Gravitational Wave Event Detection: Developing Convolutional Neural Networks and Recurrent Neural Networks for Gravitational Wave Data Analysis

Shufan Dong[1]

[1]Class of 2026, Bronx High School of Science, NY, USA.
dongs1@bxscience.edu

Abstract

Gravitational wave (GW) data analysis has evolved significantly with advancements in machine learning (ML) techniques, particularly convolutional neural networks (CNNs) and Recurrent neural networks (RNNs). This paper presents a comprehensive approach to developing both CNN and RNN models for the analysis of GW data. We convert the time-series strain data into spectrograms for the 2D CNN while retaining the time-series format for the 1D CNN and RNNs (LSTM and GRU), allowing the input of both data representations into the ML models. The overall procedure for training the 1D CNN and RNN models includes data segmentation, time-series data reshaping, data augmentation, model training, and model evaluation and visualization. Then, a similar process is applied to training the 2D CNN model, which includes data segmentation, spectrogram data generation and reshaping, data augmentation, model training, and model evaluation and visualization. Our results demonstrate the use of these methods in accurately classifying GW events, highlighting the potential for further applications in astrophysical research.

Contents

1 Introduction **2**

2 Importing Libraries **2**

3 Data Segmentation and Labeling **3**

4 Data Preparation **5**
4.1 Time-series Data Reshaping for 1D CNN and RNNs 5
4.2 Spectrogram Generation for 2D CNN 5

5 Data Augmentation **6**

6 Model Training **7**
6.1 1D CNN 7
6.2 2D CNN 7
6.3 LSTM 8
6.4 GRU 9

7 Visualization and Evaluation **10**
7.1 1D CNN 11
7.2 2D CNN 11
7.3 LSTM 12
7.4 GRU 12

8 Results and Discussion **13**

9 Conclusion and Outlook **13**

1 Introduction

The detection and analysis of gravitational waves (GWs) have opened a new window into the universe, allowing us to observe astrophysical phenomena that were previously inaccessible. The Laser Interferometer Gravitational-Wave Observatory (LIGO) and its international partners have made continuous efforts in this field, detecting numerous GW events from merging black holes and colliding neutron stars. As the volume of GW data increases, there is a growing need for efficient and accurate analysis techniques. This paper explores the use of CNNs and RNNs for classification purposes, focusing on 1D CNN, 2D CNN, LSTM, and GRU models. The preprocessing of GW data was done previously already, and you can refer to [49] for more details.

2 Importing Libraries

Necessary libraries are required for the CNN and RNN models in this paper, utilizing the TensorFlow tool for modeling our CNNs and RNNs.

```
import tensorflow as tf
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Conv1D, MaxPooling1D, Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from sklearn.model_selection import train_test_split

import warnings
warnings.filterwarnings('ignore')
# Set tf logging level to suppress warnings and info messages
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
# This ensures that the logging level is set before any tf code runs
tf.get_logger().setLevel('ERROR')
```

Figure 1: The packages imported are added in addition to the imported libraries discussed in [49], which contains the basic libraries required to preprocess and manipulate GW data.

3 Data Segmentation and Labeling

The continuous GW strain data is split into smaller, manageable segments and labeled appropriately. This step is critical for preparing the dataset for supervised learning, allowing the model to learn based on discoverable patterns.

```python
def create_segments_and_labels(strain, event_time, window_size, sample_rate):
    # Resample strain to desired sample rate (if necessary)
    strain = strain.resample(sample_rate)

    # Def segments and labels ls
    segments = []
    labels = []

    # Calc # of samples per segment
    segment_length = int(window_size * sample_rate)

    # Create segments and labels
    for i in range(0, len(strain) - segment_length, segment_length):
        segment = strain[i:i + segment_length]
        segments.append(segment.value)

        # Label based on event presence
        if segment.times.value[0] <= event_time <= segment.times.value[-1]:
            labels.append(1)  # Event present
        else:
            labels.append(0)  # No event

    # Convert to np arrays
    segments = np.array(segments)
    labels = np.array(labels)

    return segments, labels

segments, labels = create_segments_and_labels(strain, t_start, 2, fs)
```

Figure 2: The function `create_segments_and_labels` is used to split the strain data into segments of 2 seconds each, starting at `t_start` (start of GW150914 event) and sampled at `fs` Hz (4096 Hz).

```
Segments shape: (2047, 8192)
Labels shape: (2047,)
```

Figure 3: The shape of GW data's segments and labels.

