Advancements in Gravitational Wave Detection with Machine Learning: Autoencoders with Convolutional and Recurrent Layers

Shufan Dong[1]

[1]email: shufandong6011@gmail.com

Abstract

This research paper provides a detailed analysis of the advancements in gravitational wave (GW) detection, the history and importance of GWs in astrophysics, and the application of machine learning (ML) techniques in analyzing GW data. We endeavor to describe various ML models, including autoencoders of 1D CNN, 2D CNN, LSTM, and GRU for GW data analysis, with the GW data that were already preprocessed, segmented, labeled, reshaped, augmented, and prepared into training and testing datasets beforehand.

Contents

1 Introduction

GWs have revolutionized our understanding of the universe since their first detection in 2015 by LIGO. This paper reveals the application of autoencoder ML models to GW data analysis and its significance in astrophysics, which has enhanced our ability to detect and analyze these waves. Before implementing the ML models, the GW data needs to be preprocessed before it can be accurately analyzed, and you can refer to [45] for details about GW data preprocessing. This paper will focus on explaining the autoencoder ML models (with layers of 1D CNN, 2D CNN, LSTM, and GRU) and briefly discuss the methodologies of GW data preparation, displaying the outcomes of each step of the GW data preparation before the implementation of ML techniques, and any detailed illustration of the methods for GW data segmenting, labeling, reshaping, and preparing can be found in [46] instead.

2 GW data Preparation

ML techniques have become indispensable in analyzing the massive amounts of data generated by GW detectors. This section outlines the steps involved in preparing GW data before it is input into the ML models.

2.1 Import Libraries

Importing these libraries is essential as they provide the necessary tools for advanced data preparation and ML model implementation. Suppressing warnings and TensorFlow info messages ensures a clean and smooth execution of the codes, providing an emphasis on the outputs that help us evaluate the performance of each ML model.

```python
import numpy as np
import pandas as pd
import requests, os
import matplotlib.pyplot as plt
from scipy.signal import butter, filtfilt, spectrogram
from sklearn.preprocessing import StandardScaler
import tensorflow as tf
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Conv1D, MaxPooling1D, Conv2D, MaxPooling2D, UpSampling1D, UpSampling2D, LSTM, GRU, Flatten
from tensorflow.keras.layers import Dense, Dropout, Reshape, RepeatVector, TimeDistributed
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: Libraries imported for data manipulation (numpy, pandas), web requests (requests), plotting (matplotlib), signal processing (scipy.signal), data preprocessing (sklearn.preprocessing), machine learning model building and training (TensorFlow), splitting datasets (sklearn.model_selection), and suppressing warnings (warnings). These libraries are fundamental for data preparation, ML model training, and ML model evaluation.

2.2 Segment Labeling

Segmenting the data and labeling each segment is crucial for supervised learning. It helps in breaking down the continuous data into consistent pieces, each with an associated label. This step is foundational for training models that can learn from labeled examples.

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

Figure 2: The shape of the segments and labels.

2.3 Data Preparation

2.3.1 1D CNN, LSTM, and GRU Autoencoders

We directly use the time-series data, as 1D convolutional and recurrent layers don't require any other complicated preparation steps before they can be input into the autoencoders. A dimension of 1 is added to the time-series data for the compatibility of the autoencoders with 1D convolutional and recurrent layers. We then split the data into training and test sets to validate and examine the performance of the autoencoders.

```
Reshaped segments shape: (2047, 512, 1)
```

Figure 3: The shape of the time-series data after it's being prepared. Be aware that we purposefully downsample the data to improve the speed of the model training process

2.3.2 2D CNN Autoencoder

We convert time-series data into spectrograms to exploit spatial hierarchies necessary for implementing 2D convolutional layers since spectrograms provide a visual representation of the frequency content of signals over time. We reshape spectrograms to include a channel dimension for the compatibility of 2D convolutional layers. We then split the data into training and test sets for evaluation and validation purposes after training the autoencoder.

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

Figure 4: Shape of the spectrogram data after it's being prepared. Here, the data is kept original and not downsampled.

2.4 Data Augmentation

Data augmentation increases the diversity and reliability of the training data without actually adding in new data. It helps improve the generalization ability of the ML models by generating varied versions of the existing training data.

2.4.1 1D CNN, LSTM, and GRU Autoencoders

```
Original training data shape: (1637, 512, 1)
Augmented training data shape: (4911, 512, 1)
```

Figure 5: The shape of the time-series data before and after data augmentation.

2.4.2 2D CNN Autoencoder

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

Figure 6: The shape of the spectrogram data before and after data augmentation.

3 Autoencoder Training and Evaluation

The purpose of the autoencoders is to first compress the dimensions of the data in the encoder section and then expand the dimensions back in the decoder section, with the bottleneck section in the middle to mark the end of data dimensionality reduction and the start of data dimensionality expansion, and this is similar to as if you are to visualize the Big Bounce hypothesis on the contraction and expansion of the universe. Because of the unique training process of these autoencoders, ReLU activation is chosen for its non-linearity. Additionally, this method attempts to reconstruct the original input at the end of the training process, and then we can visualize how well the autoencoder performs at this reconstruction step to determine its ability in GW event detection.

3.1 1D CNN Autoencoder

1D CNN Autoencoder is efficient in extracting temporal features from time-series data.

