SERVIÇO PÚBLICO FEDERAL
UNIVERSIDADE FEDERAL DE SERGIPE
PRÓ-REITORIA DE PÓS-GRADUAÇÃO E PESQUISA

PROGRAMA INSTITUCIONAL DE BOLSAS DE INICIAÇÃO CIENTÍFICA – PIBIC

Aprendizagem estatística de dados astrofísicos

Uso de Deep Learning em sinais de ondas gravitacionais

Período da bolsa: de 08/2020 a 08/2021

Este projeto é desenvolvido com bolsa de iniciação científica

PIBIC/COPES

Sumário

1 Introdução

Descrições algébricas de ondas remontam desde estudos sobre cordas vibrantes através de Jean le Rond d'Alembert (ROCHA et al., 2011). Desse modo, algebricamente, ondas gravitacionais são soluções das equações de campo de Einstein na ausência de fontes – análogo à resolução das equações de Maxwell que produzem a matemática para a luz como uma onda eletromagnética (HARTLE, 2003). Fenomenologicamente, as mesmas são descritas como perturbações na estrutura do espaço-tempo que carregam energia e informações sobre sua origem. Desse modo, analisar eventos dessa magnitude energética proporcionará entendimento do universo no seu primórdio.

Entretanto, o processo crucial dessa busca é filtrar o sinal recebido nos interferômetros, pois, de acordo com (MAGGIORE, 2008), os dados brutos que chegam não contém somente possíveis informações sobre ondas gravitacionais, é necessário uma abordagem estatística no tratamento de dados. Ou seja, essa abordagem explicita limites na qual podemos atribuir uma confiabilidade que o sinal recebido seja realmente um dado proveniente do que tentamos mensurar. Dessa maneira, o grande sucesso da década fora a detecção de ondas gravitacionais provenientes da fusão de buracos negros a partir desse desdobramento estatístico (ABBOTT et al., 2016).

Além disso, o início do segundo milênio popularizou tecnologias que antes eram utilizadas em meios militares, acadêmicos e hospitalares. Mas a grande revolução instalou-se nos lares com o advento da telefonia móvel e dos computadores pessoais. Dada a grande capacidade de resolver problemas, a estrutura de inteligência artificial (IA) ramificou-se para todas as camadas da sociedade – desde estruturação de grande corporações tecnológicas até solução de problemas básicos pessoais. Dessa maneira, a astrofísica – como outras áreas da ciência – importaram a ideia de inteligência artificial como metodologia principal no ataque de problemas na fronteira da ciência (HUERTA et al., 2019).

Utilizando ferramental estatística para resolver a tarefa de fitragem – *matched filtering*, ou filtro de correspondência – fora possível utilizar uma função que maximiza o processo de obtenção do sinal propriamente dito. Sendo assim, com o melhor desenvolvimento de redes neurais nas últimas décadas, o desafio da Astrofísica é terceirizar esse processo intermediário através do uso de redes neurais (GEBHARD et al., 2019). Portanto, o uso de redes neurais convolucionais para trabalhar dados recebidos – séries temporais – molda o objetivo central desse trabalho. Em outras palavras, a procura pelos parâmetros geradores das perturbações, ou seja, as massas dos buracos negros, se dará através de redes neurais. As quais possuem metodologias diversificadas de acordo com a classe de problema proposto (GEORGE; HUERTA, 2018).

2 Objetivos

Objetivamos descobrir parâmetros que geram as perturbações no espaço-tempo. Para isso, alimentamos a rede com amostras que contém sinal e ruído, conforme mostra a figura 1. Desse modo, o objetivo principal é utilizar métodos de *deep learning* para realizar regressão e descobrir parâmetros de amostras ainda desconhecidas. Em outras palavras, a massa reduzida (μ) e a *chirp mass* ($\mathcal{M}$). Por outro lado, podemos apresentar as expressões necessárias para a realização dessa mudança de coordenadas no espaço dos parâmetros.

$$\mathcal{M} = \frac{(m_1 m_2)^{\frac{3}{5}}}{(m_1 + m_2)^{\frac{1}{5}}} \qquad \mu = \frac{(m_1 m_2)}{(m_1 + m_2} \tag{1}$$