4 Data Preparation

4.1 Time-series Data Reshaping for 1D CNN and RNNs

To ensure the compatibility of the time-series data for 1D CNN and RNNs, time-series data is reshaped to include an extra dimension.

```
# Reshape segments for 1D CNN
segments = segments.reshape((segments.shape[0], segments.shape[1], 1))

# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(segments, labels, test_size=0.2, random_state=42)
```

Figure 4: The segment data is reshaped with an additional dimension of 1. Then, the data is split into the training set (80% of the data) and the testing set (20% of the data).

```
Reshaped segments shape: (2047, 8192, 1)
```

Figure 5: The shape of the input data for 1D CNN and RNNs.

4.2 Spectrogram Generation for 2D CNN

To examine the spatial feature extraction capabilities of 2D CNN, time-series data is converted into spectrograms, which provide a frequency domain representation of the data.

```
# Generate spectrograms for each segment
def generate_spectrogram(segment, sample_rate):
    f, t, Sxx = spectrogram(segment, sample_rate)
    return Sxx

spectrograms = np.array([generate_spectrogram(segment, fs) for segment in segments])

# Reshape spectrograms for 2D CNN
spectrograms = spectrograms[..., np.newaxis]  # Add a channel dim

# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(spectrograms, labels, test_size=0.2, random_state=42)
```

Figure 6: The `generate_spectrogram` function converts each time-series segment into a spectrogram, and the spectrograms are then reshaped to include a channel dimension for compatibility with 2D CNN input. Then, the data is split into the training set (80% of the data) and testing set (20% of the data).

```
Reshaped spectrograms shape: (2047, 129, 36, 1)
```

Figure 7: The shape of the input data for 2D CNN.

5 Data Augmentation

To prevent overfitting and improve generalization, data augmentation techniques are applied to the training data.

```python
def augment_data(data, labels):
    augmented_data = []
    augmented_labels = []
    for d, l in zip(data, labels):
        augmented_data.append(d)
        augmented_labels.append(l)
        augmented_data.append(np.flip(d, axis=0))
        augmented_labels.append(l)
        noise = np.random.normal(0, 0.1, d.shape)
        augmented_data.append(d + noise)
        augmented_labels.append(l)
    return np.array(augmented_data), np.array(augmented_labels)

X_train_aug, y_train_aug = augment_data(X_train, y_train)
```

Figure 8: The `augment_data` function artificially increases the size of the training dataset by introducing variability.

```
Original training data shape: (1637, 8192, 1)
Augmented training data shape: (4911, 8192, 1)
```

Figure 9: The shape of the data for 1D CNN and RNNs before and after augmentation.

```
Original training data shape: (1637, 129, 36, 1)
Augmented training data shape: (4911, 129, 36, 1)
```

Figure 10: The shape of the data for 2D CNN before and after augmentation.

6 Model Training

6.1 1D CNN

A 1D CNN model is constructed and trained on the augmented time-series data.

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 8190, 16)	64
max_pooling1d (MaxPooling1D)	(None, 4095, 16)	0
conv1d_1 (Conv1D)	(None, 4093, 32)	1568
max_pooling1d_1 (MaxPooling1D)	(None, 2046, 32)	0
flatten_1 (Flatten)	(None, 65472)	0
dense_2 (Dense)	(None, 64)	4190272
dropout_1 (Dropout)	(None, 64)	0
dense_3 (Dense)	(None, 1)	65

Figure 11: The 1D CNN model processes the time-series data directly, using convolutional layers to extract temporal features, pooling layers to reduce dimensionality, dense layers to classify event presence, and a dropout layer to prevent overfitting.

