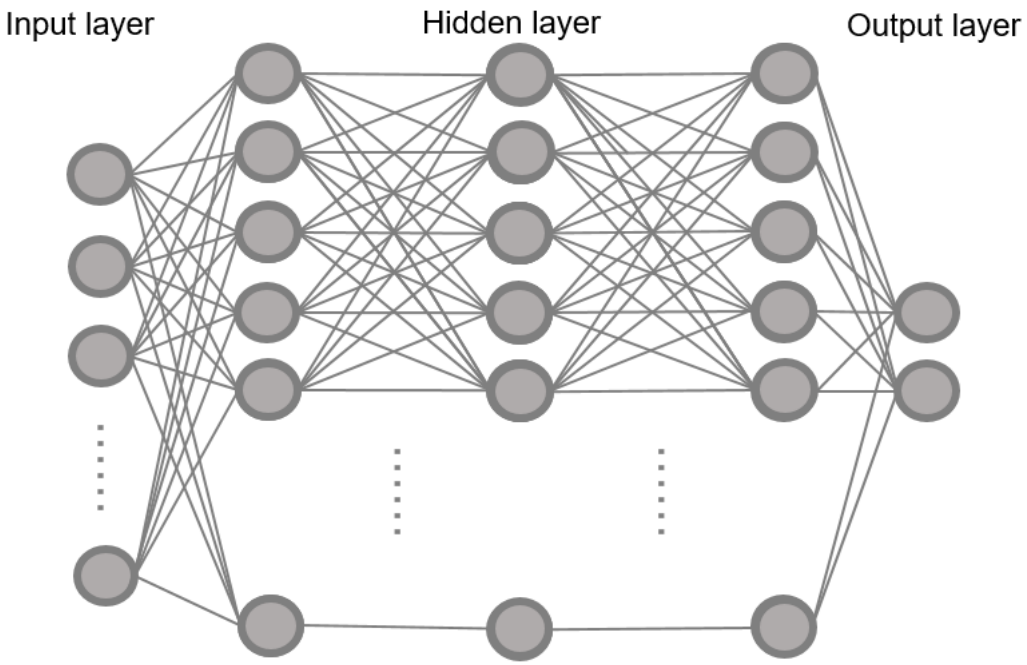

FIGURE 4.6: CNN architecture for binary classification

4.5.2 CNN Workflow

This section talks about the classification workflow using CNN. Classification aims to classify mixed data sets into several categories based on their characteristics; the characteristics of input data is called features in the term. Because parameters in CNN are not at its best from the beginning, we should let the CNN optimize its parameters by learning input data features. This process is called 'Train'. The parameter optimization method is discussed in the next section. Especially, if we attach every input data with a label, the training process is named 'Supervised learning' that significantly benefits the learning efficiency.

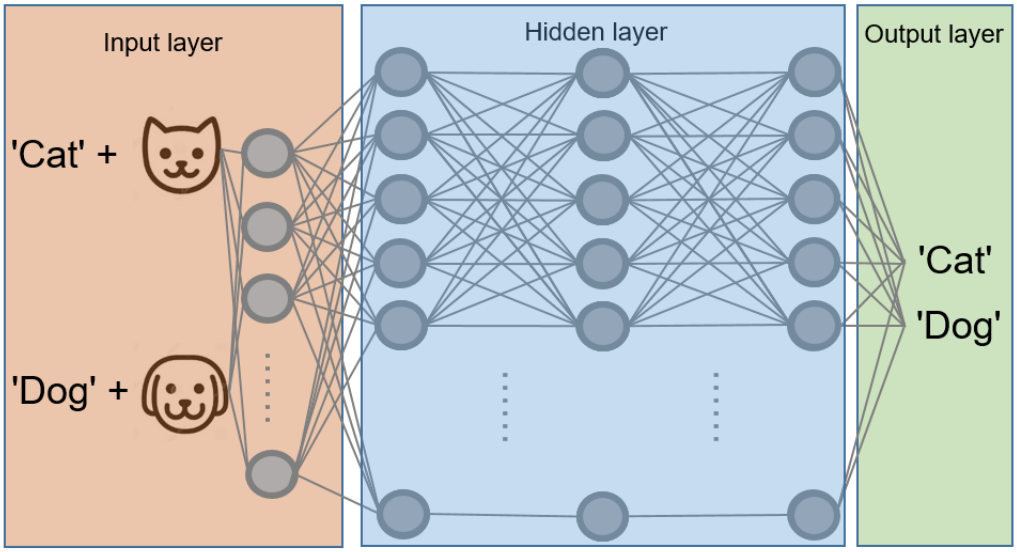

FIGURE 4.7: Cat & Dog classification – Supervised Learning

If no labels are attached, which means the training process is applying an 'Unsupervised learning', the CNN will easily mismatch the features and the train data sets. For example, in the Cat & Dog classification problem, an unsupervised learned CNN would divide the input data into 'small animal' and 'big animal' two groups, but not classify them according to their species. Unsupervised learning is usually applied to the big-data analysis, such as background gravitational waves analysis.

After all the parameters are tuned to the minimum error state, CNN is ready for a 'Test', which means classify the unknown input data. Usually, the training data set and the test data set are generated to be different, for measuring CNN's generalization ability. That is to say, a well- trained CNN should be able to handle unknown test data.

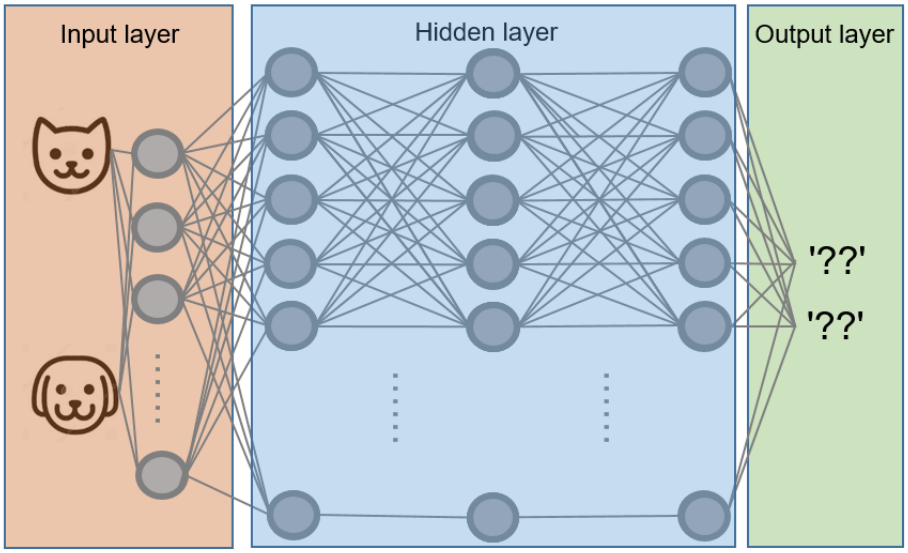

FIGURE 4.8: Cat & Dog classification – Unsupervised Learning

We can also recognize the test process as 'Label predict', because the actual output data correspond to putative labels. How to measure the prediction and classification capabilities of CNN depends on the accuracy of the test. In the Cat & Dog classification, we define the accuracy as the Equation 4.12:

$$\text{Accuracy} = \frac{\text{TC} + \text{TD}}{\text{TC} + \text{FD} + \text{FC} + \text{TD}} = \frac{\text{TC} + \text{TD}}{\text{N}} \tag{4.12}$$

Where the codes in the equation are the amounts of predicted labels:

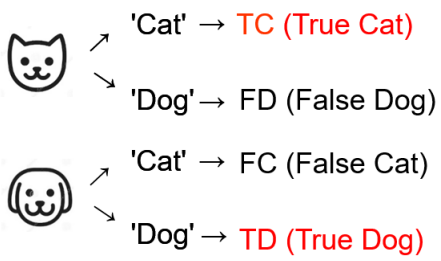

FIGURE 4.9: Predicted labels of the Cat & Dog classification

To maintain a high classification accuracy, we need to optimize the parameters inside the CNN via multiple training. The next section talks about the optimization algorithm.

4.5.3 Parameter Optimization – Backpropagation

In the study of 4-point-classification, the weights and biases of the perceptrons are assigned by ourselves. But for a CNN containing millions of parameters, setting every weights and biases manually is impossible. How to tune the parameters to the best depends on the backpropagation algorithm. The schematic diagram of backpropagation is Figure 4.10.

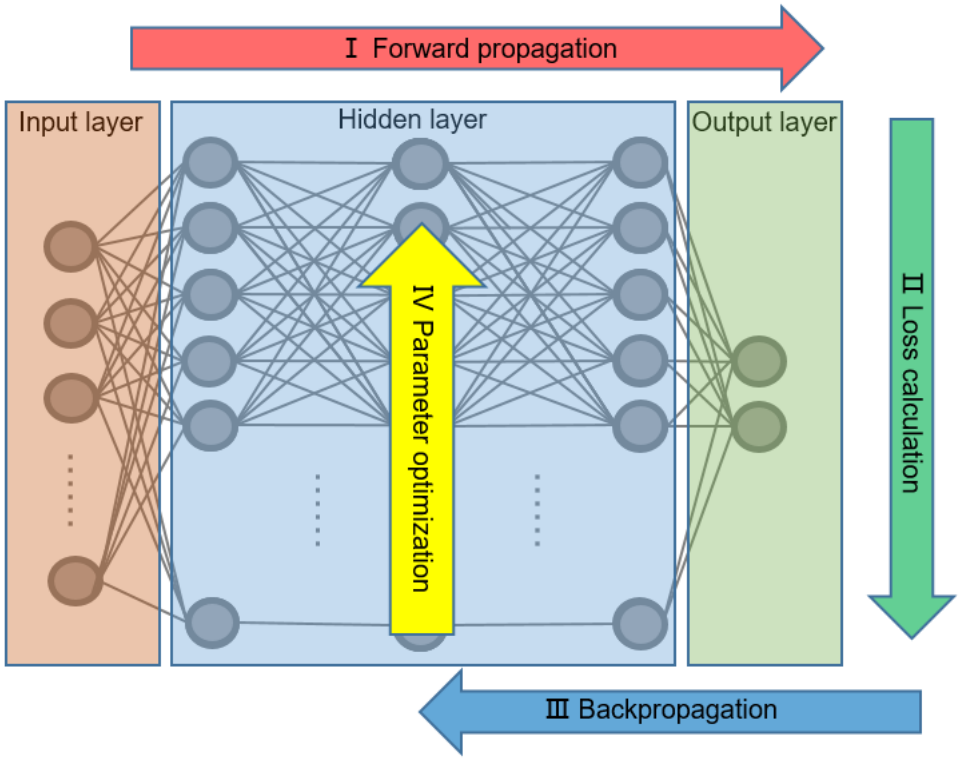

FIGURE 4.10: Parameter optimization based on backpropagation

At first, the input data is forward propagated through CNN, then the loss function calculates the difference between the right label and predicted label. Then the loss gradient of each parameter(weights & biases) is sent back to CNN. Finally, the parameters inside CNN are optimized according to the gradients. This process is named the gradient descent algorithm. To minimize the loss, the gradient descent algorithm takes steps proportional to the negative slope of the loss function. To understand this algorithm, let us consider how to come down the mountain at the fastest speed.

Assume one man trapped in the mountains, and he is trying to get down from the mountain as soon as possible. The most efficient strategy is to always walk along

the path having a maximum negative gradient. The length of each step is called the learning rate. But this method also has a disadvantage: easy to be trapped in a small pit. The gradients around the pit bottom are always positive that invalid this strategy. That is to say, this man can find the local minimum point, but he likely cannot find the global minimum.

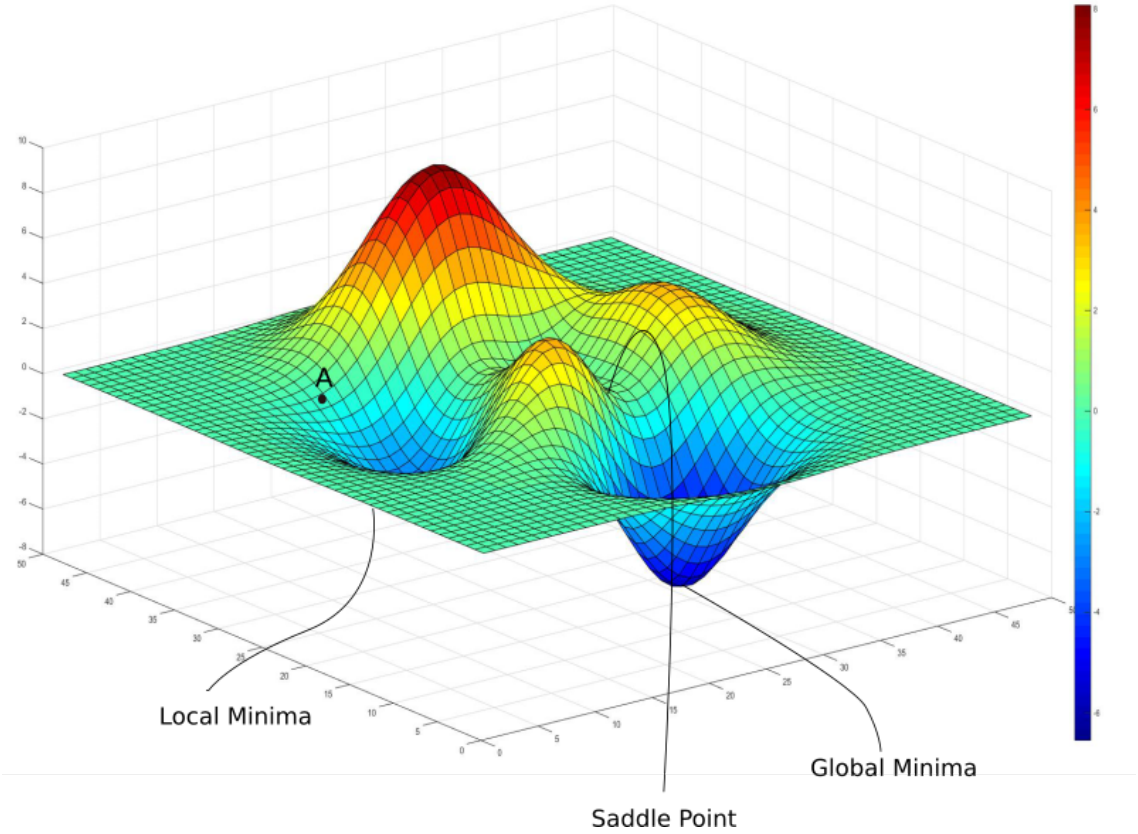

FIGURE 4.11: Gradient descent algorithm – Local minimum and Global Minimum

About how to avoid trapping in the local minimum, many researchers have proposed solutions. Such as mini-batch training, which means to train the CNN in several steps with several batches. Each train epoch gives one loss function, then finds the global minimum point from different loss maps. This method is called Batch Gradient Descent. Another way to find the global minimum is by revising the learning rate. The learning rate is one hyperparameter that we can set manually. Usually, it has a fixed value. But these years, researchers find that a variable learning rate is helpful to see the global minimum. At the beginning of training, we use a large learning rate to quickly

find the global minimum, then lower the learning rate in several steps. This strategy is called Learning Rate Decay.

4.6 Software and Development Environments

So far, we discussed every detail about the fundamental of CNN. Fortunately, we don't need to realize all the functions by writing the codes, but get the support from machine learning software. The machine learning software often used include Tensorflow[1] , Keras[2] , PyTorch[3] , Panda[4] , Chainer[5] , etc. To speed up our calculation on computers, we used CUDA[6] and cuDNN[7] GPU acceleration toolkit provided by NVIDIA. In this thesis, we mainly used Tensorflow and Keras based on Python2. Aa for the dataset generation, a gravitational wave data library, PyCBC[8] , is used for generating GW waveforms and noises. In this section, we will give introductions for the software and the development environment.

4.6.1 Softwares to Implement Convolutional Neural Network

Tensorflow

Tensorflow is an open-source framework for machine learning. It provides multiple development tools, libraries, and also a community for researchers. We used Tensorflow to construct the CNN architecture in this thesis. We also used train and test functions of Tensorflow to improve and examine our CNN.

[1] https://www.tensorflow.org/
[2] https://keras.io/
[3] https://pytorch.org/
[4] https://www.tensorflow.org/
[5] https://chainer.org/
[6] https://developer.nvidia.com/accelerated-computing-training/
[7] https://developer.nvidia.com/cudnn/
[8] https://pycbc.org/

Keras

Keras is also an open-source library for machine learning, supporting calling and applying Tensorflow faster. That is to say, it can help users build CNN more conveniently. Generally speaking, it is considered as an interface but not a framework.

GPU Acceleration Toolkit – CUDA & cuDNN

NVIDIA's CUDA provides a development environment for GPU-accelerated machine learning. We also applied NVIDIA's cuDNN library, which designed for deep neural networks. With the help of GPU acceleration, we saved our test time cost can into one second.

4.6.2 Development Environment

Hardware Environment

This research is completed on a computer with a Linux operating system.
CPU: Intel i7-6700HQ 2.60GHz
GPU: NVIDIA GeForce GTX 1070
Memory: 16GB 2133MHz

Software Environment

Here we show the versions of crucial packages we used. Please note that using a different version may cause an error.