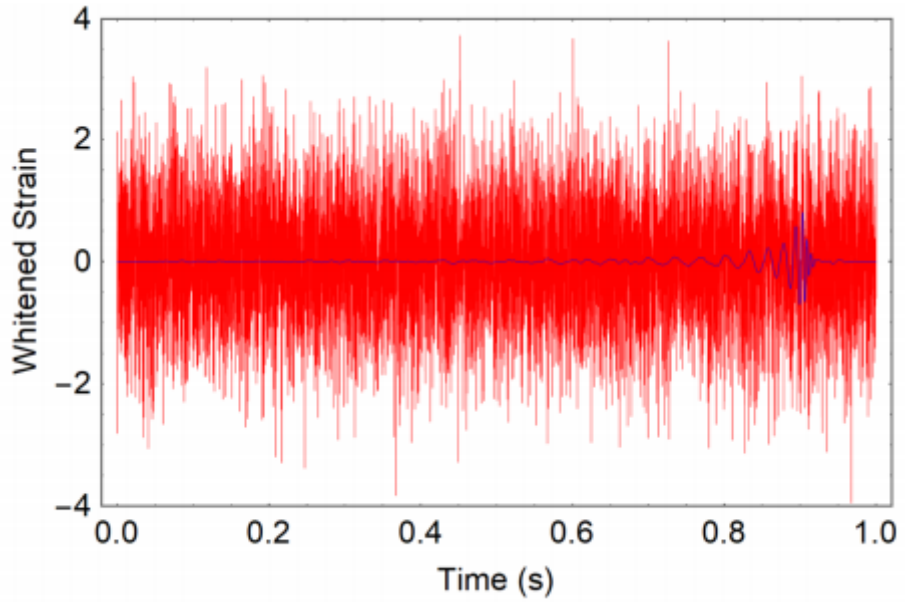

Figura 1: Amostra de sinal injetado na rede (HUERTA et al., 2019).

3 Metodologia

Devido ao cunho teórico do trabalho, a metodologia possui um viés para o desenvolvimento de ferramentas de programação e matemáticas. Dessa maneira, o objeto de pesquisa em questão apresenta dados simulados, em relatividade numérica, de possíveis candidatos a ondas gravitacionais. Por outro lado, para suprir possíveis deficiências na estrutura da rede neural e devido à presença potente de avanços na área de programação, a constante análise de artigos científicos que remetem ao objeto de pesquisa são necessários para o desenvolvimento que supra as necessidades dos objetivos delineados. Ainda na mesma linha, a atribuição de plataformas para implementação de códigos se deu pelo motivo de que as máquinas não possuem *hardware* significativamente potentes para a rodagem das linhas de código. Para conferir esse empecilho, o uso de plataformas como Jupyter Notebook, Google Colaboratory, Kaggle, Overleaf trazem estabilidade no decorrer do trabalho, pois é possível usar provedores de serviço *in cloud* que agilizam execução de estruturas longas. Desse modo, devido à versatilidade e oferta de funcionalidades, as tarefas serão desenvolvidas em um ambiente operacional de distribuição Linux.

Após escolher o banco de dados a ser trabalhado (4431 amostras de treino e 886 amostras de teste), usaremos a biblioteca do Python chamada Pandas, a qual serve para análise e manipulação de dados. Feito isso, o próximo passo se dará para a construção e remodelagem da rede neural através de (CHOLLET, 2017). Randomicamente, separamos os dados de teste e treino em uma proporção, respectivamente de 80% e 20%. Em seguida, definimos os dados de entrada em um só *dataframe* e seus respectivos valores

de massa que serão preditos: *chirp mass* e massa reduzida. Dessa maneira, antes de alimentar a rede, é necessário realizar um processo de *standartization* nos sinais para que a média seja nula e a variância seja unitária. Esse processo é realizado para que exista uma otimização do aprendizado da rede no tocante à queda do gradiente estocástico (CHOLLET, 2017).

Feito a manufatura dos dados, é necessário construir a arquitetura da rede orientada por (HUERTA et al., 2019). Definidos os dados de entrada, ou seja, um vetor com 8192 passos de tempo com suas respectivas características (*chirp mass*, massa reduzida) e como saída uma 2-tupla com os valores preditos. Além disso, a compilação do modelo é uma etapa importante do processo, pois é nela que fora definida a função de perda e o otimizador. Em problemas de regressão, o otimizador utilizado fora o ADAM, função de perda erro absoluto médio e uma métrica chamada erro relativo médio. Todas essas atribuições são importantes para verificar e mensurar se o processo de aprendizado e predição estão apresentados resultados condizentes com a realidade.