6.2 2D CNN

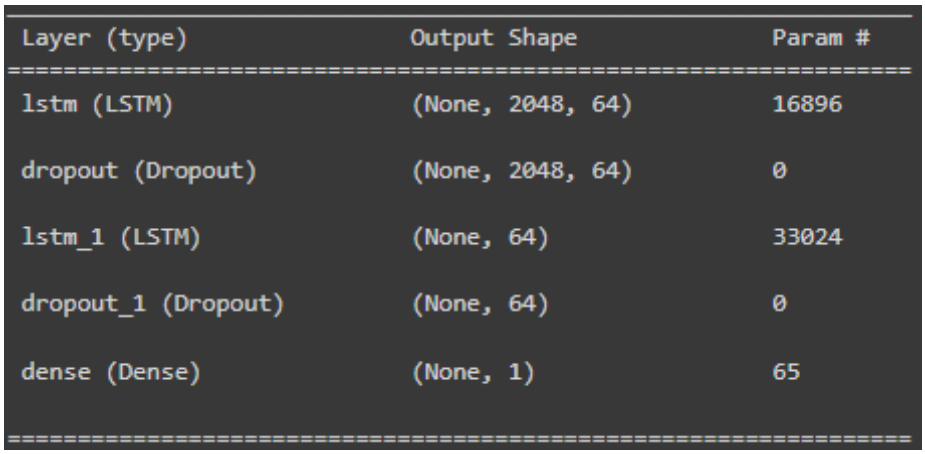
A 2D CNN model is built and trained on the augmented spectrogram data.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 127, 34, 16)	160
max_pooling2d (MaxPooling2D)	(None, 63, 17, 16)	0
conv2d_1 (Conv2D)	(None, 61, 15, 32)	4640
max_pooling2d_1 (MaxPooling2D)	(None, 30, 7, 32)	0
flatten (Flatten)	(None, 6720)	0
dense (Dense)	(None, 64)	430144
dropout (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 1)	65

Figure 12: The 2D CNN model consists of convolutional layers for feature extraction, pooling layers for dimensionality reduction, and dense layers for event classification. A dropout layer is added to help prevent overfitting.

6.3 LSTM

An LSTM model is constructed and trained on the augmented time-series data.



Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 2048, 64)	16896
dropout (Dropout)	(None, 2048, 64)	0
lstm_1 (LSTM)	(None, 64)	33024
dropout_1 (Dropout)	(None, 64)	0
dense (Dense)	(None, 1)	65

Figure 13: The LSTM model processes the time-series data directly, using two LSTM layers for feature extraction, two dropout layers to prevent overfitting, and a dense layer to classify event presence. For quicker model training, the size of the data for LSTM is resampled to four times less than the data for 1D and 2D CNN

6.4 GRU

A GRU model is constructed and trained on the augmented time-series data.

Layer (type)	Output Shape	Param #
gru (GRU)	(None, 2048, 64)	12864
dropout_2 (Dropout)	(None, 2048, 64)	0
gru_1 (GRU)	(None, 64)	24960
dropout_3 (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 1)	65

Figure 14: The GRU model processes the time-series data directly, using two GRU layers for feature extraction, two dropout layers to prevent overfitting, and a dense layer to classify event presence. For quicker model training, the size of the data for LSTM is resampled to four times less than the data for 1D and 2D CNN

7 Visualization and Evaluation

To examine the CNNs and RNNs more closely, The training and validation loss and accuracy are plotted to visualize the training process over epochs and determine the performance of each model.