Python: 2.8.0[9]
numpy: 1.16.1
PyCBC: 1.14.2
lalsuite: 6.62
tensorflow: 2.1.0
tensorflow-gpu: 1.13.1
CUDA: 10.1.0
cupy-cuda101: 6.5.0

[9]Python2 has come to an end of life, no further updating will be released. But as PyCBC only supports Python2, we had to use the old version of Python.

cuDNN: 7.6.4
Keras: 2.3.1

Chapter 5

Detection of Gravitational Waves Using Neural Network

This chapter talks about how did we construct a CNN and used it to detect the GW in a noisy background. Our goal is to enable CNN to detect the GW event as far as possible and maintain high accuracy at the same time.

5.1 Previous Research

This thesis is not the first one that is trying to apply CNN to the study of GW data analyzing. Few articles talked about the successful implementation of the GPU-accelerated CNN algorithm. Daniel George et al. presented a deep CNN framework for GW detection that enables real-time multi-messenger Astro-physics[30]. Daniel George adopted a coincident check for the multiple detectors to reduce the false-alarm rate to 0.003%. This GPU-implementation deep CNN model is believed to have a calculation speed that is 10200 times faster than Matched Filtering. But this article didn't talk about the classification and parameter estimation method of the signal, such as GW source localization.

Chayan Chatterjee et al. constructed a deep ANN model to localize the simulated GW signal using a three-detector network(HVL)[31]. This model can localize the GW150914 merger event in a 312 deg^2 with an accuracy of 90%. But this research also has a short-coming: waveform parameters were not taken into account. In other words, the ANN model can only be used for a fixed GW waveform analysis. We believe that this model has to be modified to consider more variables. So that it can realize real-time GW data analysis. At the same time, we believe adopting a four-detector

network(HLVK) would benefit localization research.

The improvement of the GW detection algorithm is expressed in this chapter, and the development of GW localization research that can realize real-time GW analysis is shown in Chapter 6.

5.2 Implement of GW Detection Using CNN

So far, we discussed the basic principles and benefits of using CNN for image recognition. The reasons and methods for applying this technology to GW's data analysis have not been explained.

GWs also contain features, especially in the Ringdown period. In the Ringdown period, the GW waveform has the highest amplitude, and the highest frequency that distinguishes it with noise, as Figure 5.1 shows. We believe that the CNN algorithm can also classify GW waveforms and noises with high accuracy.

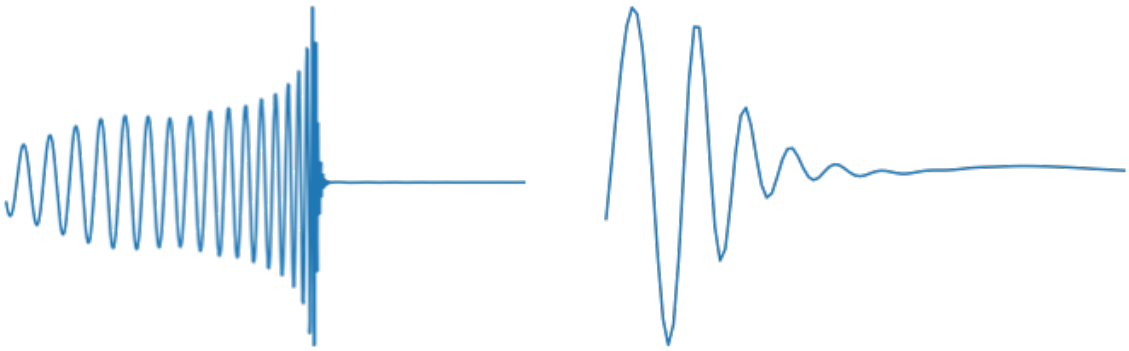

FIGURE 5.1: GW waveform(left) and its Ringdown period(right), generated by PyCBC

But, unlike the typical image classification process, in this research, we collected the time and amplitude information of the GW waveform as train and test data, but not digitizing the waveform according to its RGB value. The method of making the waveform diagram is shown in Figure 5.2. Also, all the pictures displayed in this thesis are all drawn after fitting, but this does not mean that we used the data from a fitted curve when making the data set. All amplitude and time information are accurate values provided by PyCBC.

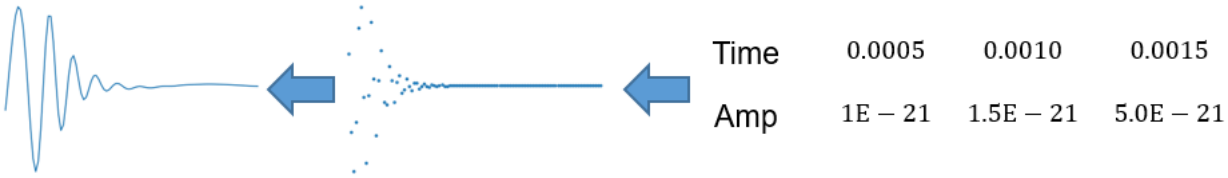

FIGURE 5.2: The waveform is derived from the time and amplitude data provided by PyCBC.

5.3 Architecture of the CNN for GW Detection and the Workflow

We used CNN as a binary classification algorithm to distinguish signal contains GW from pure noise. The workflow is similar to Figure 4.7, but the input data was replaced with the GW signal and noise signal, as Figure 5.3 shows.

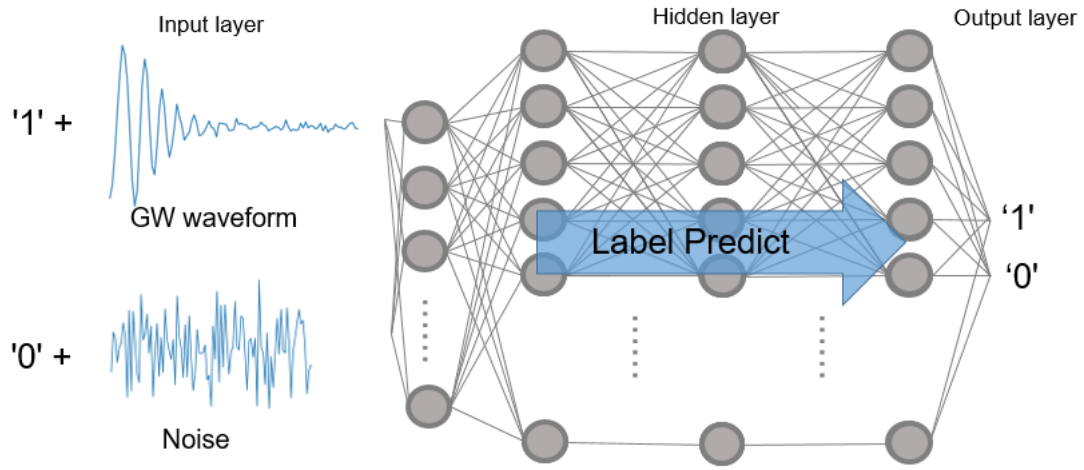

FIGURE 5.3: Workflow of the detection of gravitational waves

For the implementation of the coding, we used Keras for TensorFlow interface. The specific structure of the CNN is shown as Figure 5.4.

The input data contains noisy GW waveforms with label '1' and pure noise signals with label '0'. Three Convolutional layers are expected to learn the feature from input data. Pooling layers are inserted between the Convolutional layers, for saving computing capabilities. One Flatten layer that abandons data randomly can prevent the CNN from overfitting. The last layer is the Dense layer, it gives two output that corresponding to the possibilities of label '1' and label '0', the labels of input data.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 11, 2, 100)        22
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 11, 2, 100)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 11, 2, 22)         2222
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 11, 2, 22)         0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 11, 2, 44)         1012
_________________________________________________________________
dropout (Dropout)            (None, 11, 2, 44)         0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 11, 2, 44)         0
_________________________________________________________________
flatten (Flatten)            (None, 968)               0
_________________________________________________________________
dense (Dense)                (None, 2)                 1938
=================================================================
Total params: 5,194
Trainable params: 5,194
Non-trainable params: 0
_________________________________________________________________
```

FIGURE 5.4: CNN architecture for gravitational waves detection

Because the CNN needs to be trained several times to comprehend the laws inside the data, we need to generate an input data bank consisting of thousands of GW waveforms and noise signals. In order to make the generated waveforms as close as possible to the waveforms detected by the GW observatories, we used the GW data analysis software, PyCBC to make waveforms and noises.

5.4 PyCBC

PyCBC is an open-source software developed by LIGO and Virgo. It is used in the first direct detection of gravitational waves (GW150914) by LIGO and is used in the ongoing analysis of LIGO and Virgo data. Now PyCBC supports Windows system, Linux system, and macOS.

PyCBC has a vast database contains waveform simulation functions, PSDs of the detectors, GW events information, matched filtering algorithm, and localization pipelines. In this thesis, we mainly used it to generate GW waveforms and noises.

5.4.1 GW Waveforms Generation

The GW waveform generation function of PyCBC depends on these parameters: masses, spins, distance, approximant, tidal deformability, eccentricity, inclination, coalescence phase, sample time step, and the cutoff frequency for high-pass. The codes are shown below.

```
import matplotlib.pyplot as plt
from pycbc.waveform import get_td_waveform

hp, hc = get_td_waveform(approximant='IMRPhenomPv2',
    mass1=20,
    mass2=20,
    spin1z=0.5,
    spin2z=0.5,
    delta_t=1.0/4096, # Sample frequency = 4096Hz
    f_lower=40 # High pass filter
    )

plt.plot(hp.sample_times, # X cordinate: Sample time
    hp, # Y cordinate: Amplitude of plus polarization
    label='h1'
    )
plt.ylabel('Strain')
plt.xlabel('Time (s)')
plt.legend()
plt.show()
```

And the output of this code is Figure 5.5.

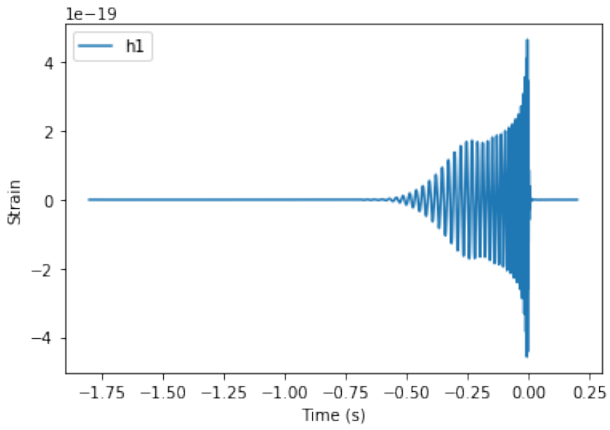

FIGURE 5.5: The waveform from a $20M_{\odot}$+$20M_{\odot}$ BBH merger

5.4.2 Noise Generation

We used PyCBC to simulate the noise signals in GW detectors. The noises should be the Gaussian noise and colored by the PSD of each detector. The code below is generating the noise in a LIGO detector.

```
import pycbc.noise
import pycbc.psd
import pylab
import numpy as np

# The color of the noise matches the PSD provided.
flow = 30.0
delta_f = 1.0 / 16
flen = int(2000 / delta_f) + 1
psd = pycbc.psd.aLIGOZeroDetHighPower(flen, delta_f, flow)

# Generate 32 seconds of noise at 4096 Hz
seed = np.random.randint(1024) # Generate a random number
delta_t = 1.0 / 4096
tsamples = int(32 / delta_t)
ts = pycbc.noise.noise_from_psd(tsamples, delta_t, psd, seed=seed)

pylab.plot(ts.sample_times, ts)
```

```
19 pylab.ylabel('Strain')
20 pylab.xlabel('Time (s)')
21 pylab.show()
```

We can see the noise is a function of duration, sample time step, PSD of the detector, and a random seed. Here we used the PSD of Advanced LIGO sensitivity. Because we also randomized the noise signal, the code gives a different picture every time it runs. One of the random noise signals is shown in Figure 5.6.

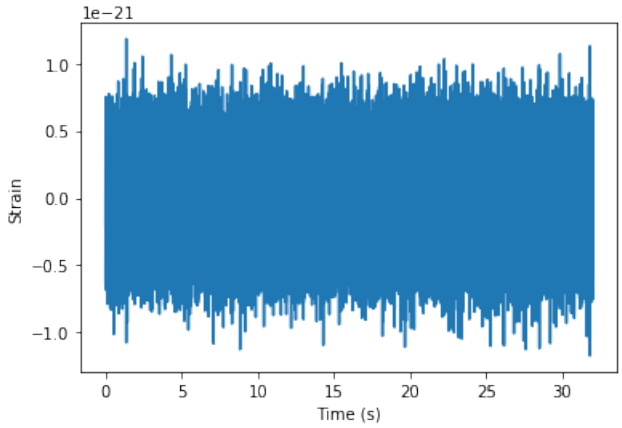

FIGURE 5.6: Simulated noise signal in Advanced LIGO

We combined the generated GW waveform with noise, to simulate the noisy waveform detected by the observatories, as Figure 5.7 shows.

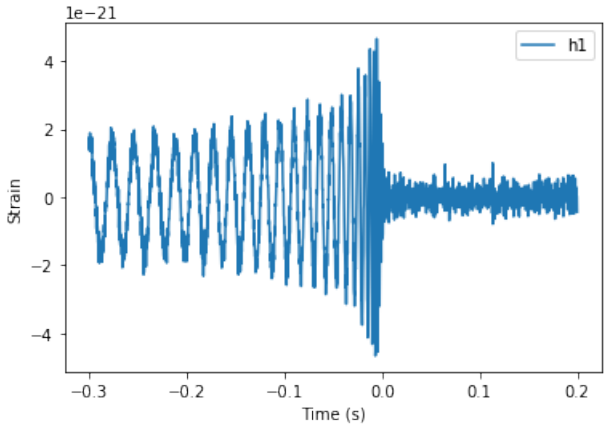

FIGURE 5.7: Simulated noisy waveform detected by Advanced LIGO

5.5 Train and Test Dataset Generation

5.5.1 Train Dataset

In order to train the generalization ability of the CNN, the train data bank should include as many types of waveforms as possible. This research mainly searched for BBH merger events. The specific parameter set is as follows:

- Masses of the binary range from $20M_{\odot}$ to $50M_{\odot}$, with an interval of $1M_{\odot}$.
- Spins of the binary range from 0 to 0.8, with an interval of 0.2.
- Distance of the binary range from 1Mpc to 700Mpc.
- Inclination, polarization, coalescence phase and the other parameters are fixed.

The parameter distribution of mass and spin is shown below.

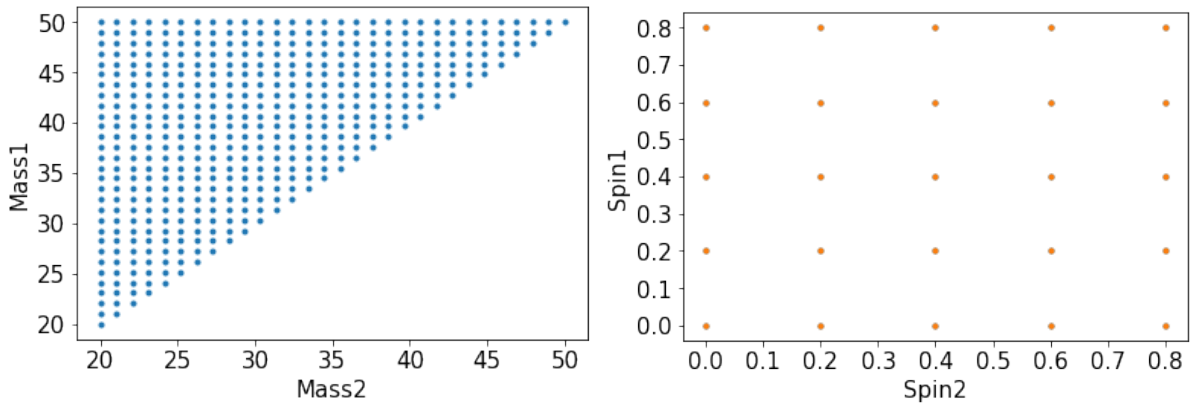

FIGURE 5.8: Parameters for train data generation

The generation code is shown below:

```
1 import pycbc
2 import numpy as np
3 import random
4 from pycbc.waveform import get_td_waveform
5 from pycbc.detector import Detector
6 import matplotlib.pyplot as  plt
7 import pycbc.noise
```

```
import pycbc.psd
apx = 'IMRPhenomPv2'
delta_f = 1.0 / 16
flen = int(2000 / delta_f) + 1
flow = 40
T = 1024
psd_h1 = pycbc.psd.aLIGOZeroDetHighPowerGWINC(flen, delta_f, flow)
mass1_range = np.linspace(20,50,30)
mass2_range = mass1_range
spin1z_range = np.linspace(0, 0.8,5)
spin2z_range = spin1z_range
inclination=2.34
pol_range=0.9
det_h1 = Detector('H1')
X = []
for i1 in range(len(mass1_range)):
    for i2 in range(len(mass2_range)):
     print((i1, i2))
     if mass1_range[i1]<=mass2_range[i2]:
        for i3 in range(len(spin1z_range)):
            for i4 in range(len(spin2z_range)):
                seed_noise = random.randint(1,127)
                delta_t = 1.0 / 2000
                tsamples = T
                noise_h1 = pycbc.noise.noise_from_psd(
                            tsamples, delta_t, psd_h1, seed=seed_noise)
                hp, hc = get_td_waveform(approximant=apx,
                                         distance=200,
                                         mass1=mass1_range[i1],
                                         mass2=mass2_range[i2],
                                         spin1z=spin1z_range[i3],
                                         spin2z=spin2z_range[i4],
                                         inclination=inclination,
                                         coa_phase=2.45,
                                         delta_t=1.0/2000,
                                         f_lower=40)
```