A escolha de duração do processo de treinamento fora empírica, ou seja, foram feitos vários processos de treino consecutivos até encontrar um dado valor de época que não provocasse um *overfitting*, ou seja, a rede aprender demasiadamente as características dos dados de treino e desconsiderar a visão macro do problema. Feito isso, o melhor valor encontrado fora 50 épocas e realizado uma divisão de 30 pontos percentuais sobre o conjunto de treino para realização da validação.

A seguir é apresentado o código que ilustra o desenvolvimento do trabalho.

```
# Importando bibliotecas necessárias

import pathlib
import seaborn as sns
from keras import layers, models, optimizers
from sklearn.preprocessing import StandardScaler
import numpy as np
import pandas as pd
import shutil, os, random
import matplotlib.pyplot as plt
from scipy.ndimage import gaussian_filter1d
from tensorflow import keras
import tensorflow as tf

# Variáveis de diretório

data_result = "/content/drive/MyDrive/ic_2021/
    predicoes_chirp_reduced_mass_with_error_50epochs"
train_dataset = "/content/drive/MyDrive/ic_2021/
    dataset_train_effective.csv"
test_dataset = "/content/drive/MyDrive/ic_2021/
    dataset_test_effective.csv"

# Sincronização com o Google Drive

from google.colab import drive
drive.mount('/content/drive')

# Leitura do conjunto de dados .csv

dataset_train = pd.read_csv(train_dataset, sep = ',', header = None
    )
dataset_train = dataset_train.sort_values(by = 1, ascending = False
```

```
    )
dataset_test = pd.read_csv(test_dataset, sep = ',', header = None)
dataset_test = dataset_test.sort_values(by = 1, ascending=False)

# Plotagem gráfica do espaço de parâmetros

fig, axes = plt.subplots(1, 2, figsize = (10,5))
sns.scatterplot(ax = axes[0], x= dataset_m1m2[2], y = dataset_m1m2
    [3], s = 7.5)
axes[0].set(xlabel = "Massa 1", ylabel = "Massa 2")
axes[0].set_title('Espaço dos parâmetros')

sns.scatterplot(ax = axes[1], x= dataset_chirpreduced[8194], y =
    dataset_chirpreduced[8195], s = 7.5)
axes[1].set(xlabel = "Massa chirp", ylabel = "Massa reduzida")
axes[1].set_title('Espaço dos parâmetros')

# Separação dos features e labels no train_dataset
x_train = dataset_train[dataset_train.columns[3:8195]]
y_train = dataset_train[dataset_train.columns[8195:]]

# Separação dos features e labels no test_dataset
x_test = dataset_test[dataset_test.columns[3:8195]]
y_test = dataset_test[dataset_test.columns[8195:]]

#Standardization

from sklearn.preprocessing import StandardScaler

# train the standardization
scaler = StandardScaler()
scaler = scaler.fit(x_train)
# standardization the dataset and print the first 5 rows
normalized_train = scaler.transform(x_train)
for i in range(5):
  print(normalized_train[i])
# inverse transform and print the first 5 rows
inversed = scaler.inverse_transform(normalized_train)
for i in range(5):
  print(inversed[i])

# Construção e estrutura da rede neural

model = models.Sequential()

model.add(layers.Input(shape = (8192, 1)))
model.add(layers.Conv1D(64, kernel_size= 16, strides= 1,
    dilation_rate = 1, activation= None))
model.add(layers.MaxPooling1D(pool_size = 4, strides= 4))
model.add(layers.Dense(64, activation = 'relu'))

model.add(layers.Conv1D(128, kernel_size= 16, strides= 1,
    dilation_rate = 2, activation= None))
model.add(layers.MaxPooling1D(pool_size = 4, strides= 4))
model.add(layers.Dense(128, activation = 'relu'))

model.add(layers.Conv1D(256, kernel_size= 16, strides= 1,
    dilation_rate = 2, activation = None))
```

```
model.add(layers.MaxPooling1D(pool_size = 4, strides= 4))
model.add(layers.Dense(256, activation = 'relu'))

model.add(layers.Conv1D(512, kernel_size= 32, strides= 1,
    dilation_rate=  2, activation= 'relu'))
model.add(layers.MaxPooling1D(pool_size = 4, strides= 4))
model.add(layers.Dense(512, activation = 'relu'))

model.add(layers.Flatten())
model.add(layers.Dense(128, activation= 'linear'))
model.add(layers.Dense(128, activation= 'relu'))
model.add(layers.Dense(64, activation= 'linear'))
model.add(layers.Dense(64, activation  ='relu'))

model.add(layers.Dense(2, activation= 'linear'))

# Sumário da rede neural

model.summary()

# Compilação do modelo

optimizer_adam = keras.optimizers.Adam(learning_rate=0.0001,
    amsgrad = True)
model.compile(loss = 'mean_absolute_error',  optimizer =
    optimizer_adam, metrics = 'mean_absolute_percentage_error')

# Reshape dados de treino

scaler_test = StandardScaler()
scaler_test = scaler_test.fit(x_test)
standard_test = scaler_test.transform(x_test)
x_pred_reshaped = standard_test.reshape(886, 8192, 1)
data_y = (np.array(y_test)).reshape(3545, 2, 1)

# # Treinando o modelo
history = model.fit(data_x, data_y, epochs = 50, validation_split=
    0.3, batch_size = 32, verbose = 1)

# Plotagem das métricas de treino e validação

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Função de perda')
plt.ylabel('Erro absoluto médio')
plt.xlabel('Épocas')
plt.legend(['Treino', 'Validação'], loc='upper left')
plt.show()

plt.plot(history.history['mean_absolute_percentage_error'])
plt.plot(history.history['val_mean_absolute_percentage_error'])
plt.title('Métrica')
plt.ylabel('Erro relativo médio')
plt.xlabel('Épocas')
plt.legend(['Treino', 'Validação'], loc='upper left')
plt.show()

# Predição
```

```
136
137 predicao = model.predict(x = x_pred_reshaped, verbose = 1)
```