7.1 1D CNN

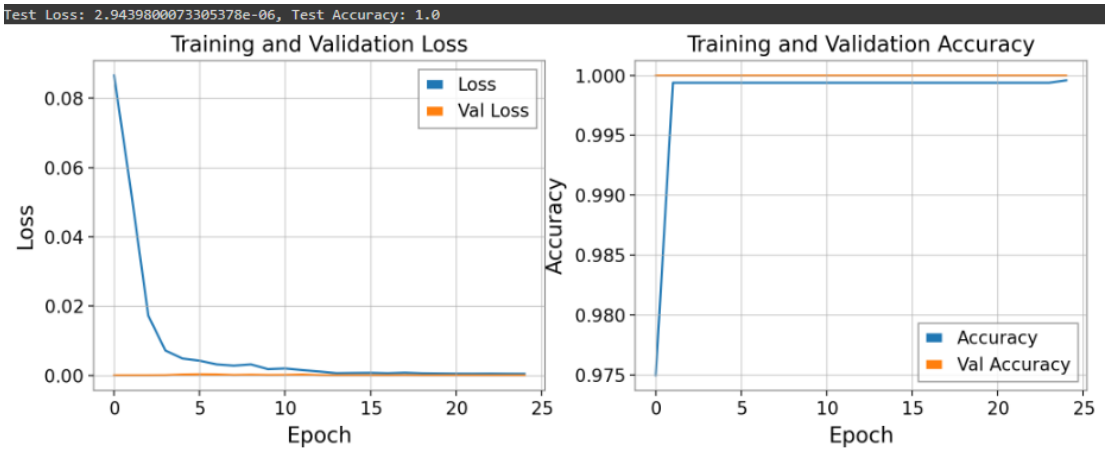


Figure 15: These plots show the training history of the 1D CNN, including the test loss and accuracy evaluation.

7.2 2D CNN

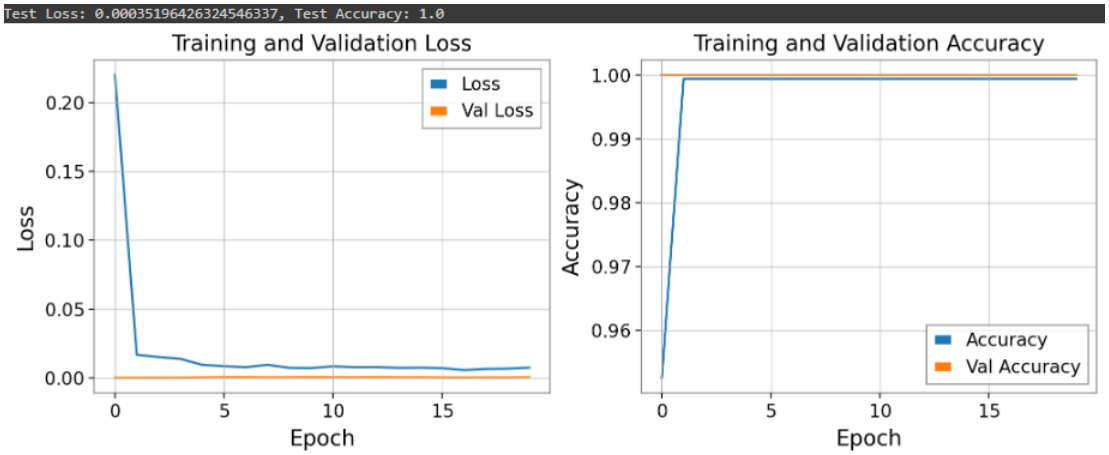


Figure 16: These plots show the training history of the 2D CNN, including the test loss and accuracy evaluation.