```
44                  signal_h1 = det_h1.project_wave(hp, hc,0,0,  0.9)
45                  signal_h1_array = np.array(signal_h1)[-T:]
46                  list_h1  = signal_h1_array. tolist ()
47                  h1_max_index = list_h1.index(max(list_h1))
48                  list_h1  = signal_h1_array. tolist ()
49                  X0 = []
50                  X1 = []
51                  X0.append([100*np.array(signal_h1.sample_times[h1_max_index-10:h1_max_index+11])
                        -100*signal_h1.sample_times[h1_max_index-10], 10**21*signal_h1_array[
                        h1_max_index-10:h1_max_index+11]])
52                  X1.append([100*np.array(signal_h1.sample_times[h1_max_index-10:h1_max_index+11])
                        -100*signal_h1.sample_times[h1_max_index-10], 10**21*noise_h1[h1_max_index-10:
                        h1_max_index+11]])
53                  X.append(X0)
54                  X.append(X1)
55  X = np.array(X)
56  np.save('X_4_27_1_train200mpc.npy', X)
```

Explanations for the lines in the block above:

- Line 1 to line 8: Import PyCBC and other packages.
- Line 9: Assign what approximant algorithm that PyCBC uses.
- Line 10 to line 21: Assign constant parameters.
- Line 22, 49, 50: Make empty lists to collect waveform information
- Line 23 to line 43: Generate waveforms.
- Line 44 to line 55: Post-process of the waveforms, collect time and amplitude information.
- Line 56: Save the information as a npy file.

In this way, we generated the template waveforms where are 200Mpc far away from the earth. Some of the codes are easily confused. Here we give some explanations to the codes.

1) GW waveforms in this template bank were not mixed with noises. The pure GW signals and noises were alternately inserted into the data set for training.

2) Not all information from the entire BBH merger is included in the data set. Based on the highest peak, 10 points on the left and 10 points on the right are selected. In other words, a search window having ten milliseconds duration is adopted.

3) As line 51 and line 52 show, the time values were enlarged by 100 times, while the amplitude values were enlarged by 10^{21} times. This action facilitates the CNN to analyze the data because the CNN is not very sensitive to small numbers.

Finally, 11,625 waveforms like Figure 5.5 were generated, and an equal amount of noise signal was inserted between these waveforms. So far, the train data set preparation is accomplished and is saved as a npy file.

5.5.2 Test Dataset

In principle, the train data and the test data must be separated. The method of making the test set is roughly the same as the method of making the train set, but the parameter set needs to be modified. Also, the GW waveforms should be mixed with noise to make noisy waveforms. The test data set parameters is as follows:

- Masses of the binary range from $20.5M_{\odot}$ to $50.5M_{\odot}$, with an interval of $1M_{\odot}$.
- Spins of the binary range from 0.1 to 0.9, with an interval of 0.2.
- Distance, Inclination, polarization, coalescence phase, and the other parameters were the same as the train data set.

The parameter distribution of the training set and test set is as follows:
The test data set is derived from these codes:

```
1 import pycbc
2 import numpy as np
3 import random
4 from pycbc.waveform import get_td_waveform
5 from pycbc.detector import Detector
6 from pycbc.waveform import get_fd_waveform
```

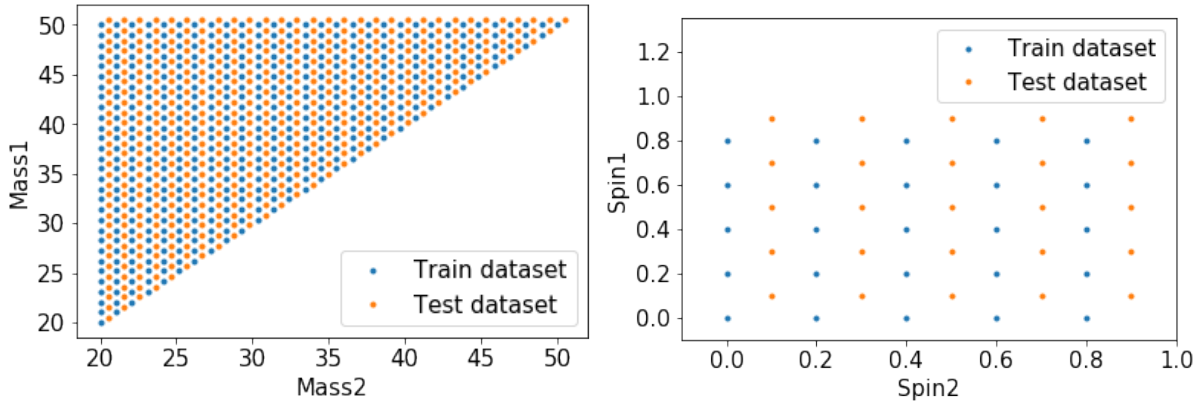

FIGURE 5.9: Parameters for train data and test data generation

```
import matplotlib.pyplot as plt
apx = 'IMRPhenomPv2'

import pycbc.noise
import pycbc.psd
delta_f = 1.0 / 16
flen = int(2000 / delta_f) + 1
flow = 40

T = 1024
psd_h1 = pycbc.psd.aLIGOZeroDetHighPowerGWINC(flen, delta_f, flow)
mass1_range = np.linspace(20.5,50.5,30)
mass2_range = mass1_range
spin1z_range = np.linspace(0.1, 0.9,5)
spin2z_range = spin1z_range
inclination=2.34
pol_range=0.9
det_h1 = Detector('H1')
X = []
for i1 in range(len(mass1_range)):
    for i2 in range(len(mass2_range)):
     print((i1, i2))
     if mass1_range[i1]<=mass2_range[i2]:
        for i3 in range(len(spin1z_range)):
```

```
            for i4 in range(len(spin2z_range)):
                seed_noise = random.randint(1,127)
                delta_t = 1.0 / 2000
                tsamples = T
                noise_h1 = pycbc.noise.noise_from_psd(tsamples, delta_t, psd_h1, seed=seed_noise)
                hp, hc = get_td_waveform(approximant=apx,
                                         distance=200,
                                         mass1=mass1_range[i1],
                                         mass2=mass2_range[i2],
                                         spin1z=spin1z_range[i3],
                                         spin2z=spin2z_range[i4],
                                         inclination=inclination,
                                         coa_phase=2.45,
                                         delta_t=1.0/2000,
                                         f_lower=40)
                signal_h1 = det_h1.project_wave(hp, hc,0,0, 0.9)
                signal_h1_array = np.array(signal_h1)[-T:] + np.array(noise_h1[-T:])
                list_h1 = signal_h1_array.tolist()
                h1_max_index = list_h1.index(max(list_h1))
                X0 = []
                X1 = []
                X0.append([100*np.array(signal_h1.sample_times[h1_max_index-10:h1_max_index+11])
                    -100*signal_h1.sample_times[h1_max_index-10], 10**21*signal_h1_array[
                    h1_max_index-10:h1_max_index+11]])
                X1.append([100*np.array(signal_h1.sample_times[h1_max_index-10:h1_max_index+11])
                    -100*signal_h1.sample_times[h1_max_index-10], 10**21*noise_h1[h1_max_index-10:
                    h1_max_index+11]])
                X.append(X0)
                X.append(X1)
X = np.array(X)
np.save('X_4_27_1_test200mpc.npy', X)
```

The block above shows the code to generate the test GW template waveform at 200Mpc. The GW waveforms and the noises are superimposed, to compose the noisy GW waveforms. At the same time, the same amount of noise signals are inserted into the waveforms. In total, 11,625 waveforms like Figure 5.7 and 11,625 noise signals were

included by the test data set.

So far we have prepared the train & test dataset.

5.6 Train and Test Process

5.6.1 Train Process – Parameter Optimization

The code block below represents the whole train process, and then store the CNN as a file.

```
import tensorflow as tf
import pathlib
from tensorflow.keras import datasets, layers, models
gpu_options = tf.compat.v1.GPUOptions(per_process_gpu_memory_fraction=0.8)
import numpy as np
from sklearn.model_selection import train_test_split

#Label set generation
label_set=[]
for i in range(23250/2):
    label_set.append(1)
    label_set.append(0)
label_set = np.array(label_set)

#Load the train data set file
X = np.load('X_4_27_1_train200mpc.npy')
X = np.asarray(X, np.float32)
Y = label_set
#Use 10 percent of the data to validate the model
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=0.1)

#Construct a CNN, Keras used.
model = models.Sequential()
model.add(layers.Conv2D(11, kernel_size = 1, strides=(1, 1), padding='valid',input_shape = (1,2,21),
    data_format='channels_first', activation='relu'))
model.add(layers.MaxPooling2D(pool_size=1, strides=None, data_format='channels_first'))
```

```
26 model.add(layers.Conv2D(22, kernel_size = 1, strides=(1, 1), padding='valid', activation='relu'))
27 model.add(layers.MaxPooling2D(pool_size=1, strides=None, data_format='channels_first'))
28 model.add(layers.Conv2D(44, kernel_size = 1, strides=(1, 1), padding='valid', activation='relu'))
29 model.add(layers.Dropout(0.02, noise_shape=None, seed=None))
30 model.add(layers.MaxPooling2D(pool_size=1, strides=None, data_format='channels_last'))
31 model.add(layers.Flatten())
32 model.add(layers.Dense(2, activation='softmax'))
33
34 #Optimizing. The Optimizer is Adam, default hyperparameters were used.
35 model.compile(optimizer=tf.keras.optimizers.Adam(0.001),
36     loss='sparse_categorical_crossentropy',
37     metrics=["accuracy"])
38
39 # Train the model in 10 epochs.
40 history = model.fit(X_train, Y_train, epochs=10, validation_data=(X_validation, Y_validation))
41
42 # Save the model as a h5 file , accessable when testing.
43 model.save('CNN_4_27_200mpc.h5')
```

About the details of the code block:

- Line 9 to line 13: Generate labels for the data set; '1' for GW waveforms and '0' for noises.
- Line 15 to line 20: Load and split the data set
- Line 23 to line 32: Construct the CNN architecture
- Line 35 to line 37: Call and define the optimizer, Adam.
- Line 40: Train the CNN, use Adam to optimize the parameters of CNN.
- Line 43: Save the CNN model as an h5 file.

Here we show the accuracy and loss during the trian process.

From Figure 5.3, we can see that in the first two epochs[1] the accuracy has reached one, which means the CNN has learned the features of the train data set. Also, because

[1] In terms of neural networks, an epoch refers to one cycle through the full training dataset.

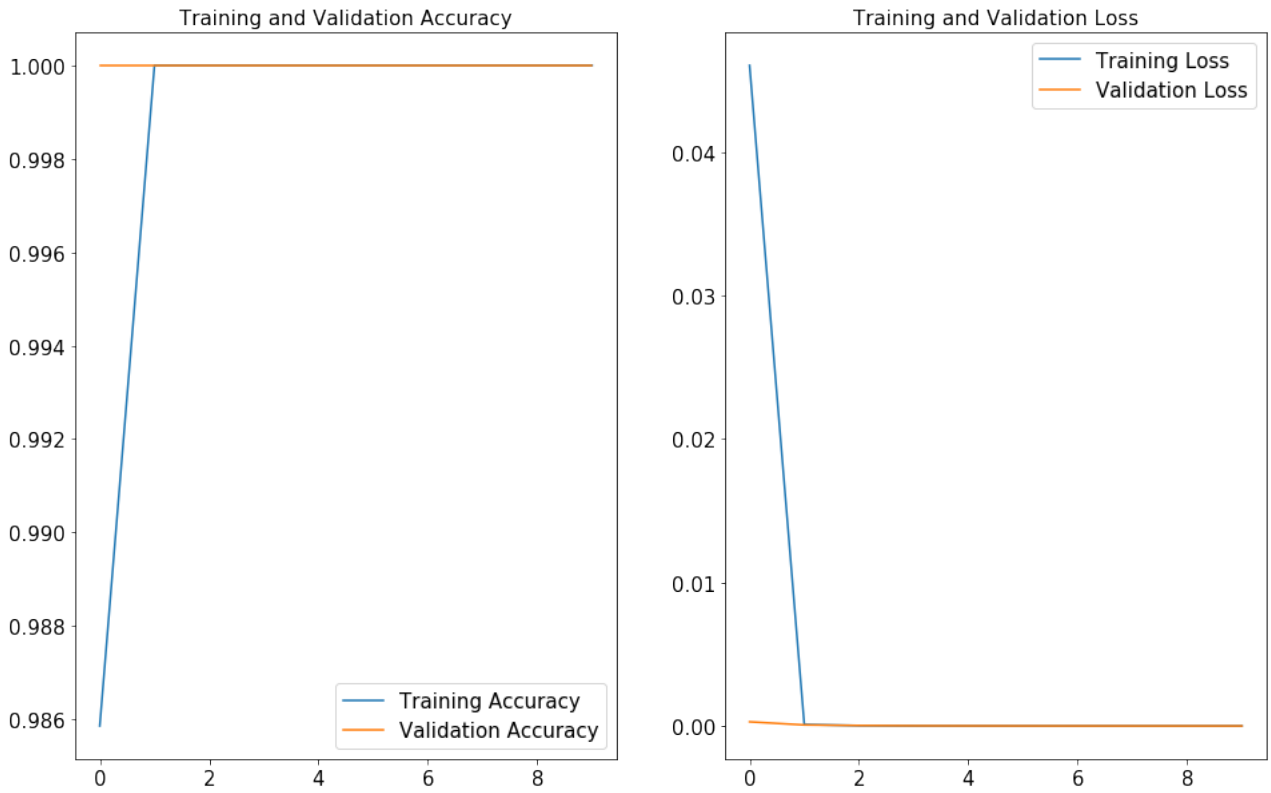

FIGURE 5.10: Loss and Accuracy of train and validation process

of the optimization, the loss is reduced to zero. Now we test the CNN with known data set to verify its generalization ability.

5.6.2 Test Process – Label Prediction

The code block below shows the test process.

```
import numpy as np
from tensorflow.keras import models
new_model = models.load_model('CNN_4_27_200mpc.h5') # Load the model file we trained.
test_set  = np.load('X_4_27_1_test200mpc.npy')
test_set  = np.asarray(test_set, np.float32)
test_label  = new_model.predict(test_set)  # Label prediction.
test_label  =  test_label . tolist ()