4 Resultados e discussões

Executado o processo de treino, a etapa de teste é análoga. Nesse momento, a rede já possui uma distribuição de pesos em seus parâmetros aos quais dá sustentação do aprendizado da mesma de acordo como apresentado pelas métricas. Nesse momento, tomaremos uma regressão sob os parâmetros dos sinais $(\mathcal{M}, \mu)$.

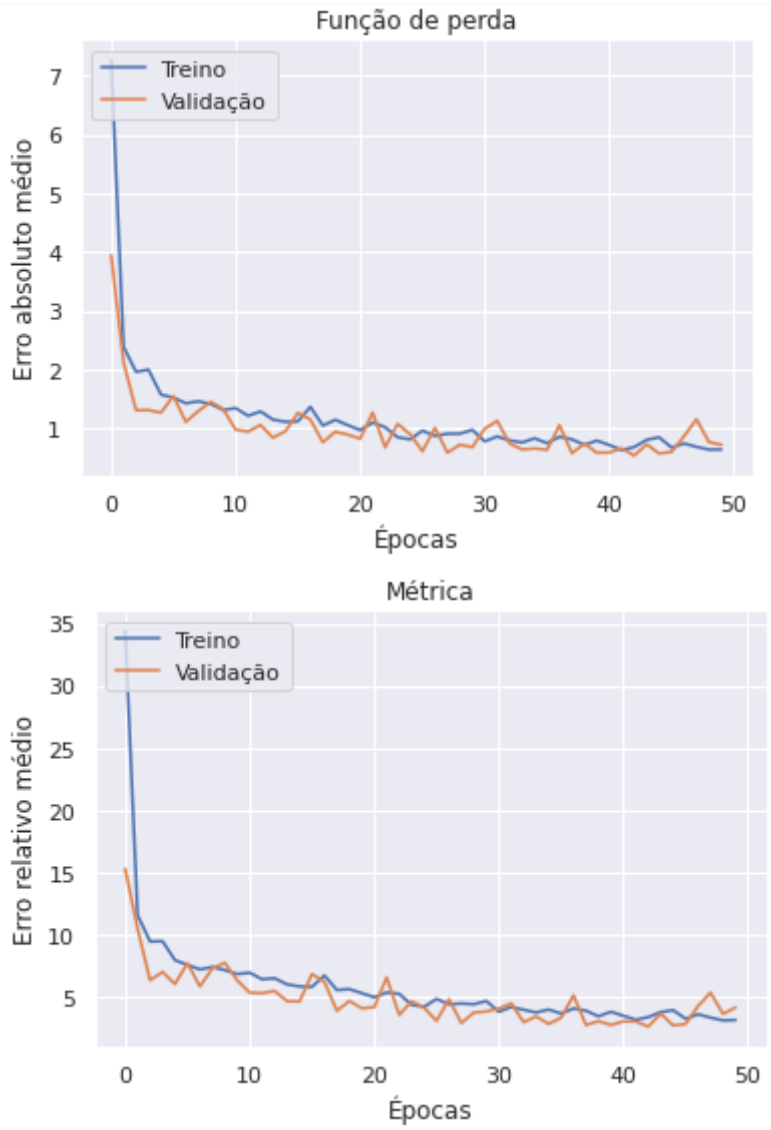

Figura 2: Métricas da etapa de treino e validação.

A figura acima apresenta informações sobre o aprendizado da rede em suas respectivas épocas. Enquanto o erro absoluto e relativo médio relacionados ao treino não divergirem dos valores da etapa de validação a rede estará aprendendo e evitando o *overfitting*. Empiricamente, escolhemos o quinquagésima época também como meio de limitação de *hardware* – as células do GoogleColab possuem intervalo de tempo fixo de execução.

Realizada a predição, a rede retornou uma 2-tupla com 886 linhas. Em seguida, plotamos gráficos de regressão comparando valores teóricos e preditos bem como uma