7.3 LSTM

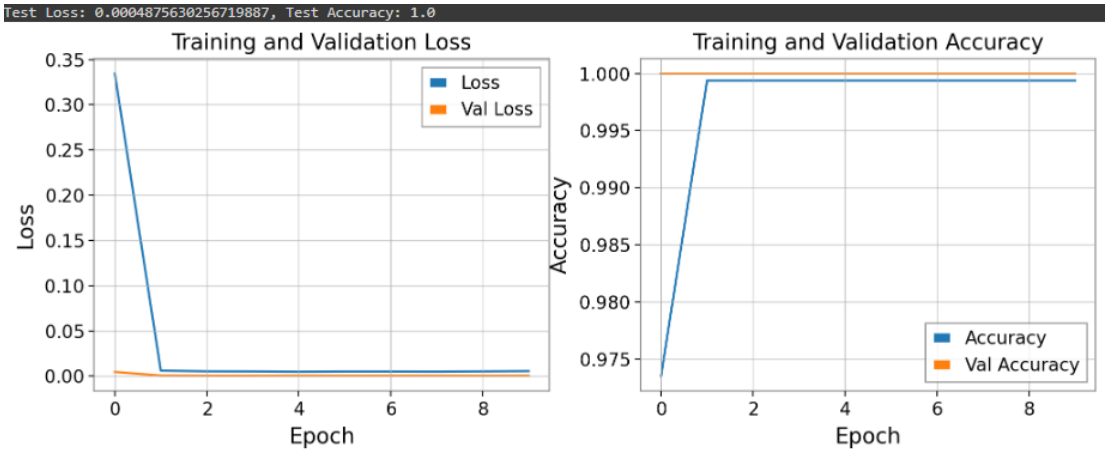


Figure 17: These plots show the training history of the LSTM, including the test loss and accuracy evaluation.

7.4 GRU

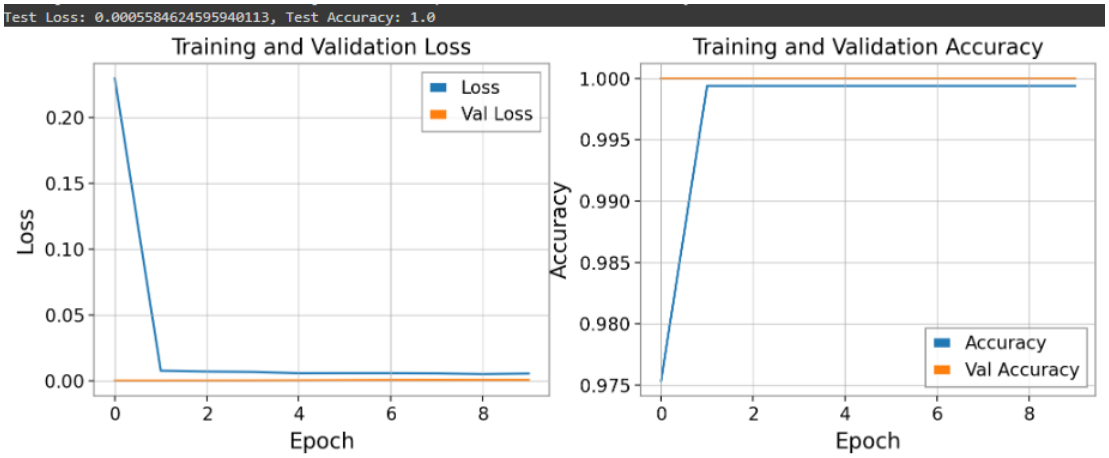


Figure 18: These plots show the training history of the GRU, including the test loss and accuracy evaluation.

8 Results and Discussion

Both CNN and RNN models showed promising results in classifying GW data based on event presence. The 1D CNN and RNNs effectively captured temporal patterns directly from the reshaped time-series data, while the 2D CNN used the frequency domain data representation provided by spectrograms. The models were evaluated based on test accuracy and loss, demonstrating their application in real-time GW event detection.