# Make the correct labels
label_set=[]
```

```
11 for i in range(23250/2):
12     label_set.append(1)
13     label_set.append(0)
14 label_set = np.array(label_set)
15 X = np.load('X_4_27_1_test200mpc.npy')
16 X = np.asarray(X, np.float32)
17 Y = label_set
18
19 # Calculate the loss and accuracy, according to the test data set and its correct labels.
20 test_loss, test_acc = new_model.evaluate(X, Y, verbose=2)
```

About the details of the code block:

- Line 3: Load the CNN model.
- Line 4 to line 5: Load the test data set.
- Line 6 to line 7: Predict the test data set labels.
- Line 10 to line 14: Make the correct labels, because the arrangement of GW waveform and noise signal is known.
- Line 15 to line 20: Calculate the loss and accuracy by comparing the correct labels and predicted labels.

The accuracy is defined as Figure 5.11 shows. The red text represents the correct prediction amount, and the black text represents the wrong prediction amount.

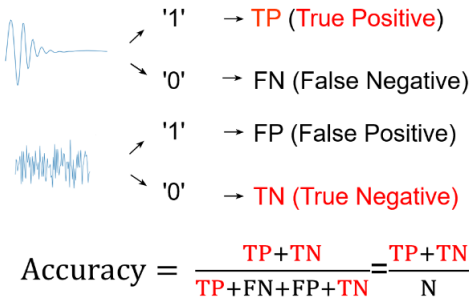

FIGURE 5.11: Detection accuracy definition

Because we have found that the test accuracy is severely restricted by the distance of CBC, here we show the figure of test accuracy versus distance.

In Figure 5.12, we ended the test at 800Mpc. At 800Mpc, about all of the predicted labels are '0', which means CNN recognizes every input data as noise. Because half of the input data is the noise signal, the accuracy closes to 0.5.

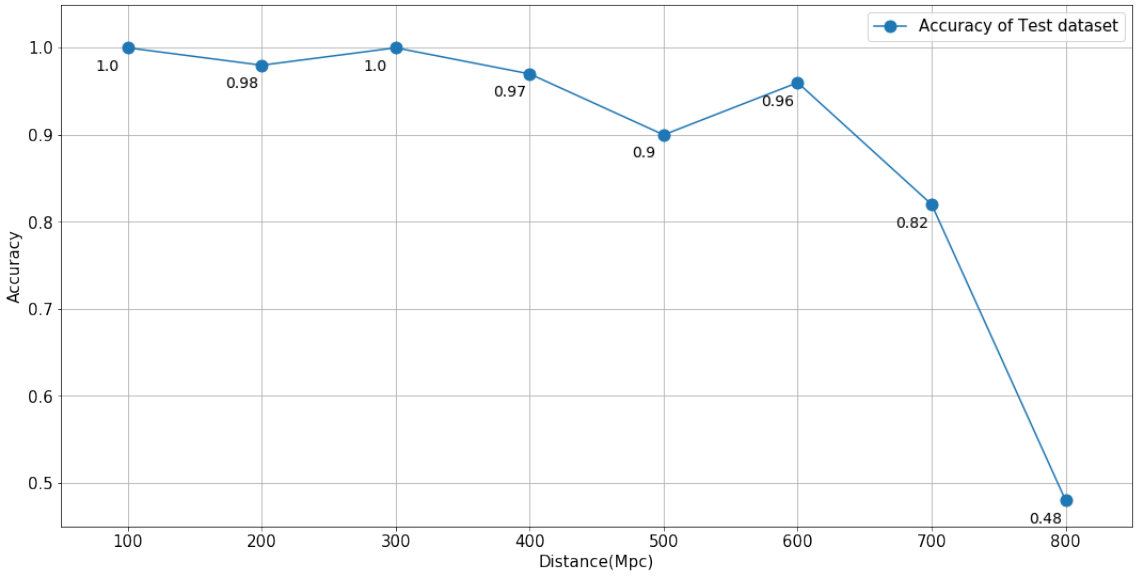

FIGURE 5.12: Detection accuracy versus distance

5.6.3 Result and Discussion

In this chapter we have talked about the detection of GW using CNN. We used accuracy to represent the generalization ability of CNN. As Figure 5.12 shows, the accuracy has a poor performance in a long distance. This is because the amplitude of GW is inversely proportional to the distance, as Figure 5.13 shows.

Also, the BBH having larger mass would emit stronger GW that is more likely to be detected. Meanwhile, small mass events are easily overwhelmed by noise.

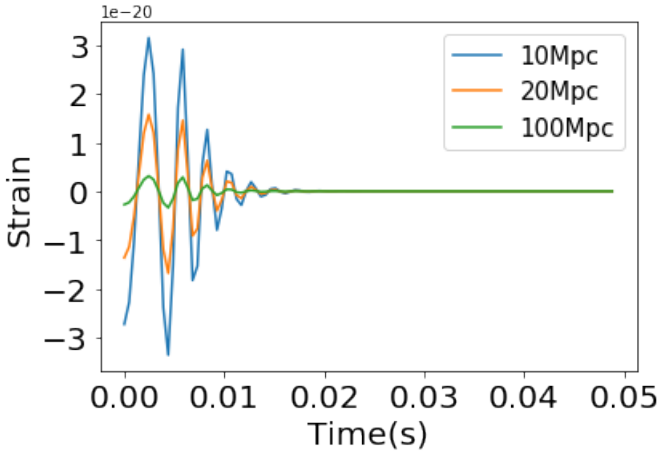

FIGURE 5.13: The further the distance is, the weaker the GW amplitude would be.

Chapter 6

Localization of Gravitational Waves Using Neural Network

This chapter talks about the localization of the GW source using CNN. Localization means figuring out the coming direction of the GW source. To position the source in a smaller area, simultaneous observation of at least two GW detectors is essential. Three or four detector's combination will benefit not only the detection confidence but also the sky localization.

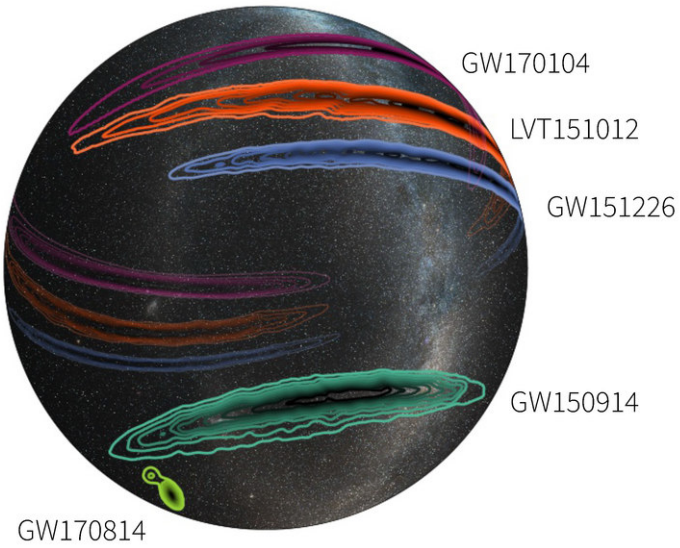

FIGURE 6.1: O1-O2-skymap

From Figure 6.1, we can see the localization precision of GW170814 is better than the other events because of the joining of Virgo. LIGO's observation obtained different

localization results. To strengthen the GW source search, they combined the result from four pipelines that used different analysis methods, to search particular sources.

As a three-detector network (LIGO Hanford, LIGO Livingston and Virgo) is convinced to have a more precise localization, we believe that introducing KAGRA to the network will benefit the localization work. Although the detectors' sensitivities are different, the effective use of the less sensitive detector information is still under study.

The following sections talk about using CNN to localize the GW source and the necessity of introducing KAGRA to the network.

6.1 Implement of GW Localization Using CNN

As we have discussed in Section 3.4 and Section 5.2, we can locate the gravitational wave source in a way similar to image recognition. We need to prepare the GW waveforms containing time lag information to infer the direction of GW. The time lags can be simulated by calling PyCBC's projection function, which means project one GW signal to several detectors, like Figure 6.2 shows.

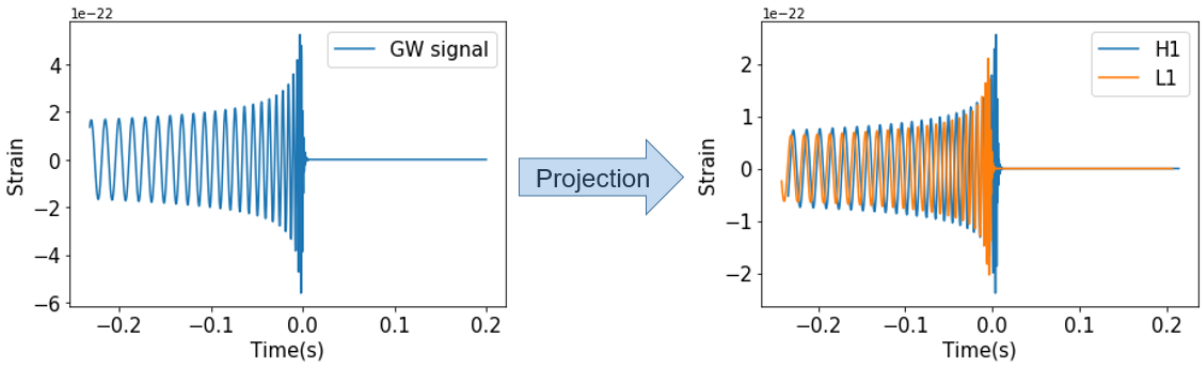

FIGURE 6.2: Waveform projection

The left side of Figure 6.2 is a GW waveform generated by PyCBC, and the right side shows the waveforms detected by GW observatories. The relative position of these waveforms is simulated by PyCBC basing on the direction of the GW source. Thus the time lag between detectors can be measured. In this research, we take the time interval between the highest peaks as time lags. From time lags among the detector network, we can estimate the direction of the GW source. In this way, a localization problem is transformed into a classification problem.

6.2 Architecture of the CNN for GW Localization and the Workflow

The CNN we used for detection is not suitable for the localization problem anymore, because sky localization is a multiclass classification but not a binary classification. The amount of classes depends on the sector number on the sky map. The finer we divide the sky map, the larger the sector amount, and the higher the localization precision. But at the same time, the localization accuracy would drop because the GW is easily localized to a neighboring sector. We tried to divide the sky map into 81 sectors, and a finer dividing caused low test accuracy. The CNN architecture for localization is similar to the CNN in Figure 5.3, but had a different output layer, as Figure 6.3 shows.

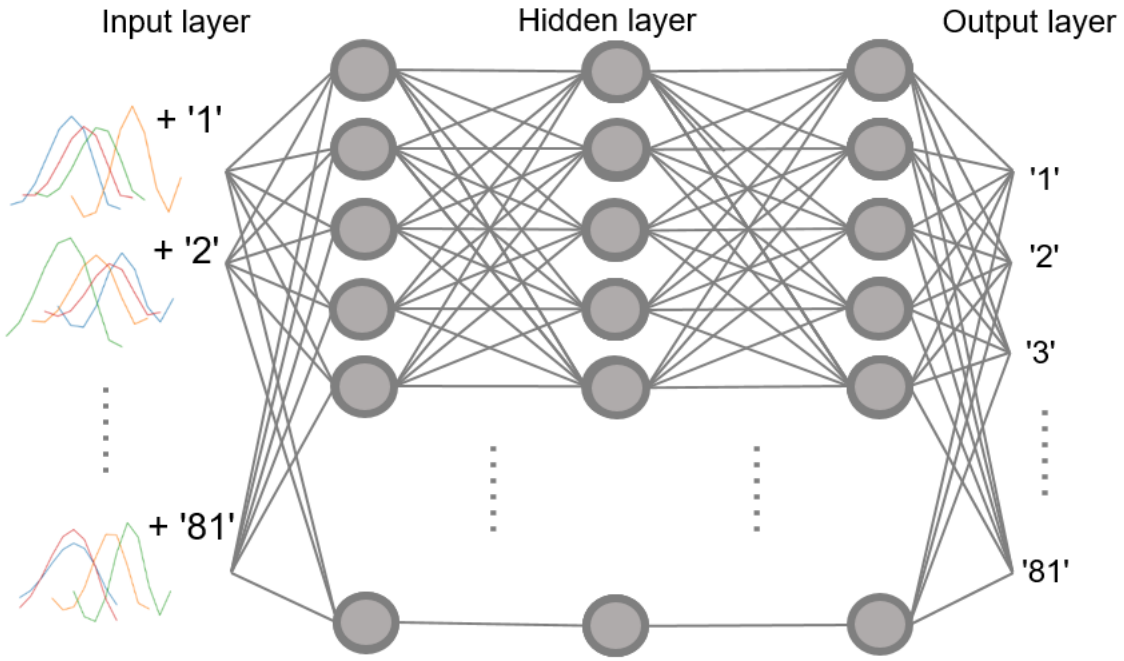

FIGURE 6.3: Workflow of the localization of gravitational waves

The specific architecture of the CNN is shown in Figure6.4.

This CNN has an output layer that gives a possibility list contains 81 numbers; each number corresponds to the possibility of one label.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 11, 2, 11)         55
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 11, 2, 11)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 11, 2, 22)         264
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 11, 2, 22)         0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 11, 2, 44)         1012
_________________________________________________________________
dropout (Dropout)            (None, 11, 2, 44)         0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 11, 2, 44)         0
_________________________________________________________________
flatten (Flatten)            (None, 968)               0
_________________________________________________________________
dense (Dense)                (None, 162)               156978
_________________________________________________________________
dense_1 (Dense)              (None, 81)                13203
=================================================================
Total params: 171,512
Trainable params: 171,512
Non-trainable params: 0
_________________________________________________________________
```

FIGURE 6.4: CNN architecture of gravitational waves localization

6.3 Train and Test Dataset Generation

In this study, the train and test data set is the low-parameter-density data set we used in the GW detection, but coming from 81 directions. The specific generation process is as follows:

1) Simulate a GW waveform emitted from the CBC.

2) Project the waveform to the detector network, get the signals in the four detectors.

3) Extract the ringdown part of each waveform.

4) Combine four ringdown period signals together.

So that the combined signals are maintained, like Figure 6.5 shows.

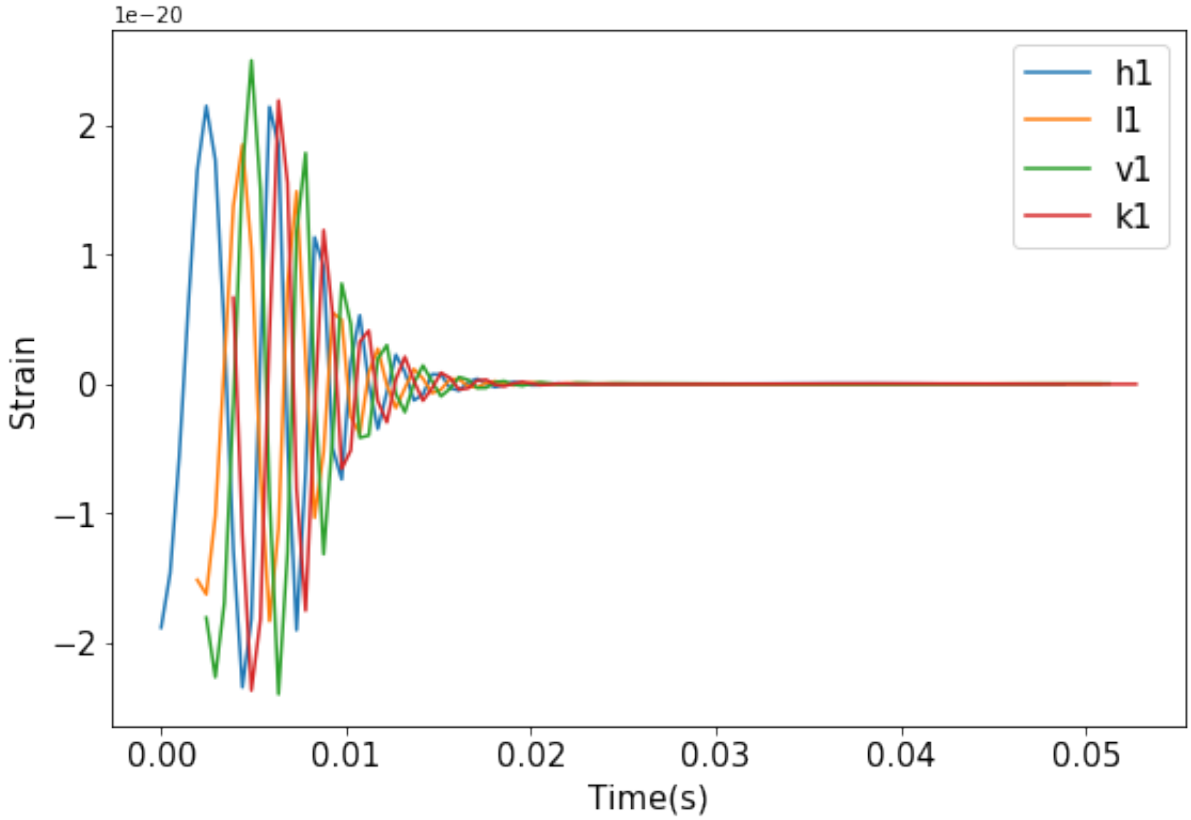

FIGURE 6.5: Ringdown period of the waveforms in four detectors

The labels 'h1', 'l1', 'v1', 'k1' are corresponding to LIGO Hanford, LIGO Livingston, Virgo, and KAGRA, respectively. But as we can see, the lines in Figure 6.5 is still a mess. The waveforms are too messy to distinguish them from each other. We have further reduced the size of the input:

5) Extract the maximum peaks from the ringdown periods. Specifically, select ten points around the crest as input data. Unlike GW detection research, the search window for a single detector is five milliseconds, but the window for the combined network is not fixed due to time lags.

Finally, the signal we used for data set generation is a combination of four peaks, like Figure 6.6 shows.

In total 12,960 pictures like Figure 6.6 was prepared.

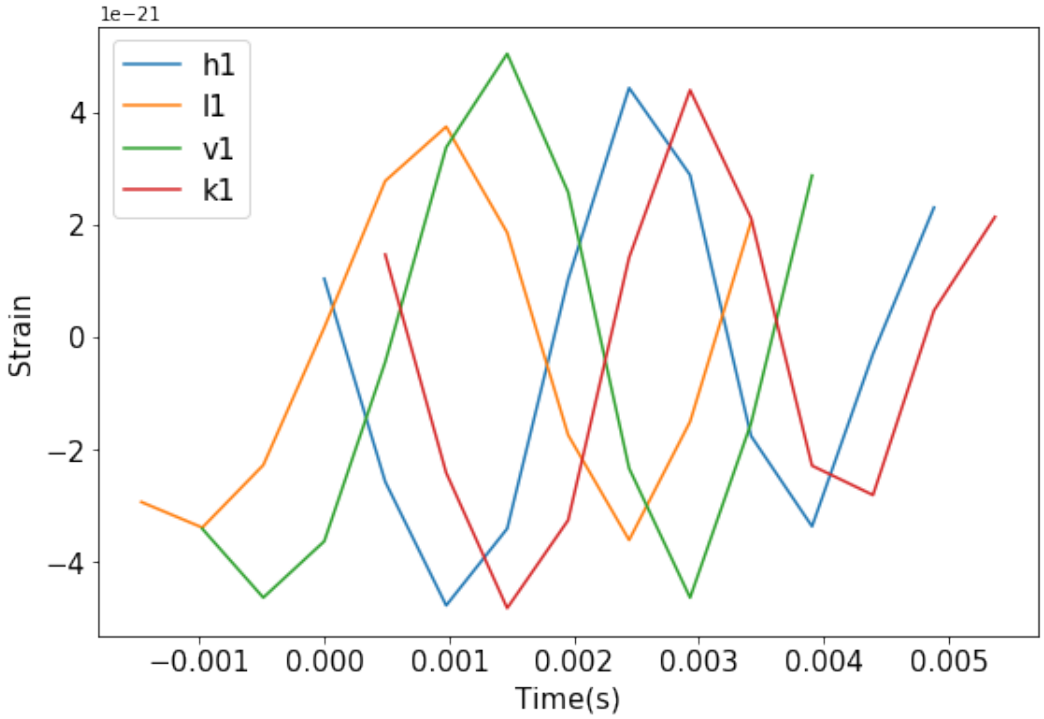

FIGURE 6.6: Highest waveform peaks of the HLVK network.