relação entra a métrica de erro relativo médio e a taxa de sinal-ruído (SNR) (HUERTA et al., 2019), (KRASTEV et al., 2021).

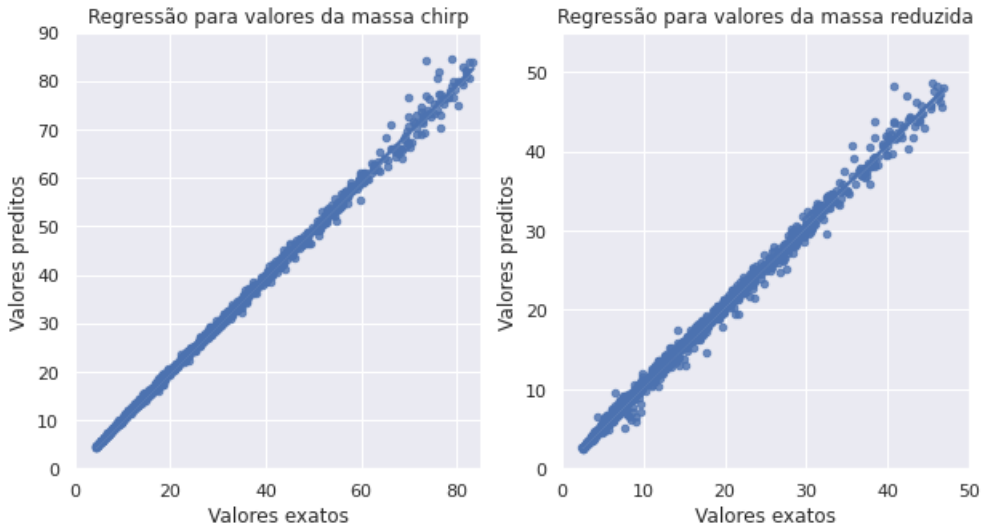

Figura 3: Regressão linear entre valores preditos e teóricos da *chirp mass* e massa reduzida.

Como em toda regressão linear, se a disposição de dados reflete uma função afim com coeficiente linear nulo, a predição fora satisfatória. Em relação a figura acima, percebemos que a distribuição para valores pequenos estão bem próximos de uma reta – confirmando o caráter satisfatório do aprendizado da rede. Vale ressaltar que, para valores > 80 a distribuição passa a dispersar, indicando que a rede ainda não possui informações suficientes para predizer amostras com altos valores em seus parâmetros. Portanto, o gráfico acima confirma que a rede convolucional treinada consegue predizer os valores. No entanto, fora necessário analisar os resultados para a resolução do problema de mudança de coordenadas, ou seja, se a rede trabalha melhor nas coordenadas utilizadas no presente trabalho.