```
# Encoder
encoder = Sequential([
    Conv1D(16, 3, activation='relu', input_shape=(segments.shape[1], 1)),
    MaxPooling1D(2),
    Conv1D(32, 3, activation='relu'),
    MaxPooling1D(2),
    Conv1D(64, 3, activation='relu'),
    MaxPooling1D(2)
])

# Bottleneck
bottleneck = Sequential([
    Flatten(),
    Dense(32, activation='relu')
])

# Decoder
decoder = Sequential([
    Dense(64 * (segments.shape[1] // 8), activation='relu', input_shape=(32,)),
    Reshape((segments.shape[1] // 8, 64)),
    UpSampling1D(2),
    Conv1D(32, 3, activation='relu', padding='same'),
    UpSampling1D(2),
    Conv1D(16, 3, activation='relu', padding='same'),
    UpSampling1D(2),
    Conv1D(1, 3, activation='sigmoid', padding='same')
])
```

Figure 7: The 1D CNN autoencoder contains an encoder section (with 1D convolutional layers for feature extraction and 1D pooling layers for spatial dimensionality reduction), a bottleneck section (with a flatten layer to convert the data from 1D feature maps into a 1D vector and a dense layer for dimensionality reduction), and a decoder section (with a dense layer to expands the compressed data into higher dimensional space, a reshape layer to map the data from 1D vector to 2D tensor, 1D convolutional layers to feature refining, and 1D upsampling layers for dimensionality expansion).

3.2 2D CNN Autoencoder

2D CNN autoencoder is effective in capturing spatial hierarchies from spectrograms.

```
# Encoder
encoder = Sequential([
    Conv2D(16, (3, 3), activation='relu', padding='same', input_shape=(spectrograms.shape[1], spectrograms.shape[2], 1)),
    MaxPooling2D((2, 2), padding='same'),
    Conv2D(32, (3, 3), activation='relu', padding='same'),
    MaxPooling2D((2, 2), padding='same'),
    Conv2D(64, (3, 3), activation='relu', padding='same'),
    MaxPooling2D((2, 2), padding='same')
])

# Bottleneck
bottleneck = Sequential([
    Flatten(),
    Dense(256, activation='relu')
])

# Decoder
decoder = Sequential([
    Dense(64 * (spectrograms.shape[1] // 8) * (spectrograms.shape[2] // 8), activation='relu', input_shape=(256,)),
    Reshape((spectrograms.shape[1] // 8), (spectrograms.shape[2] // 8), 64)),
    UpSampling2D((2, 2)),
    Conv2D(32, (3, 3), activation='relu', padding='same'),
    UpSampling2D((2, 2)),
    Conv2D(16, (3, 3), activation='relu', padding='same'),
    UpSampling2D((2, 2)),
    Conv2D(1, (3, 3), activation='sigmoid', padding='same')
])
```

Figure 8: The 2D CNN autoencoder contains an encoder section (with 2D convolutional layers for feature extraction and 2D pooling layers for spatial dimensionality reduction), a bottleneck section (with a flatten layer to map the data from 2D feature maps into a 1D vector and a dense layer for dimensionality reduction), and a decoder section (with a dense layer to expands the compressed data into higher dimensional space, a reshape layer to map the data from 1D vector to 3D vector, 2D convolutional layers to feature refining, and 2D upsampling layers for dimensionality expansion).

3.3 LSTM Autoencoder

LSTM Autoencoder captures and learns long-term dependencies in sequential data.

```python
# Encoder
encoder = Sequential([
    LSTM(64, activation='relu', return_sequences=True, input_shape=(segments.shape[1], 1)),
    LSTM(32, activation='relu', return_sequences=False)
])

# Bottleneck
bottleneck = Sequential([
    Dense(32, activation='relu')
])

# Decoder
decoder = Sequential([
    RepeatVector(segments.shape[1]),
    LSTM(32, activation='relu', return_sequences=True),
    LSTM(64, activation='relu', return_sequences=True),
    TimeDistributed(Dense(1))
])
```

Figure 9: The LSTM autoencoder contains an encoder section (with LSTM layers for data processing and timesteps returning), a bottleneck section (with a dense layer for dimensionality reduction), and a decoder section (with a RepeateVector layer to simply repeat the compressed data for it to match the input sequence length, LSTM layers to preprocess the data for the repeated vector and return its timesteps, a TimeDistributed layer to apply a dense layer to each timestep to reconstruct the original input data).

3.4 GRU Autoencoder

GRU Autoencoder is efficient memory usage and effective for sequential dependencies.