6.3.1 Train Dataset

Just like in the GW detection problem we talked about before, we made a training set and data set that do not have the same waveform at all. The specific parameter set is as follows:

- Masses of the binary range from $20M_{\odot}$ to $50M_{\odot}$, with an interval of $10M_{\odot}$.
- Spins of the binary range from 0.2 to 0.8, with an interval of 0.2.
- Distance of the binary range from 20Mpc to 200Mpc.
- Values of declination: [-1.41, -1.10, -0.74, -0.37, 0, 0.37, 0.74, 1.10, 1.41] (Unit: Radian; North pole & South Pole abandoned)
- Values of right ascension: $[\frac{2\pi}{9}, \frac{4\pi}{9}, \frac{6\pi}{9}, \frac{8\pi}{9}, \frac{10\pi}{9}, \frac{12\pi}{9}, \frac{14\pi}{9}, \frac{16\pi}{9}, 2\pi]$ (Unit: Radian)
- Inclination, polarization, coalescence phase and the other parameters are fixed.

Here we show the codes for train data set generation:

```
import pycbc
import numpy as np
import random
from pycbc.waveform import get_td_waveform
from pycbc.detector import Detector
apx = 'IMRPhenomPv2'
import matplotlib.pyplot as plt
import pycbc.noise
import pycbc.psd

# The color of the noise matches a PSD which you provide
flow = 40
delta_f = 1.0 / 40
flen = int(2000 / delta_f) + 1
T = 1024

psd_h1 = pycbc.psd.aLIGOZeroDetHighPower(flen, delta_f, flow)
psd_v1 = pycbc.psd.analytical.AdvVirgo(flen, delta_f, flow)
psd_l1 = pycbc.psd.aLIGOZeroDetHighPower(flen, delta_f, flow)
psd_k1 = pycbc.psd.analytical.KAGRADesignSensitivityT1600593(flen, delta_f, flow)
# NOTE: Inclination runs from 0 to pi, with poles at 0 and pi
# coa_phase runs from 0 to 2 pi.

det_h1 = Detector('H1')
det_l1 = Detector('L1')
det_v1 = Detector('V1')
det_k1 = Detector('K1')

# Choose a GPS end time, sky location, and polarization phase for the merger
# NOTE: Right ascension and polarization phase runs from 0 to 2pi
# Declination runs from pi/2. to -pi/2 with the poles at pi/2. and -pi/2.
dec_range = np.linspace(0.1-np.pi/2, -0.1+np.pi/2, 9)
ra_range = np.linspace(2*np.pi/9, 2*np.pi,9)

mass1_range = np.linspace(20,50,4)
mass2_range = mass1_range
```

```
spin1z_range = np.linspace(0.2,  0.8,4)
spin2z_range = spin1z_range
inclination_range=2.34
pol_range=0.9

X = []
Y = []
for i1 in range(len(dec_range)):
    for i2 in range(len(ra_range)):
        print((i1, i2))
        for i3 in range(len(mass1_range)):
            for i4 in range(len(mass2_range)):
                if mass1_range[i3]<=mass2_range[i4]:
                    for i5 in range(len(spin1z_range)):
                        for i6 in range(len(spin2z_range)):

                            seed_noise = random.randint(1,127)
                            # Generate noise
                            delta_t = 1.0 / 2000
                            tsamples = T

                            noise_h1 = pycbc.noise.noise_from_psd(tsamples, delta_t, psd_h1, seed=
                                seed_noise)
                            noise_l1 = pycbc.noise.noise_from_psd(tsamples, delta_t, psd_l1, seed=
                                seed_noise)
                            noise_v1 = pycbc.noise.noise_from_psd(tsamples, delta_t, psd_v1, seed=
                                seed_noise)
                            noise_k1 = pycbc.noise.noise_from_psd(tsamples, delta_t, psd_k1, seed=
                                seed_noise)

                            X0 = []

                            #Generate waveforms
                            hp, hc = get_td_waveform(approximant=apx,
                                                     distance=10,
                                                     mass1=mass1_range[i3],
```

```
                                        mass2=mass2_range[i4],
                                        spin1z=spin1z_range[i5],
                                        spin2z=spin2z_range[i6],
                                        inclination=inclination_range,
                                        coa_phase=2.45,
                                        delta_t=1.0/2000,
                                        f_lower=40)

                signal_h1 = det_h1.project_wave(hp, hc, ra_range[i2], dec_range[i1],
                    pol_range[i8])
                signal_l1 = det_l1.project_wave(hp, hc, ra_range[i2], dec_range[i1],
                    pol_range[i8])
                signal_v1 = det_v1.project_wave(hp, hc, ra_range[i2], dec_range[i1],
                    pol_range[i8])
                signal_k1 = det_k1.project_wave(hp, hc, ra_range[i2], dec_range[i1],
                    pol_range[i8])

                signal_h1_array = np.array(signal_h1)[-T:] #+ np.array(noise_h1)
                signal_l1_array = np.array(signal_l1)[-T:] #+ np.array(noise_l1)
                signal_v1_array = np.array(signal_v1)[-T:] #+ np.array(noise_v1)
                signal_k1_array = np.array(signal_k1)[-T:] #+ np.array(noise_k1

                list_h1 = signal_h1_array.tolist()
                list_l1 = signal_l1_array.tolist()
                list_v1 = signal_v1_array.tolist()
                list_k1 = signal_k1_array.tolist()

                h1_max = max(list_h1)*10**21
                l1_max = max(list_l1)*10**21
                v1_max = max(list_v1)*10**21
                k1_max = max(list_k1)*10**21


                h1_max_index = list_h1.index(max(list_h1))
                l1_max_index = list_l1.index(max(list_l1))
                v1_max_index = list_v1.index(max(list_v1))
```

```
101                         k1_max_index = list_k1.index(max(list_k1))
102
103                         X0.append([100*np.array(signal_h1.sample_times[h1_max_index-5:
                                h1_max_index+6])-100*signal_h1.sample_times[h1_max_index-5],
104                                     10**21*signal_h1_array[h1_max_index-5:h1_max_index+6]])
105                         X0.append([100*np.array(signal_l1.sample_times[l1_max_index-5:
                                l1_max_index+6])-100*signal_l1.sample_times[h1_max_index-5],
106                                     10**21*signal_l1_array[l1_max_index-5:l1_max_index+6]])
107                         X0.append([100*np.array(signal_v1.sample_times[v1_max_index-5:
                                v1_max_index+6])-100*signal_v1.sample_times[h1_max_index-5],
108                                     10**21*signal_v1_array[v1_max_index-5:v1_max_index+6]])
109                         X0.append([100*np.array(signal_k1.sample_times[k1_max_index-5:
                                k1_max_index+6])-100*signal_k1.sample_times[h1_max_index-5],
110                                     10**21*signal_k1_array[k1_max_index-5:k1_max_index+6]])
111
112                         X.append(X0)
113
114 X = np.array(X)
115 np.save('X_4_28_1_train10mpc.npy', X)
```

Explanation on the code block:

- Line 17:to line 27: Call the detectors and their PSDs.
- Line 32 to line 40: CBC parameter assignment.
- Line 44 to line 101: GW waveform generation. (Note the noise is not mixed here.)
- Line 103 to line 115: Collect the time and amplitude data of eleven points surrounding the crest, save them in the npy file.
- Taking the starting time point of H1 as the reference point, the time of the other waveforms is shifted.
- Noise is not mixed in the waveforms for training. But when we generate the test data set, noises are superimposed.
- Nine declinations and nine right ascensions appointed 81 points on sky maps, which divide the sky into 81 sectors.

6.3.2 Test Dataset

The test data set for localization consists of noisy waveforms based on different parameters with training. Figure 6.8 is one of the waveforms. Because the detectors' sensitivities are different, we should use each detector's PSD to color the noise. The PSDs can be inquired from PyCBC.

At present(2020/7), the O3 observation run will end soon, LIGO and Virgo will reach their designed sensitivity in O4 run. However, KAGRA's sensitivity is below its expected standard. According to the forecast, KAGRA's actual sensitivity in O4 run would be the expected sensitivity in O3 run. So in this research, we use the expected sensitivity of LIGO and VIRGO during O4 run as reference sensitivity. And also use the expected sensitivity of KAGRA during O3 run as the reference sensitivity.

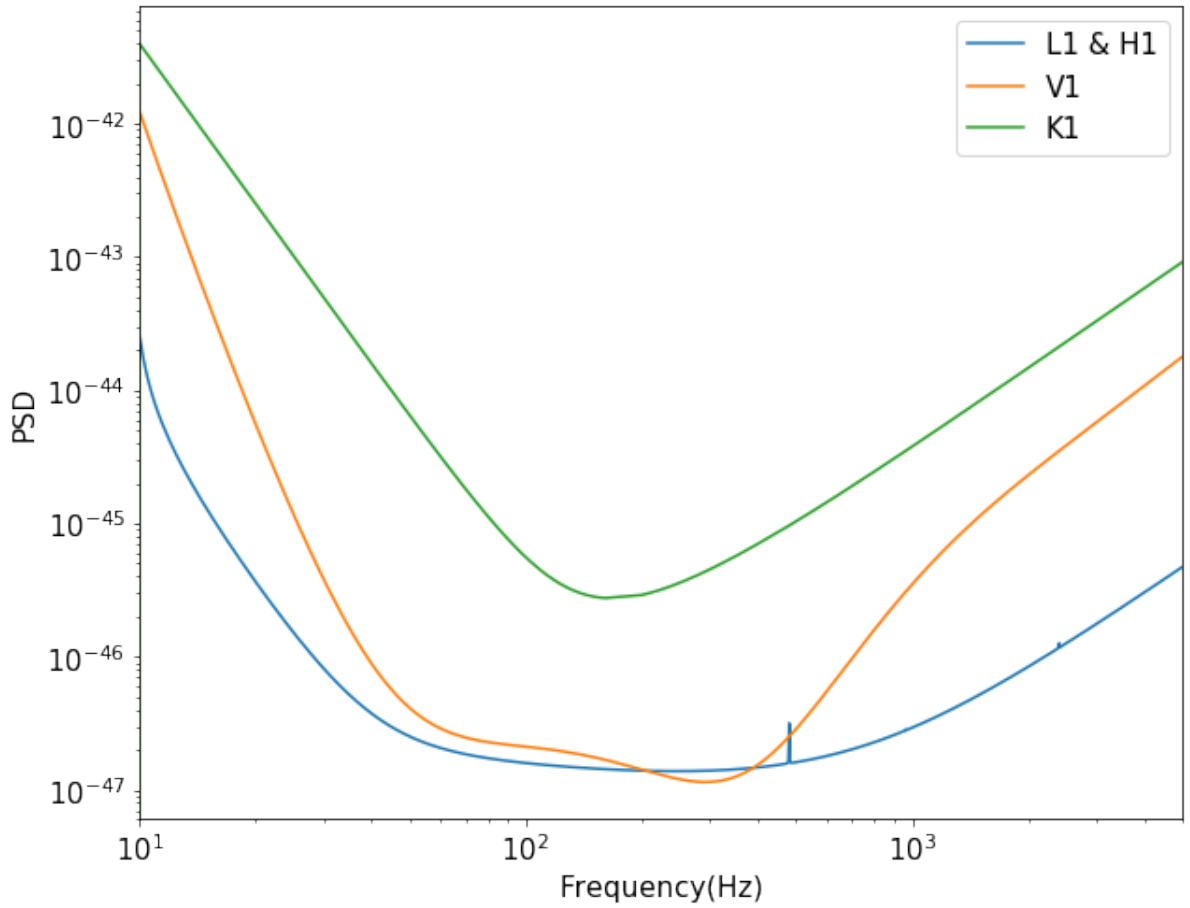

FIGURE 6.7: PSDs of LIGO, Virgo in O4 run and KAGRA in O3 run

At present, LIGO has the best sensitivity overall, followed by Virgo. Data quotes

from a report of LIGO in July 2019. To make the noisy test data set, we mix the noise signals with pure GW signals, to get the noisy waveforms like Figure 6.8 shows.

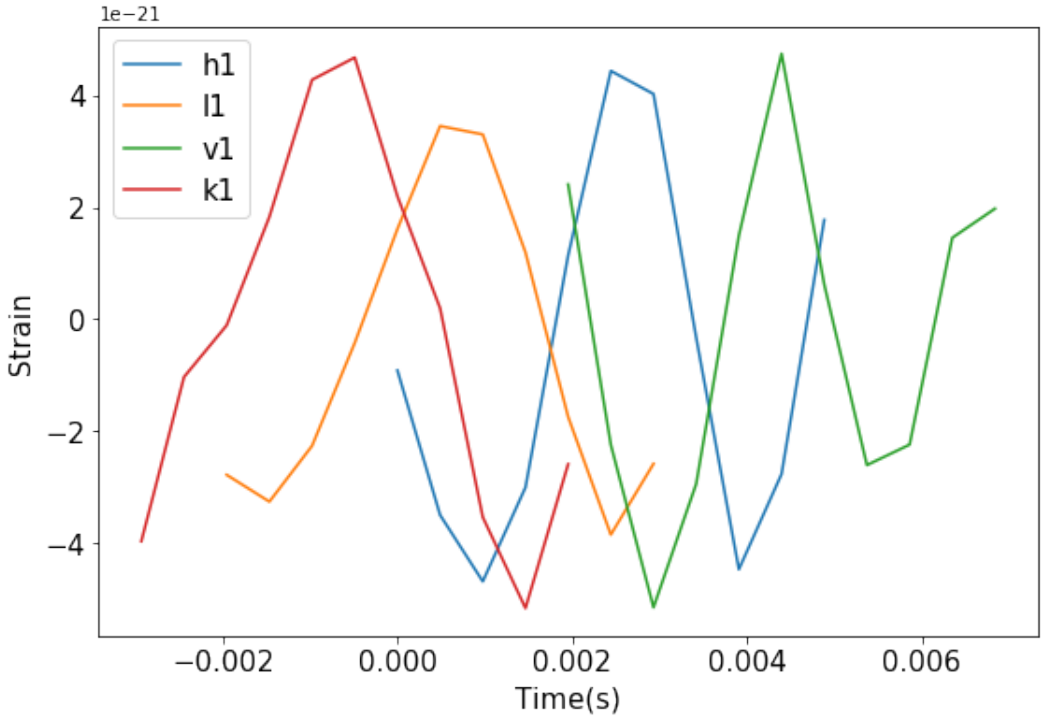

FIGURE 6.8: Noisy waveforms for dataset generation

The parameters are shown below.

- Masses of the binary range from $25M_{\odot}$ to $55M_{\odot}$, with an interval of $10M_{\odot}$.
- Spins of the binary range from 0.1 to 0.7, with an interval of 0.2.
- Distance, declination, and right ascension were the same with the train data set.
- Inclination, polarization, coalescence phase, and the other parameters are fixed.

About the test data generation code, we don't show the complete codes in order to save the space. The code can be written out by revising the train data set generation codes:

1) Mix noise to pure GW waveforms. Specifically, delete the '#' in line 82~85.
2) Modify the parameter assignment, from line 32 to line 40.
3) Revise the file name in line 115.

6.4 Train and Test Process

6.4.1 Train Process

The block below shows the code for training.

```
import tensorflow as tf
import pathlib
from tensorflow.keras import datasets, layers, models
gpu_options = tf.compat.v1.GPUOptions(per_process_gpu_memory_fraction=0.8)
import numpy as np

sector = 160

label_set=np.zeros((sector*81))
for i in range(81):
    label_set[i*sector:sector*(i+1)] = i
print(label_set.shape)

from sklearn.model_selection import train_test_split
X = np.load('X_4_28_1_train10mpc.npy')
X = np.asarray(X, np.float32)
Y = label_set

print(X.shape)
print(Y.shape)
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=0.1)

# construct a CNN, Keras used.
model = models.Sequential()
model.add(layers.Conv2D(11, kernel_size = 1, strides=(1, 1), padding='valid',input_shape = (4,2,11),
    data_format='channels_first', activation='relu'))
model.add(layers.MaxPooling2D(pool_size=1, strides=None, data_format='channels_first'))
model.add(layers.Conv2D(22, kernel_size = 1, strides=(1, 1), padding='valid', activation='relu'))
model.add(layers.MaxPooling2D(pool_size=1, strides=None, data_format='channels_first'))
model.add(layers.Conv2D(44, kernel_size = 1, strides=(1, 1), padding='valid', activation='relu'))
```

```
30 model.add(layers.Dropout(0.02, noise_shape=None, seed=None))
31 model.add(layers.MaxPooling2D(pool_size=1, strides=None, data_format='channels_last'))
32 model.add(layers.Flatten())
33 model.add(layers.Dense(162, activation='relu'))
34 model.add(layers.Dense(81, activation='softmax'))
35
36 model.compile(optimizer=tf.keras.optimizers.Adam(0.001), loss='sparse_categorical_crossentropy', metrics
       =["accuracy"])
37
38 history = model.fit(X_train, Y_train, epochs=50, validation_data=(X_validation, Y_validation))
39
40 model.save('CNN_4_28_10mpc.h5')
```

In the block above,

- Line 7:to line 12: Labels generation.
- Line 14 to line 21: Load and split the train data set.
- Line 24 to line 34: CNN construction
- Line 36: Make a parameter optimizaer
- Line 38: Train the CNN within 50 epochs.
- Line 40: Save the trianed CNN model.