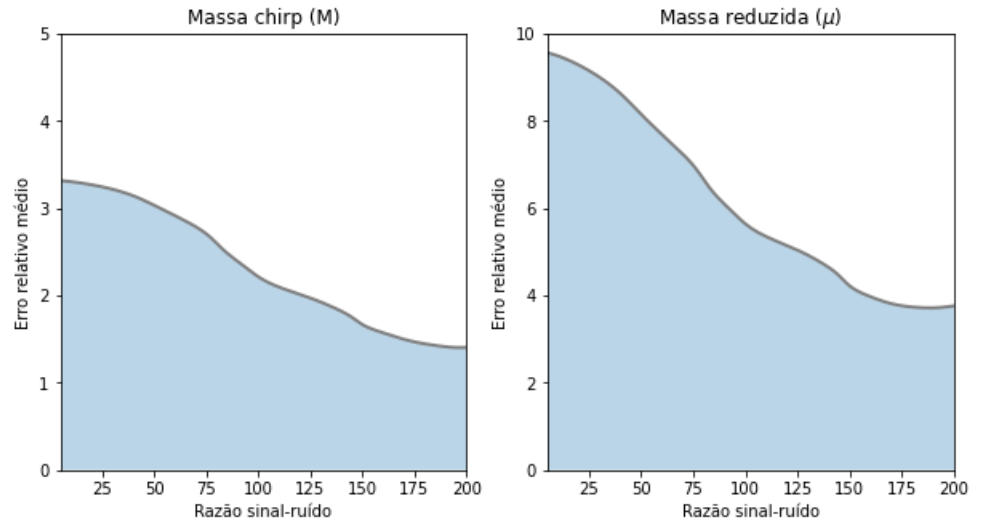

Figura 4: Erro relativo médio dos dados preditos em relação a baixo SNR (*signal noise ratio*).

A figura 4 reforça o argumento utilizado anteriormente, ou seja, de que a rede teve sucesso ao aprender e obter a predição dos parâmetros. A taxa de sinal-ruído do conjunto de dados é de 0 a 1.000, mas a figura acima tem o intuito de comparação com a literatura atual — analisa o erro médio para os primeiros valores de SNR para apresentação de um comportamento inversamente proporcional dessas grandezas (HUERTA et al., 2019), (KRASTEV et al., 2021). Além disso, a magnitude do erro relativo médio encontrado nesse trabalho é inferior aos resultados encontrados na literatura. Logo, a mudança de coordenadas executada é de ótimo proveito para otimização de resultados.

Por outro lado, realizando uma análise macro dos resultados, obtivemos um comportamento anômalo nos valores intermediários de SNR.

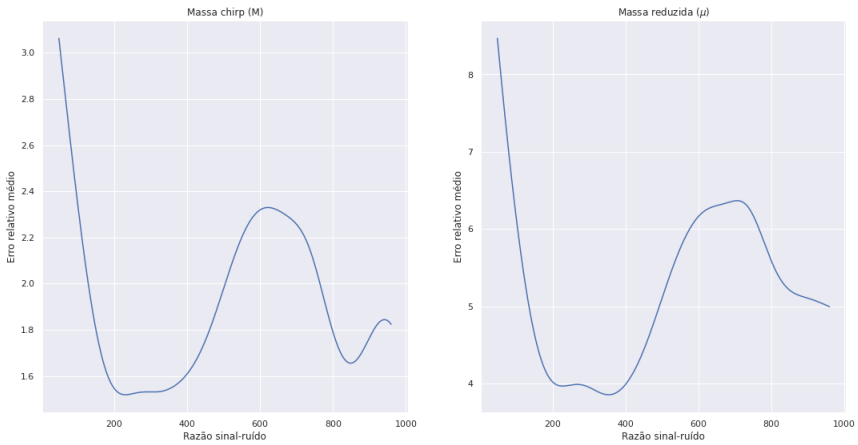

Figura 5: Erro relativo médio dos dados preditos em relação ao SNR.

O comportamento anômalo na figura acima diz respeito a um aumento do erro relativo médio na região intermediária do SNR. Isso pode ser explicado pela ausência de amostras com o mesmo valor de SNR na etapa de teste.