9 Conclusion and Outlook

This study highlights the effectiveness of CNNs and RNNs in analyzing GW data, with both 1D and 2D CNNs, LSTM, and GRU producing satisfactory results. Future work will involve exploring more advanced ML models applicable to the means of detecting GW event presence. The integration of these ML techniques into the GW analysis will enhance our ability to detect and interpret these cosmic events, contributing to our understanding of the universe and the future outlook of astrophysical research.

References

[1] Aasi, J., et al. (2015). Advanced LIGO. Classical and Quantum Gravity, 32(7), 074001.

[2] Abbott, B. P., et al. (2016). Observation of gravitational waves from a binary black hole merger. Physical Review Letters, 116(6), 061102.

[3] Abbott, B. P., et al. (2016). Properties of the binary black hole merger GW150914. Physical Review Letters, 116(24), 241102.

[4] Abbott, B. P., et al. (2016). Tests of general relativity with GW150914. Physical Review Letters, 116(22), 221101.

[5] Abbott, B. P., et al. (2016). Astrophysical implications of the binary black hole merger GW150914. The Astrophysical Journal Letters, 818(2), L22.

[6] Abbott, B. P., et al. (2017). GW170817: Observation of gravitational waves from a binary neutron star inspiral. Physical Review Letters, 119(16), 161101.

[7] Abbott, B. P., et al. (2017). Multi-messenger observations of a binary neutron star merger. The Astrophysical Journal Letters, 848(2), L12.

[8] Abbott, B. P., et al. (2019). A gravitational-wave standard siren measurement of the Hubble constant. Nature, 551(7678), 85-88.

[9] Abbott, B. P., et al. (2020). GWTC-2: Compact Binary Coalescences Observed by LIGO and Virgo During the First Half of the Third Observing Run. arXiv preprint arXiv:2010.14527.

[10] Abbott, B. P., et al. (2018). GWTC-1: A gravitational-wave transient catalog of compact binary mergers observed by LIGO and Virgo during the first and second observing runs. Physical Review X, 9(3), 031040.

[11] Abbott, R., et al. (2021). GWTC-3: Compact Binary Coalescences Observed by LIGO and Virgo During the Second Half of the Third Observing Run. arXiv preprint arXiv:2111.03606.

[12] Acernese, F., et al. (2015). Advanced Virgo: a second-generation interferometric gravitational wave detector. Classical and Quantum Gravity, 32(2), 024001.

[13] Acernese, F., et al. (2007). Status of Virgo. Classical and Quantum Gravity, 24(19), S381.

[14] Amaro-Seoane, P., et al. (2017). Laser Interferometer Space Antenna. arXiv preprint arXiv:1702.00786.

[15] Chatziioannou, K., et al. (2017). The last gravitational wave in the window: An improved waveform model for binary black hole inspirals. Physical Review D, 95(10), 104027.

[16] Cutler, C., & Thorne, K. S. (2002). An overview of gravitational-wave sources. General Relativity and Gravitation, 39(5), 151-165.

[17] Fairhurst, S. (2011). Source localization with an advanced gravitational wave detector network. Classical and Quantum Gravity, 28(10), 105021.

[18] Finn, L. S., & Chernoff, D. F. (1993). Observing binary inspiral in gravitational radiation: One interferometer. Physical Review D, 47(6), 2198.

[19] LIGO: The Laser Interferometer Gravitational-Wave Observatory. arXiv. 2007;0711.3041.

[20] Magee, R., et al. (2021). First demonstration of early warning gravitational wave alerts. Astrophys J Lett. 910(2).

[21] Martynov, D. V., et al. (2016). Sensitivity of the Advanced LIGO detectors at the beginning of gravitational wave astronomy. Physical Review D, 93(11), 112004.

[22] Parameter estimation with gravitational waves. Rev Mod Phys. 2022;94:025001.

[23] Polarization-based tests of gravity with the stochastic gravitational-wave background. Phys Rev X. 2017;7:041058.