Loss and acuracy during the train process is shown by Figure 6.9.

Compared to the GW detection problem, we can see that the CNN took longer epochs to learn from the train data for the localization of the GW source.

6.4.2 Test Process

The block below shows the test process.

```
1 import numpy as np
2 from tensorflow.keras import models
3
4 new_model = models.load_model('CNN_4_28_10mpc.h5') # load the model file we have trained before.
```

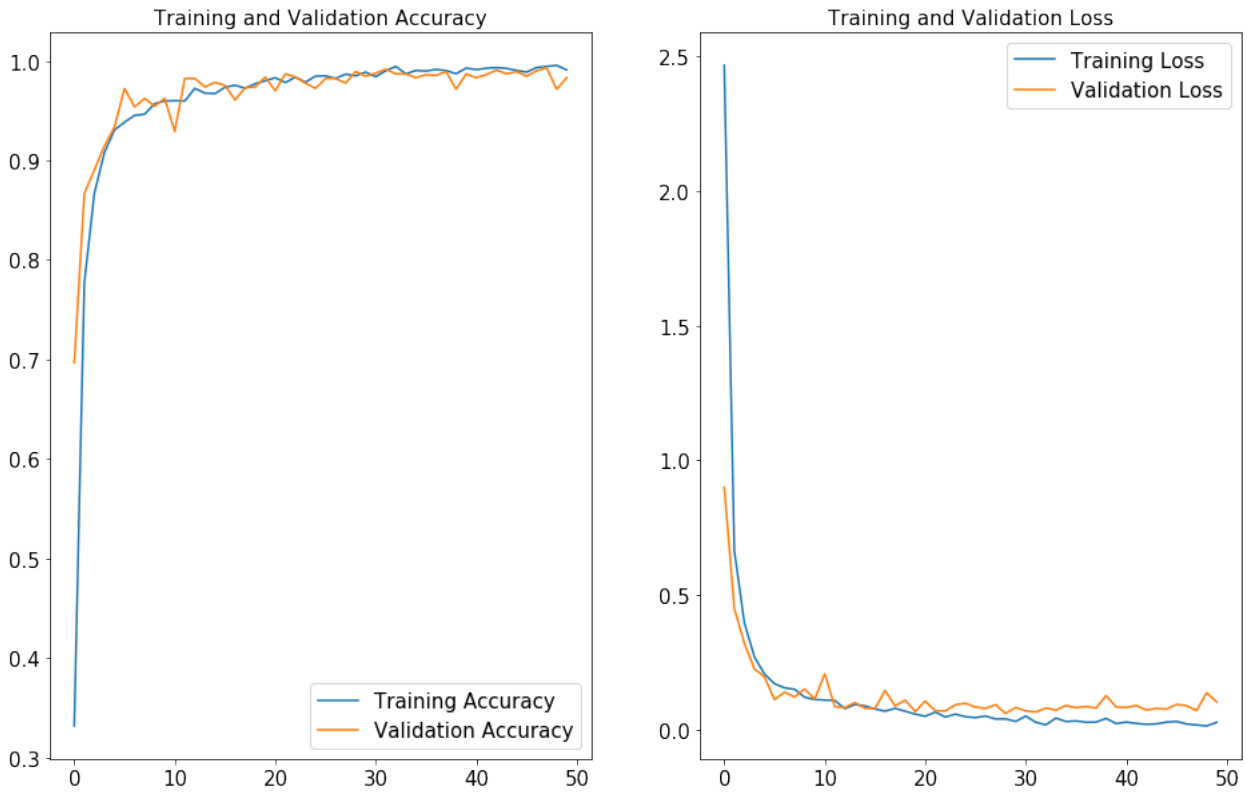

FIGURE 6.9: Noisy waveforms for dataset generation

```
test_set = np.load('X_4_28_1_test10mpc.npy')
test_set = np.asarray(test_set, np.float32)
test_label = new_model.predict(test_set)
test_label = test_label.tolist()

sector = 160
label_set=np.zeros((sector*81))
for i in range(81):
    label_set[i*sector:sector*(i+1)] = i

X = np.load('X_4_28_1_test10mpc.npy')
X = np.asarray(X, np.float32)
Y = label_set

test_loss, test_acc = new_model.evaluate(X, Y, verbose=2)
```

Explanations on the codes:

- Line 4: Load the trained CNN model.
- Line 5 to line 6: Load the test data set.
- Line 7 to line 8: Label prediction.
- Line 10 to line 13: Correct label making.
- Line 15 to line 19: Calculate the loss and accuracy by comparing the right labels and predicted labels.

Like we talked in Section 5.6.2, the localization accuracy was mainly restricted by the distance of CBC. Figure 6.10 shows the relation of the localization accuracy and the distance:

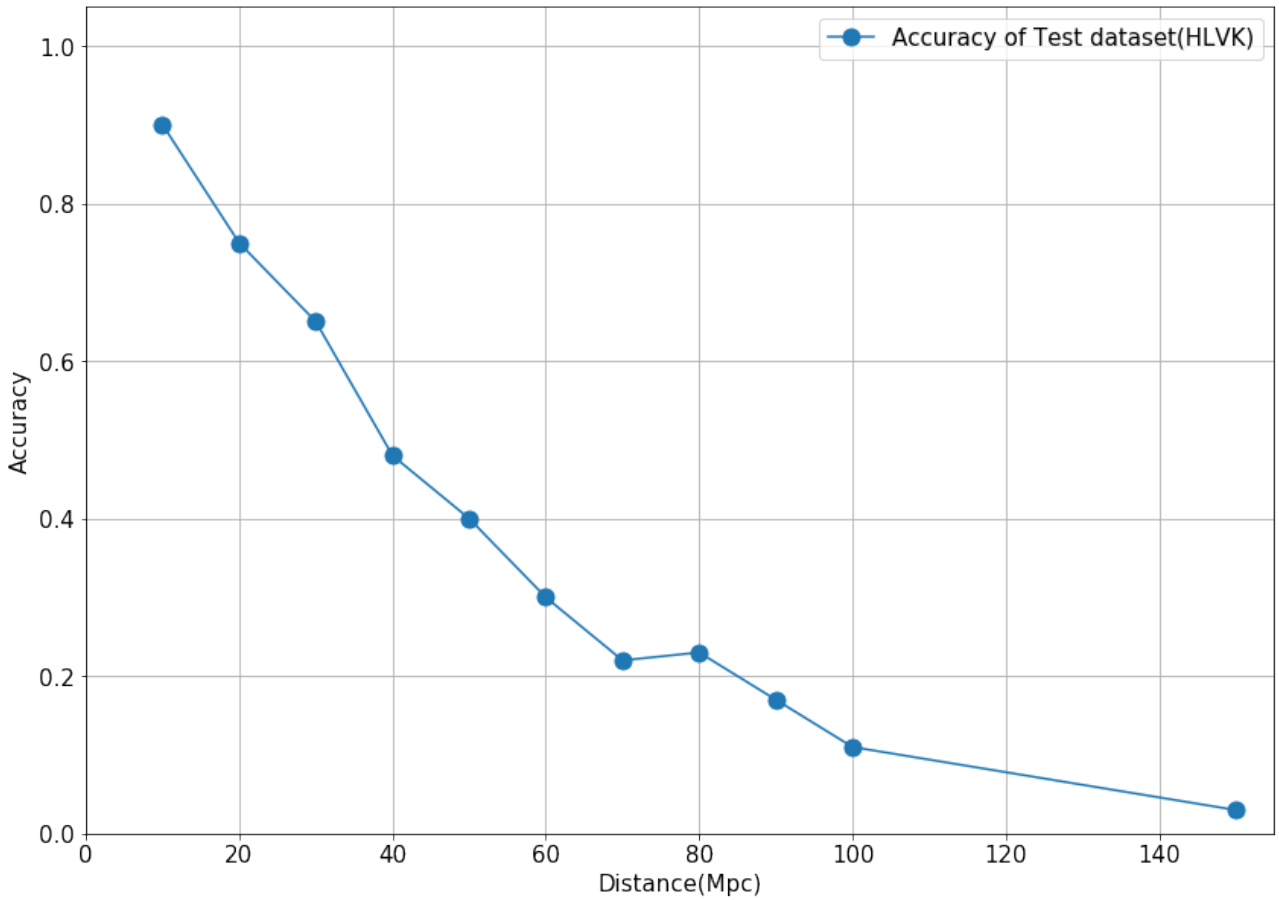

FIGURE 6.10: Localization accuracy versus distance

We can see in Figure 6.10, the CNN has a high classification accuracy when the distance is small, but the accuracy drops when we increase the distance. When distance=200Mpc, the accuracy is only 0.13, it means 87% input waveforms are localized into the wrong sectors.

6.4.3 Research of a Three-detector Network Localization

In this chapter, the localization research is based on the combination of four detectors. But in fact, KAGRA has just joined the network for a short time, few works have proceeded with KAGRA. We want to know whether KAGRA's joining is valuable, and when KAGRA can contribute to the detector network. So in this chapter, we removed KAGRA from the network and tried to localize the GW source using LIGO and Virgo data.

We still examine the relationship between accuracy and distance using three detectors (LIGO Hanford, LIGO Livingston and Virgo).

About the data set preparation, the parameter set of the template bank was the same as above. Also, CNN architecture was revised to adapt the three-dimensional data set. Here we compare the result of the three-detector network localization accuracy and the four-detector network localization accuracy, as Figure 6.11 shows.

Unfortunately, introducing KAGRA has a small contribution to the network. It helps improve the localization accuracy only when the CBC is near enough. According to the research in the section below, we guess that due to the low sensitivity of KAGRA, the signal it detects is easily overwhelmed by noise, so KAGRA gives that wrong triggers. To some extent, KAGRA hinders localization analysis when the CBC is far away.

From the current point of view, people can not find positive reasons for KAGRA to join the localization network.

6.4.4 Increase KAGRA Sensitivity to Improve Localization Accuracy

This section talks about the study of high-sensitivity KAGRA's contribution to the detector network. As Figure 3.2 shows, KAGRA needs further sensitivity improvement to catch up with other detectors. Here we simulated a higher sensitivity KAGRA which has the expected sensitivity during O4 observing run, like Figure 6.12 shows.

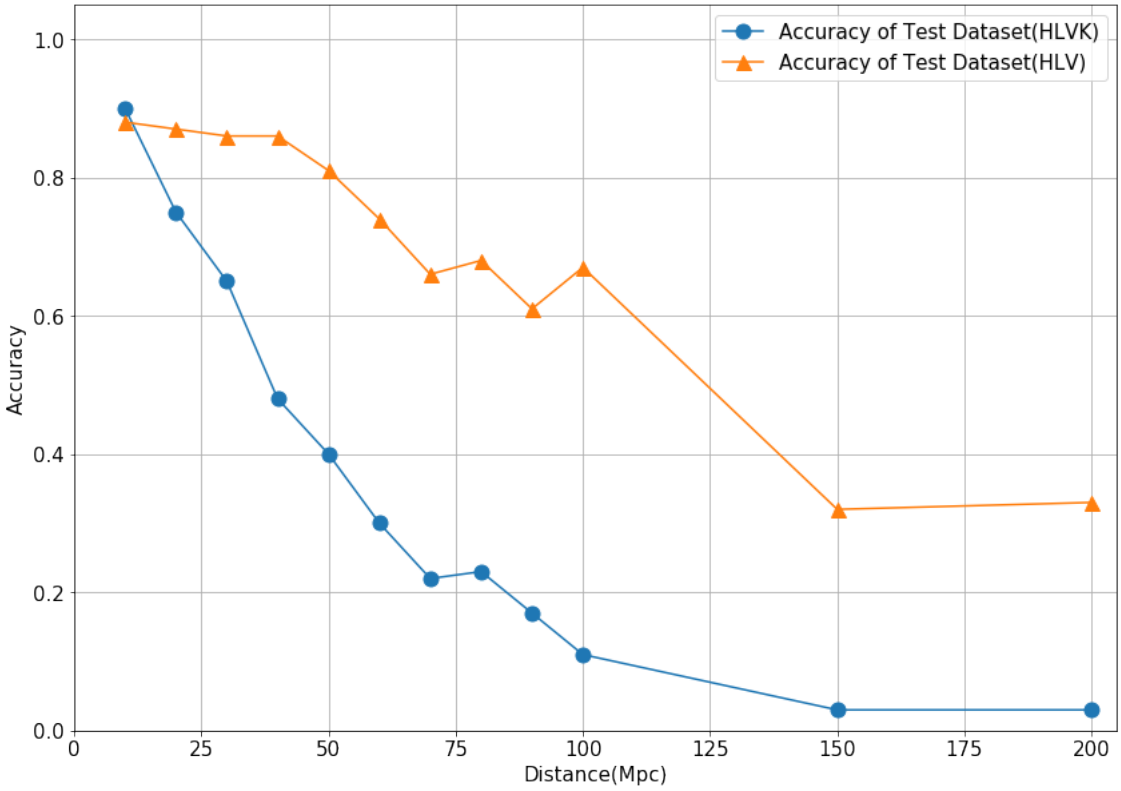

FIGURE 6.11: Accuracy of HLVK network (blue) and HLV network(red)

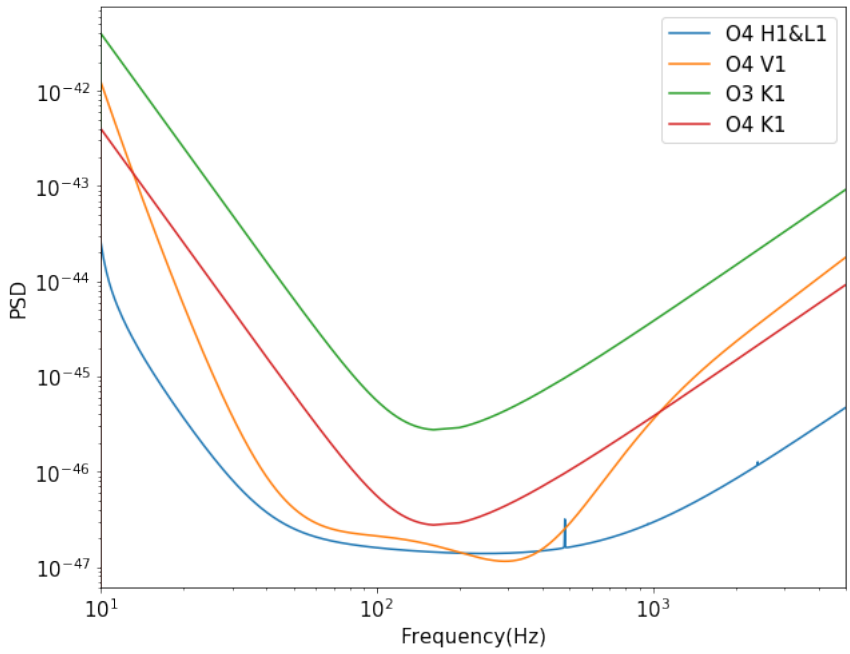

FIGURE 6.12: PSDs of HLVK in O4 run, and KAGRA in O3 run

The sensitivity of O4 KAGRA is obtained by dividing the current PSD by ten times.

Figure 6.13 shows the result of GW localization accuracy using network contains more sensitive KAGRA.

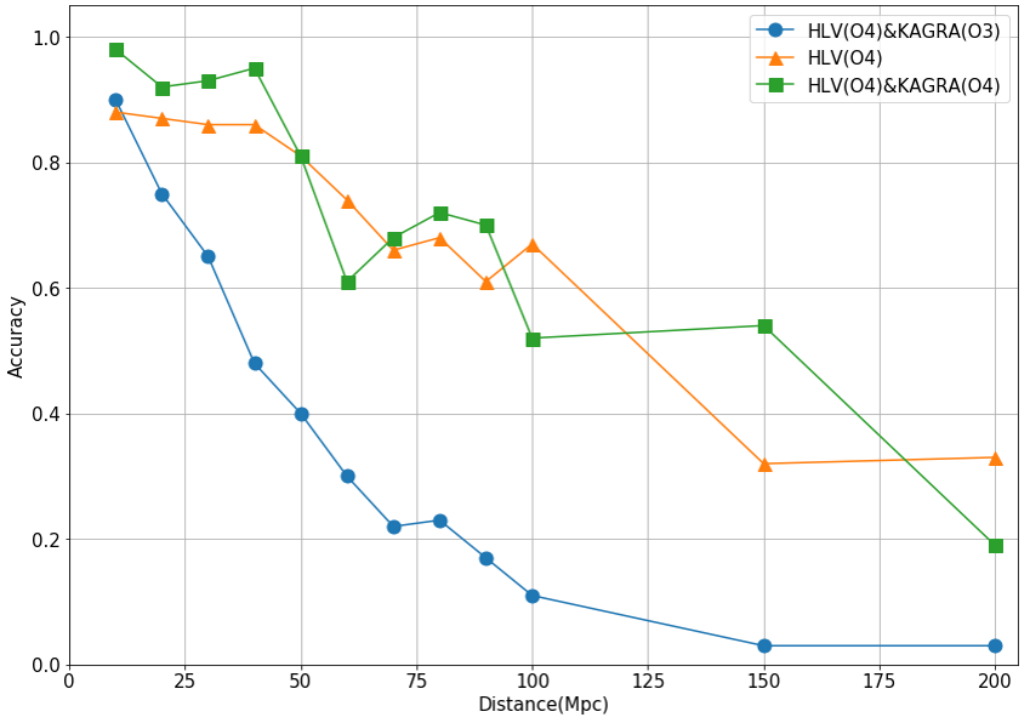

FIGURE 6.13: Accuracy improvements by raising KAGRA sensitivity.