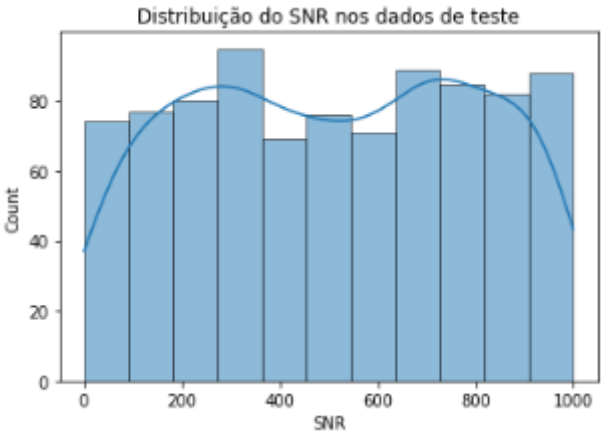

Figura 6: Distribuição de amostras de teste em relação ao SNR.

Verificando a distribuição de dados em torno do intervalo de 400 e 800, é perceptível que há uma ausência de amostras. Para solução de tal anomalia, e com uma nova remessa de dados, a distribuição — além de randômica — deve atribuir um grau de homogeneidade em relação ao SNR. Além disso, aumentar a magnitude de dados também deve solucionar tal empecilho.

5 Perspectivas de futuros trabalhos

Portanto, redes convolucionais são ótimas alternativas para análise de parâmetros e a qualidade dos dados é um fator vital para o desenvolvimento de qualquer projeto voltado à aplicação de métodos de aprendizado profundo.

Devido ao baixo número de amostras analisadas – por motivos de limitação de *hardware* –, para futuros trabalhos, o objetivo é englobar uma quantidade maior de dados para que a rede abranja diversos tipos de sinais. Além disso, isso reforçará os resultados obtidos na seção anterior, ou seja, existe um sistema de coordenadas onde a predição é mais exata. Logo, além de abranger a quantidade de dados, a qualidade será relevante, ou seja, usar dados realísticos disponibilizados pelo LIGO, Virgo, etc.

6 Outras atividades

No referente período de vigência do plano de trabalho, o bolsista compareceu aos seminários como espectador e apresentador do grupo de pesquisa de Astrofísica do Departamento de Físíca (DFI-UFS) às terças-feiras numa periodicidade quinzenal. Além disso, executou o estágio obrigatório e as disciplinas da grade curricular.

Referências

ABBOTT, B. P. et al. Observation of gravitational waves from a binary black hole merger. *Phys. Rev. Lett.*, American Physical Society, v. 116, p. 061102, Feb 2016. Disponível em: <https://link.aps.org/doi/10.1103/PhysRevLett.116.061102>. page.33

CHOLLET, F. *Deep Learning with Python*. 1st. ed. USA: Manning Publications Co., 2017. ISBN 1617294438. page.44, page.55

GEBHARD, T. D. et al. Convolutional neural networks: A magic bullet for gravitational-wave detection? *Phys. Rev. D*, American Physical Society, v. 100, p. 063015, Sep 2019. Disponível em: <https://link.aps.org/doi/10.1103/PhysRevD.100.063015>. page.33

GEORGE, D.; HUERTA, E. A. Deep neural networks to enable real-time multimessenger astrophysics. *Phys. Rev. D*, American Physical Society, v. 97, p. 044039, Feb 2018. Disponível em: <https://link.aps.org/doi/10.1103/PhysRevD.97.044039>. page.33

HARTLE, J. *Gravity: An Introduction to Einstein's General Relativity*. Addison-Wesley, 2003. ISBN 9780805386622. Disponível em: <https://books.google.com.br/books?id=ZHgpAQAAMAAJ>. page.33

HUERTA, E. A. et al. Enabling real-time multi-messenger astrophysics discoveries with deep learning. *Nature Reviews Physics*, v. 1, n. 10, p. 600–608, Oct 2019. ISSN 2522-5820. Disponível em: <https://doi.org/10.1038/s42254-019-0097-4>. page.33, page.44, page.55, page.99, page.1010

KRASTEV, P. G. et al. Detection and parameter estimation of gravitational waves from binary neutron-star mergers in real ligo data using deep learning. *Physics Letters B*, v. 815, p. 136161, 2021. ISSN 0370-2693. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0370269321001015>. page.99, page.1010

MAGGIORE, M. *Gravitational Waves: Volume 1: Theory and Experiments*. OUP Oxford, 2008. (Gravitational Waves). ISBN 9780198570745. Disponível em: <https://books.google.com.br/books?id=AqVpQgAACAAJ>. page.33

ROCHA, J. F. M. et al. *Origens e evolução das idéias da física*. [S.l.]: EDUFBA, 2011. page.33