[24] Sachdev, S., et al. (2020). An early-warning system for electromagnetic follow-up of gravitational-wave events. Astrophys J Lett. 905(2).

[25] Schutz, B. F. (2011). Networks of gravitational wave detectors and three figures of merit. Classical and Quantum Gravity, 28(12), 125023.

[26] Singer, L. P., et al. (2014). The first two years of electromagnetic follow-up with advanced LIGO and Virgo. The Astrophysical Journal, 795(2), 105.

[27] The LIGO Scientific Collaboration, the Virgo Collaboration, Abbott, R., et al. (2020). GWTC-2: Compact Binary Coalescences Observed by LIGO and Virgo During the First Half of the Third Observing Run. arXiv preprint arXiv:2010.14527.

[28] The LIGO Scientific Collaboration, the Virgo Collaboration, Abbott, R., et al. (2021). GWTC-3: Compact Binary Coalescences Observed by LIGO and Virgo During the Second Half of the Third Observing Run. arXiv preprint arXiv:2111.03606.

[29] What is LIGO? LIGO Lab, Caltech. Caltech. 2023.

[30] Abbott, R., et al. (2021). Population properties of compact objects from the second LIGO-Virgo gravitational-wave transient catalog. Astrophys J Lett. 913:7.

[31] Abbott, R., et al. (2021). Tests of general relativity with binary black holes from the second LIGO-Virgo gravitational-wave transient catalog. Phys Rev D. 103:122002.

[32] Abbott, R., et al. (2021). Upper limits on the isotropic gravitational-wave background from Advanced LIGO's and Advanced Virgo's third observing run. arXiv. 2101.12130.

[33] Abbott, R., et al. (2021). Searches for continuous gravitational waves from young supernova remnants in the early third observing run of Advanced LIGO and Virgo. arXiv. 2101.12130.

[34] Abbott, B. P., et al. (2016). Improved analysis of GW150914 using a fully spin-precessing waveform model. Physical Review X, 6(4), 041014.

[35] Abbott, B. P., et al. (2017). Multi-messenger observations of a binary neutron star merger. The Astrophysical Journal Letters, 848(2), L12.

[36] Finn, L. S., & Chernoff, D. F. (1993). Observing binary inspiral in gravitational radiation: One interferometer. Physical Review D, 47(6), 2198.

[37] Fortifying gravitational-wave tests of general relativity against astrophysical assumptions. Phys Rev D. 2023;108:124060.

[38] GW170817: Observation of gravitational waves from a binary neutron star inspiral. arXiv. 2017;1710.05832.

[39] Machine learning for gravitational-wave astronomy: Methods and applications for high-dimensional laser interferometry data. Academic Commons. 2021.

[40] Bai, S., Kolter, J. Z., Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv preprint arXiv:1803.01271.

[41] Dosovitskiy, A., Springenberg, J. T., Brox, T. (2015). Learning to generate chairs with convolutional neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 1538-1546).

[42] Goodfellow, I., Bengio, Y., Courville, A. (2016). Deep learning. MIT Press.

[43] Kiranyaz, S., Ince, T., Gabbouj, M. (2019). 1D Convolutional Neural Networks and Applications: A Survey. arXiv preprint arXiv:1905.03554.

[44] Krizhevsky, A., Sutskever, I., Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems, 25, 1097-1105.

[45] Lecun, Y., Bottou, L., Bengio, Y., Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

[46] Lin, M., Chen, Q., Yan, S. (2013). Network in network. arXiv preprint arXiv:1312.4400.

[47] Zhang, X., Zou, J., He, K., Sun, J. (2015). Accelerating very deep convolutional networks for classification and detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 38(1), 194-205.

[48] Hu, J., Shen, L., Sun, G. (2018). Squeeze-and-excitation networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 7132-7141).

[49] Dong, S. (2024). Astrophysical Insights Through Gravitational Wave Data Analysis: Data Preprocessing. viXra preprint viXra:2407.0026.