We can see that increasing the sensitivity of KAGRA will significantly improve the accuracy of localization. If the researchers of KAGRA can suppress the noise power to one-tenth of the current status, then people have a good reason to let KAGRA join the localization network.

Chapter 7

Case Study: Injection of GW170814 Data

This chapter talks about a case study of GW170814 data analysis. GW170814 is a BBH merger event detected by LIGO and Virgo in Aug. 2017. The raw data has been released for researching. We take this event as a research object because GW170814 has the ideal binary masses and a high SNR = 15.9. The real data was imported from the GW Open Science Center[1] via PyCBC.

7.1 GW170814 Detection

7.1.1 Data Pre-processing

Raw data from LIGO and Virgo can be accessed, but here we choose LIGO Livingston, because L1 has better sensitivity at that moment. Figure 7.1 shows the signal in the L1 detector around GPS time 1186741861.53, the event occurring time. HPF and LPF were adopted to filter noises, and the strain was also smoothed.

As we talked in chapter 4, the search window for GW detection is ten milliseconds. So we extract the highest peak out like Figure 7.2 shows. Time starts from the leftmost side of the waveform. The duration is ten milliseconds.

In order to test CNN's classification ability, noise data is also essential. We randomly selected a part of the L1 signal after the merger as a noise signal. The noise signal is shown in Figure 7.3.

So far the data pre-processing has finished.

[1] https://www.gw-openscience.org/eventapi/html/GWTC-1-confident/GW170814/v3/

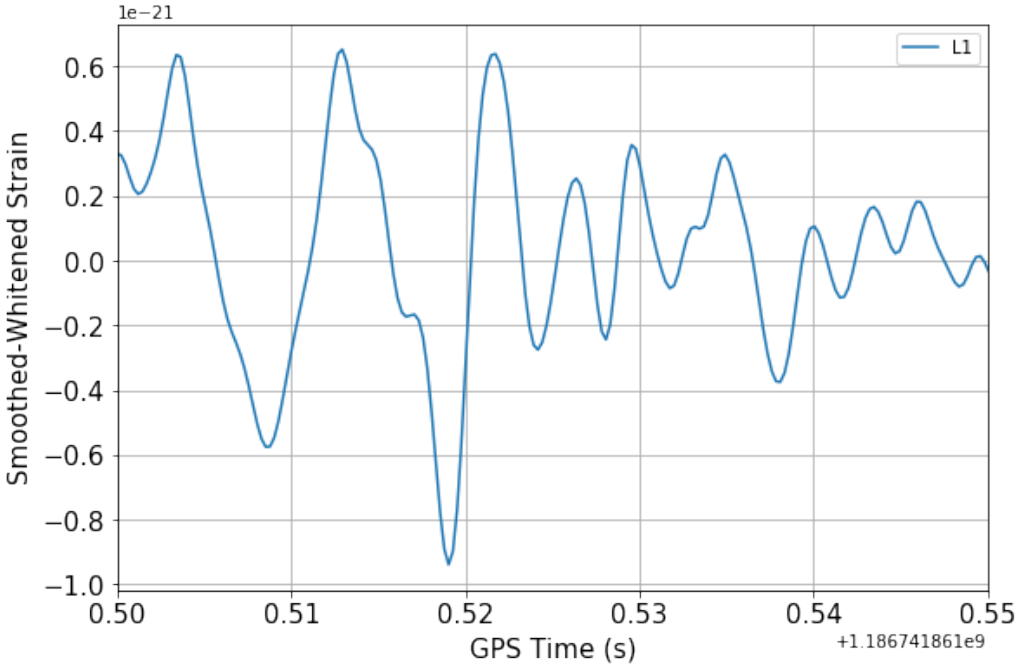

FIGURE 7.1: H1 signal of event GW170814

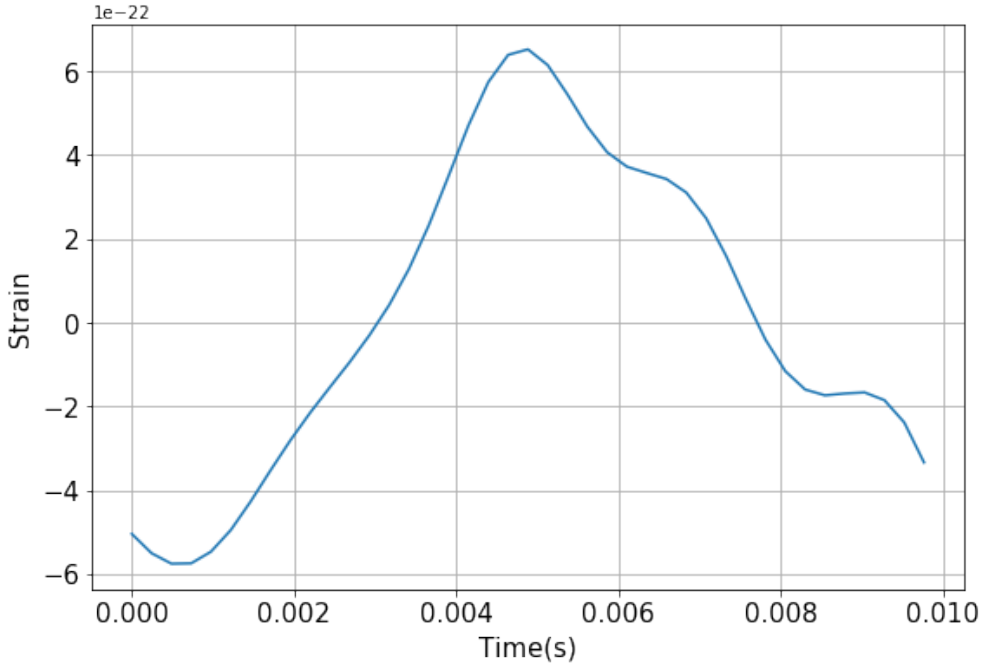

FIGURE 7.2: H1 signal of event GW170814, around the highest peak

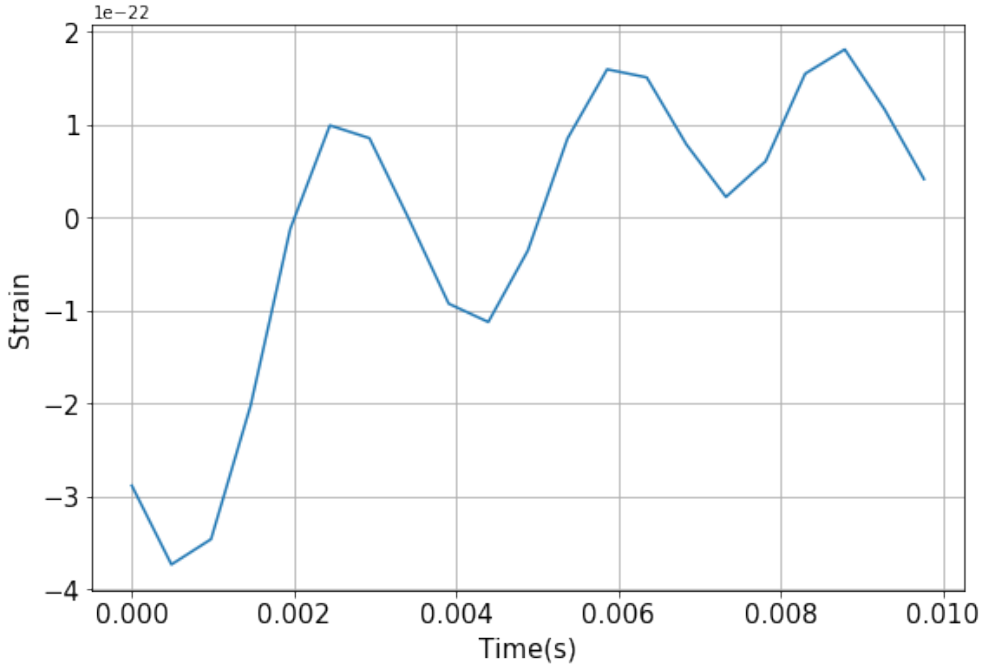

FIGURE 7.3: Noise signal in H1

7.1.2 Injection Test

Now a well-trained CNN is used for injection test. The workflow follows Figure 5.4. The GW waveform with label '1' and noise signal with label '0' was send to the CNN, then the classification result is show in Table 7.1.

TABLE 7.1

	GW possibility	noise possibility
GW signal	0.811	0.189
noise signal	0.001	0.999

We can see that CNN give high probabilities to the correct label. From this we believe that the injection experiment was successful.

However, before the classification experiment conducts, the signal was pre-processed. In another test, we did not filter the real signal and found that the GW signal has been submerged in noise. In this case, this experiment is impossible to complete.

7.2 GW170814 Localization

This section talks about the localization of GW170814 using real data. Since KAGRA was not able to operate at the time, here we use the data of the three-detector network(HLV).

7.2.1 Data Pre-processing

Via PyCBC, real data of LIGO and Virgo can be imported from GW Open Science center. As we talked in the previous section, the real data is filtered by HPF and LPF, then smoothed. Signals in LIGO and Virgo around GW170814 occurring time are shown in Figure 7.4.

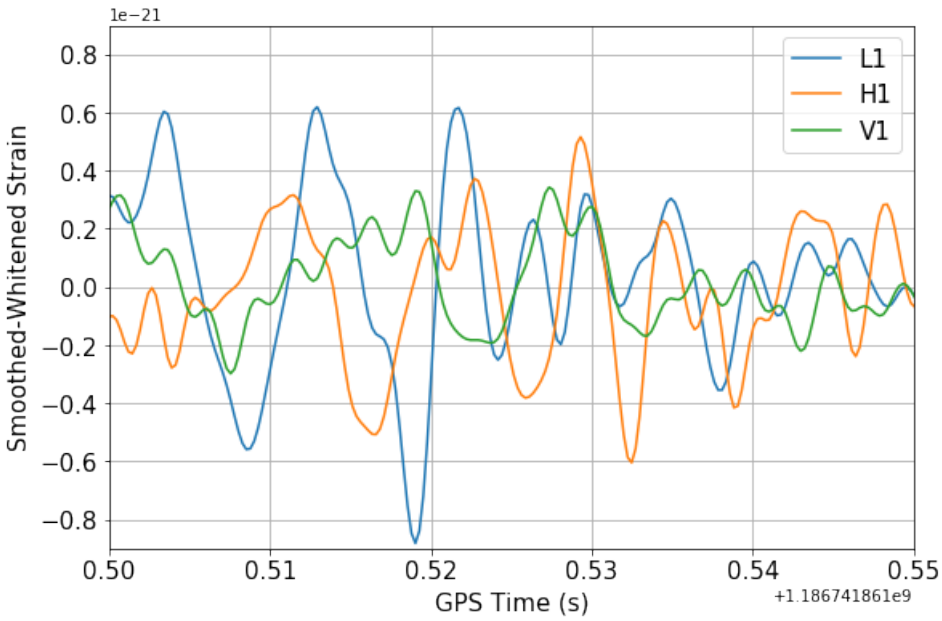

FIGURE 7.4: Signals of GW170814 in HLV detectors, H1 waveform inverted.

It isn't easy to recognize the position of the peaks in each detector, especially in the V1 detector. So we shift the signal according to the detection time delay: use L1 as the reference detector, move H1 signal forward for 8 ms, and move V1 signal forward for 14 ms[16]. Aligned signals are shown in Figure 7.5.

Now we can clearly see the simultaneous response of each detector. We select the signals between 0.51 and 0.52 seconds as the input data. After restoring their relative positions, we got the input data for localization, like Figure 7.6 shows.

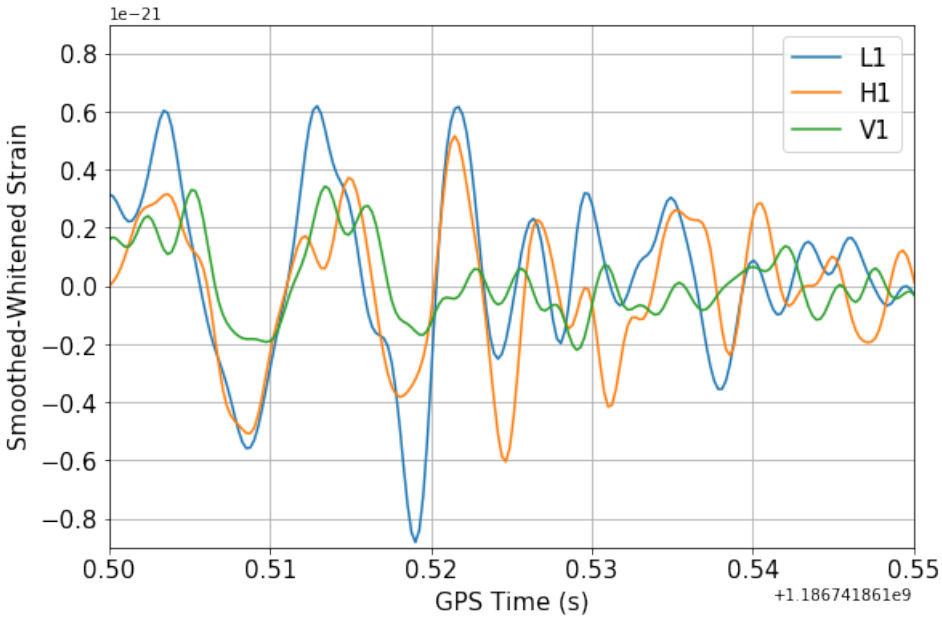

FIGURE 7.5: Aligned signals of GW170814 in the detectors, H1 waveform inverted.

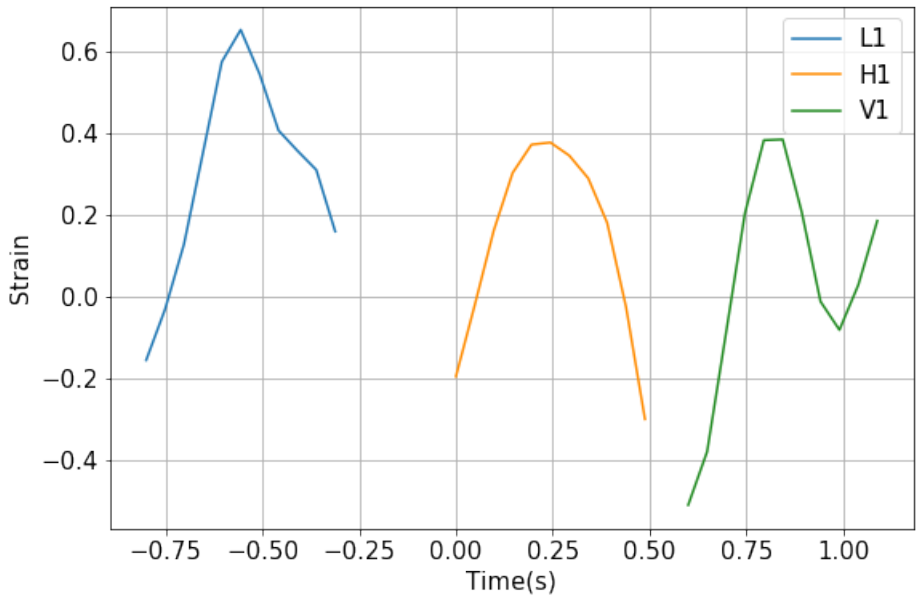

FIGURE 7.6: Peaks in HVL detectors, sample frequency is 2kHz. Strain was enlarged by 10^{21} times, time was enlarged by 10^2 times.

The time and amplitude information were sent to a trained CNN, as input data. The workflow follows Figure 6.3.

7.2.2 Injection Test

The CNN predicts the label corresponds to input data. As a result, the possibility of the 19th sector has a maximum possibility. That is to say, the CNN predicts that the input data is more close to the point (RA = $\frac{2\pi}{9}$, Dec = -0.74) than any other points, with a possibility of 99.9 percent. Figure 7.7 shows the localization result.

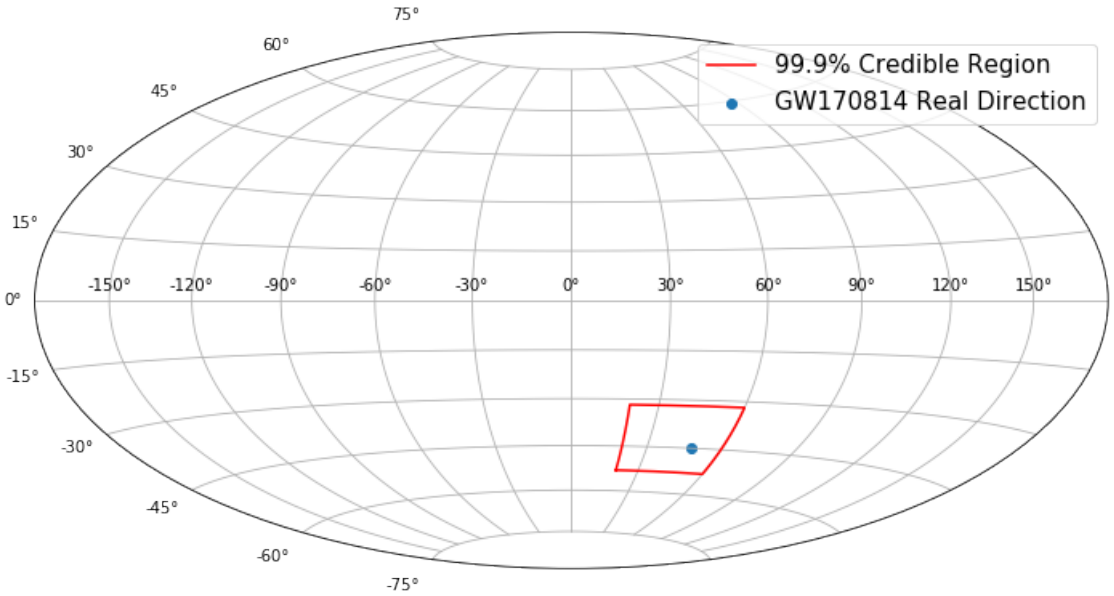

FIGURE 7.7: Blue point is the real direction of GW170814, and the red box is the credible regine of predicted direction.

The area of the red box is about 500 deg^2, when the rapid localization areas calculated by LIGO and Virgo are 1160 deg^2(LIGO data only) and 60 deg^2 (with Virgo data)[16]. Figure 7.8 shows the rapid localization result.

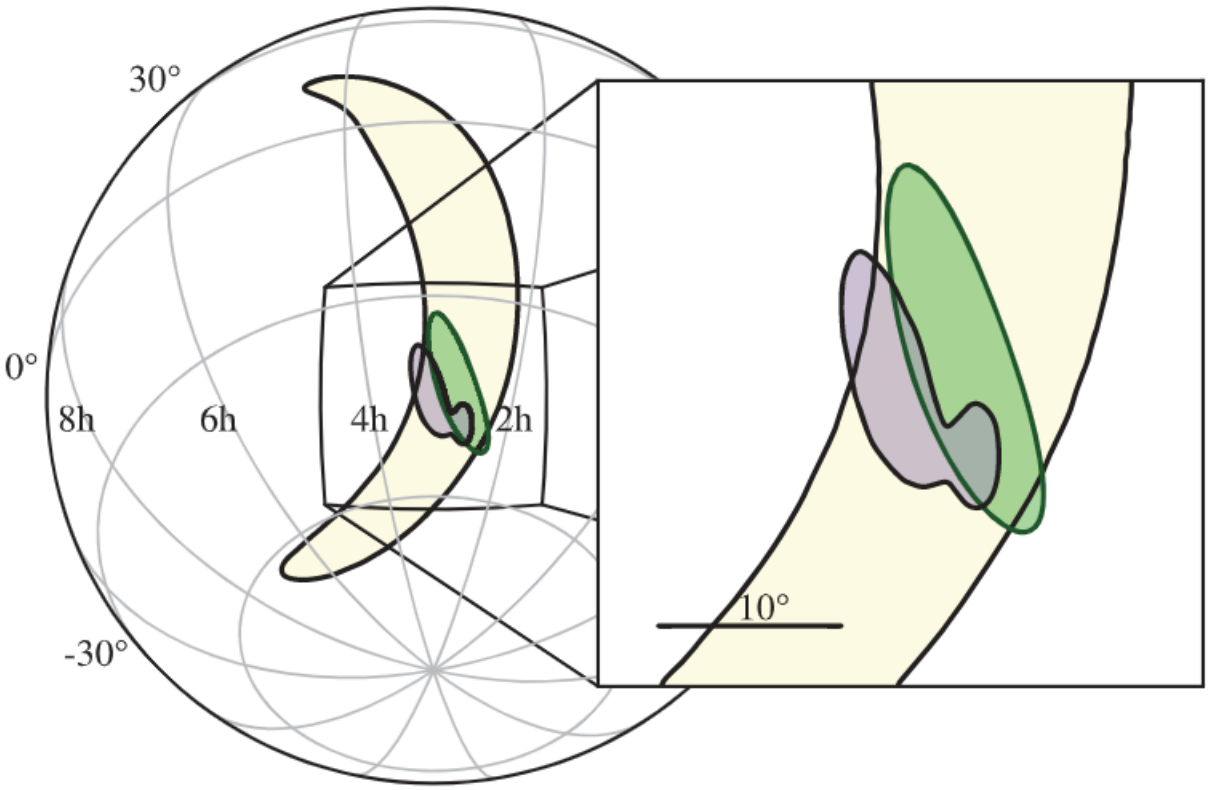

FIGURE 7.8: Localization of GW170814. The rapid localization using data from the two LIGO sites is shown in yellow, with the inclusion of data from Virgo shown in green . The full Bayesian localization is shown in purple. The contours represent the 90% credible regions.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

This thesis discussed GW data analysis research: detecting GW from noise background and localizing GW source. We used Tensorflow and Keras to construct a CNN algorithm to accomplish the analysis. We found that the main restriction of this research is the sensitivities of the detectors. For example, in the localization research, a lower sensitivity detector is not so important to the detector network. It will even hinder high sensitivity detectors from obtaining correct results.

In Chapter 5, we generated the template waveforms depending on varying parameters, such as masses and spins. This template bank contains ten thousand waveforms emit from medium mass BBH mergers, used for GW detection experiment. After the train and test process, the CNN was capable of detecting BBH events up to 600Mpc with an over 90% accuracy.

In Chapter 6, we talked about the localization of GW. We did not only prepare a template bank but also considered the response of GW detectors based on observatories' locations. We conducted two experiments: three-detector network localization(HLV) and four-detector network localization(HLVK), and compared each network's performance. We found that introducing KAGRA to the detector network will indeed benefit the localization, but KAGRA needs to increase its sensitivity to show its value more. Our simulation shows that if the researchers of KAGRA can suppress the noise power to one-tenth of the current status, the four-detector network would have a better performance than a three-detector network.

In Chapter 7, we imported the released data of GW170814 event from LIGO data centre. Using the recorded signal from LIGO Livingston, we successfully distinguished

GW signal from the noise, with an accuracy of 81%. Then we combined the signals in LIGO and Virgo to localize the GW source. As a result, we localized the source in a 500 deg^2 area with an accuracy of over 99%.

CNN Architecture

In this thesis, the CNN structure we constructed was derived from repeated modification and testing. The reason why the hyperparameter set in the hidden layer has such values is for obtaining higher accuracy. Modifying the hyperparameter set can further reduce the calculation time, but this will reduce the model's accuracy.

Algorithm Time Cost and Multi-messenger Astronomy

The most significant advantage of the CNN algorithm is a low time cost. The starting point and focus of this research are to use CNN's low computing cost feature. Since we directly used the GW information as input data, instead of using traditional image recognition methods, time cost was further reduced. Although the specific time cost varies depending on the input data's size, the average cost of a detection calculation is 0.04ms, and the cost of a localization calculation is 0.08ms. From this, we can see that the CNN algorithm has great potential for the realization of multi-messenger astronomy.

8.2 Future Work

Based on our experiments, we found that the CNN algorithm is capable of real-time analysis. However, in this study, because no GW denoising process is included, analysis accuracy was restricted. In the injection experiment described in Chapter 7, we filtered the raw data to remove a part of the noise, and then we were able to obtain a satisfactory result. Thus, we believe that before employing this algorithm for real-time analysis, denoising techniques should be included.

For the localization research, we divided the skymap into 81 sectors, far from the level of search pipelines. Limited by our experimental equipment and the calculation speed of Python, further improvements are difficult to realize. But we believe that by

adopting better hardware and improving our algorithm, we can divide the sky map finer. In other words, more accurate GW source localization can be achieved.

Acknowledgements

This thesis is accomplished with the help of my advisor, my family, and my colleagues. I thank every one gave me valuable support.

I sincerely appreciate my advisor Kentaro Somiya, who always gives my support and advice in these two years. I learned a lot of new knowledge from the conversation with him; his outstanding personality encouraged me to complete my research. Although my research is not very related to his research direction, he did his best to support me. Considering my financial situation, he helped me get the TA and RA jobs to support my living expenses.

I sincerely appreciate Kazuki Sakai from the National Institute of Technology, Nagaoka College. During the week I visited him, he welcomed me and taught me the precious knowledge responsibly. Part of my code ran on a server that he borrowed me generously.

I also sincerely appreciate Takahiro S. Yamamoto from Kyoto University. He accepted Prof. Somiya's invitation and gave an excellent presentation about applying Machine Learning on Gravitational Waves data analysis. His talking inspired me a lot.

I thank the research lecturer Kenichi Harada of Somiya Lab. I learned the basic principles of the interferometer from his speech. He also helped me to solve the problems of coding.

The students in our laboratory also gave me great support and help. I thank Sotatsu Otabe, who always take academic issues very seriously. He explains every detail to me when I have trouble in experiments or theatrical calculations. I thank Hiraku Sasaki, who taught me how to assemble an interferometer hand to hand. He spared his precious time to help me complete the experiments. When I give a report on my research, S. Otabe and H. Sasaki pointed out my shortcomings sharply and gave me great inspiration.

I had a great time with Jun Ogawa, the same grade student as me. We shared the difficulties and gains of research and classes. I determined the direction of employment after our discussion.

Valuable evaluations about my research were proposed from Homare Abe, Makoto Kuribayashi, and Koki Tachihara. We studied about coding skills together in the laboratory. The undergraduate students, Kaido Suzuki and Takanori Suzuki were interested in the research about data analysis and discussed the coding details with me. I

hope someone in our group can carry on my study further.

K. Tachihara and T. Suzuki made an outstanding contribution to the laboratory by making manuals of PyCBC. The manual editing should be my work, but they selflessly shared my responsibility.

I thank the Tech. Assis. Fumiko Miyama and Naomi Maekawa of Somiya Lab. I received their help with paperwork and daily life. They took care of the laboratory members very sincerely.

I also express my thankfulness to the former members of the Somiya group. Thanks to Ryosuke Nakashima, I learned basic knowledge of circuit board soldering. He also introduced the status of employment in Japan. The idea of this research comes from Masahiro Hisatomi's work. Influenced by his research, I decided to work on the data analysis of Gravitational Waves. He also kindly explained his work and code to me before his graduation. Kohei Kusayanagi and the foreign student Melodie Ribes helped me adapt the life in Tokyo Tech.

Finally, I want to thank my parents. They expressed support for my decision to study abroad, although they kept worrying about me, the only child of the family. They are the heroes standing behind me in my lifetime.

Bibliography

[1] A. Einstein. Die grundlage der allgemeinen relativitätstheorie. *Annalen der Physik*, 354(7):769–822, 1916.

[2] Peter Fritschel. Second generation instruments for the laser interferometer gravitational wave observatory (ligo). In *Gravitational-Wave Detection*, volume 4856, pages 282–291. International Society for Optics and Photonics, 2003.

[3] Yoichi Aso, Yuta Michimura, Kentaro Somiya, Masaki Ando, Osamu Miyakawa, Takanori Sekiguchi, Daisuke Tatsumi, and Hiroaki Yamamoto. Interferometer design of the KAGRA gravitational wave detector. *Phys. Rev. D*, 88(4):043007, 2013.

[4] A.C. Green, D.D. Brown, M. Dovale-Álvarez, C. Collins, H. Miao, C. Mow-Lowry, and A. Freise. The Influence of Dual-Recycling on Parametric Instabilities at Advanced LIGO. *Class. Quant. Grav.*, 34(20):205004, 2017.

[5] Yuhang Zhao et al. Frequency-dependent squeezed vacuum source for broadband quantum noise reduction in advanced gravitational-wave detectors. *Physical review letters*, 124 17:171101, 2020.

[6] L. McCuller et al. Frequency-Dependent Squeezing for Advanced LIGO. *Phys. Rev. Lett.*, 124(17):171102, 2020.

[7] Herbert B. Callen and Theodore A. Welton. Irreversibility and generalized noise. *Phys. Rev.*, 83:34–40, Jul 1951.

[8] Harald Dimmelmeier, Jose A. Font, and Ewald Muller. Relativistic simulations of rotational core collapse. 2. Collapse dynamics and gravitational radiation. *Astron. Astrophys.*, 393:523–542, 2002.

[9] Zaven Arzoumanian, Paul T Baker, Adam Brazier, Paul R Brook, Sarah Burke-Spolaor, Bence Becsy, Maria Charisi, Shami Chatterjee, James M Cordes, Neil J

Cornish, et al. Multi-messenger gravitational wave searches with pulsar timing arrays: Application to 3c66b using the nanograv 11-year data set. *arXiv preprint arXiv:2005.07123*, 2020.

[10] CJ Hogan. Gravitational radiation from cosmological phase transitions. *Monthly Notices of the Royal Astronomical Society*, 218(4):629–636, 1986.

[11] B.P. Abbott et al. GW150914: First results from the search for binary black hole coalescence with Advanced LIGO. *Phys. Rev. D*, 93(12):122003, 2016.

[12] B. S. Sathyaprakash and S. V. Dhurandhar. Choice of filters for the detection of gravitational waves from coalescing binaries. *Phys. Rev. D*, 44:3819–3834, Dec 1991.

[13] Duncan A. Brown, Prayush Kumar, and Alexander H. Nitz. Template banks to search for low-mass binary black holes in advanced gravitational-wave detectors. *Phys. Rev. D*, 87:082004, Apr 2013.

[14] Bruce Allen, Warren G. Anderson, Patrick R. Brady, Duncan A. Brown, and Jolien D. E. Creighton. Findchirp: An algorithm for detection of gravitational waves from inspiraling compact binaries. *Phys. Rev. D*, 85:122006, Jun 2012.

[15] Albert Lazzarini. The analysis of ligo data - physics 237b lecture on data analysis.

[16] B. P. Abbott et al. Gw170814: A three-detector observation of gravitational waves from a binary black hole coalescence. *Phys. Rev. Lett.*, 119:141101, Oct 2017.

[17] Stephen Fairhurst. Triangulation of gravitational wave sources with a network of detectors. *New J. Phys.*, 11:123006, 2009. [Erratum: New J.Phys. 13, 069602 (2011)].

[18] Bruce Allen. χ^2 time-frequency discriminator for gravitational wave detection. *Phys. Rev. D*, 71:062001, Mar 2005.

[19] Alexander H. Nitz, Thomas Dent, Tito Dal Canton, Stephen Fairhurst, and Duncan A. Brown. Detecting binary compact-object mergers with gravitational waves: Understanding and improving the sensitivity of the PyCBC search. *The Astrophysical Journal*, 849(2):118, nov 2017.

[20] Tito Dal Canton, Alexander H. Nitz, Andrew P. Lundgren, Alex B. Nielsen, Duncan A. Brown, Thomas Dent, Ian W. Harry, Badri Krishnan, Andrew J. Miller, Karl Wette, Karsten Wiesner, and Joshua L. Willis. Implementing a search for aligned-spin neutron star-black hole systems with advanced ground based gravitational wave detectors. *Phys. Rev. D*, 90:082004, Oct 2014.

[21] Surabhi Sachdev, Sarah Caudill, Heather Fong, Rico KL Lo, Cody Messick, Debnandini Mukherjee, Ryan Magee, Leo Tsukada, Kent Blackburn, Patrick Brady, et al. The gstlal search analysis methods for compact binary mergers in advanced ligo's second and advanced virgo's first observing runs. *arXiv preprint arXiv:1901.08580*, 2019.

[22] B. P. Abbott et al. Gwtc-1: A gravitational-wave transient catalog of compact binary mergers observed by ligo and virgo during the first and second observing runs. *Phys. Rev. X*, 9:031040, Sep 2019.

[23] S Klimenko, I Yakushin, A Mercer, and G Mitselmakher. A coherent method for detection of gravitational wave bursts. *Classical and Quantum Gravity*, 25(11):114029, may 2008.

[24] T. Adams, D. Buskulic, V. Germain, G.M. Guidi, F. Marion, M. Montani, B. Mours, F. Piergiovanni, and G. Wang. Low-latency analysis pipeline for compact binary coalescences in the advanced gravitational wave detector era. *Class. Quant. Grav.*, 33(17):175012, 2016.

[25] Shaun Hooper, Linqing Wen, Chad Hanna, Kipp Cannon, Drew Keppel, David Blair, Shin-Kee Chung, Leo Singer, and Yanbei Chen. Progress on the low-latency inspiral gravitational wave detection algorithm known as SPIIR. *Journal of Physics: Conference Series*, 363:012027, jun 2012.

[26] B.P. Abbott et al. Prospects for Observing and Localizing Gravitational-Wave Transients with Advanced LIGO, Advanced Virgo and KAGRA. *Living Rev. Rel.*, 21(1):3, 2018.

[27] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[28] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[29] Emine Cengil, Ahmet Çinar, and Zafer Güler. A gpu-based convolutional neural network approach for image classification. In *2017 International Artificial Intelligence and Data Processing Symposium (IDAP)*, pages 1–6. IEEE, 2017.

[30] Daniel George and E.A. Huerta. Deep Neural Networks to Enable Real-time Multimessenger Astrophysics. *Phys. Rev. D*, 97(4):044039, 2018.

[31] Chayan Chatterjee, Linqing Wen, Kevin Vinsen, Manoj Kovalam, and Amitava Datta. Using Deep Learning to Localize Gravitational Wave Sources. *Phys. Rev. D*, 100(10):103025, 2019.