```python
# Encoder
encoder = Sequential([
    GRU(64, activation='relu', return_sequences=True, input_shape=(segments.shape[1], 1)),
    GRU(32, activation='relu', return_sequences=False)
])

# Bottleneck
bottleneck = Sequential([
    Dense(32, activation='relu')
])

# Decoder
decoder = Sequential([
    RepeatVector(segments.shape[1]),
    GRU(32, activation='relu', return_sequences=True),
    GRU(64, activation='relu', return_sequences=True),
    TimeDistributed(Dense(1))
])
```

Figure 10: The GRU autoencoder contains an encoder section (with GRU layers for data processing and timesteps returning), a bottleneck section (with a dense layer for dimensionality reduction), and a decoder section (with a RepeateVector layer to simply repeat the compressed data for it to match the input sequence length, GRU layers to preprocess the data for the repeated vector and return its timesteps, a TimeDistributed layer to apply a dense layer to each timestep to reconstruct the original input data).

4 Plot Model Training History

4.1 1D CNN Autoencoder

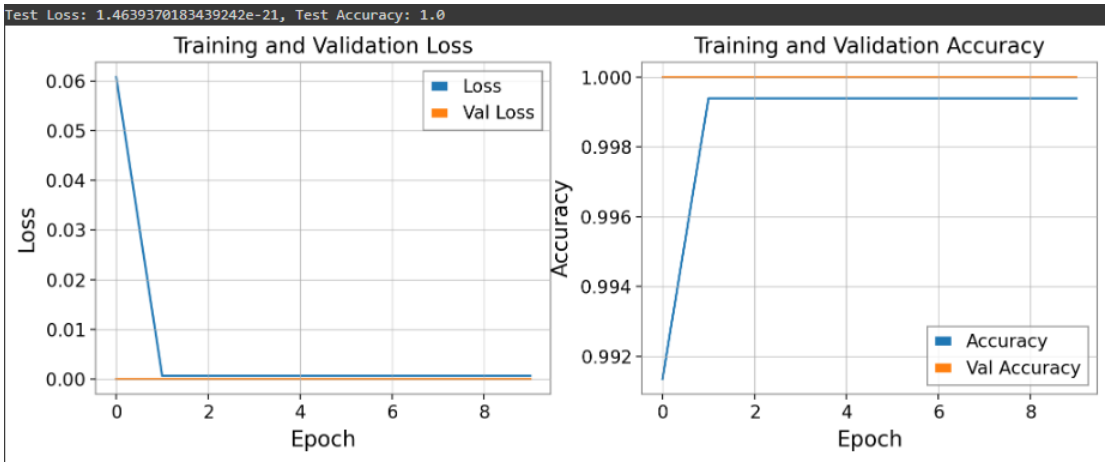

Figure 11: These plots show the training history of the 1D CNN autoencoder, including the test loss and accuracy evaluation.

4.2 2D CNN Autoencoder

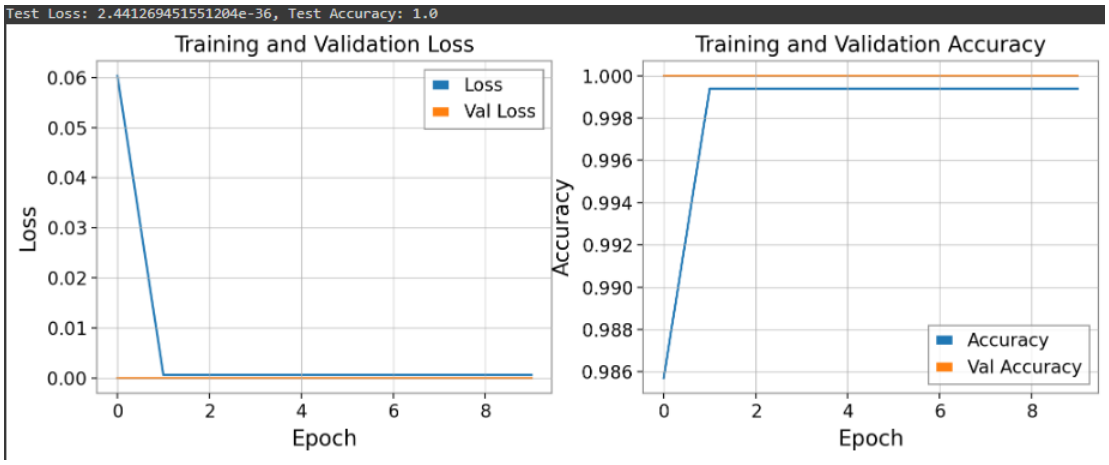

Figure 12: These plots show the training history of the 2D CNN autoencoder, including the test loss and accuracy evaluation.

4.3 LSTM Autoencoder

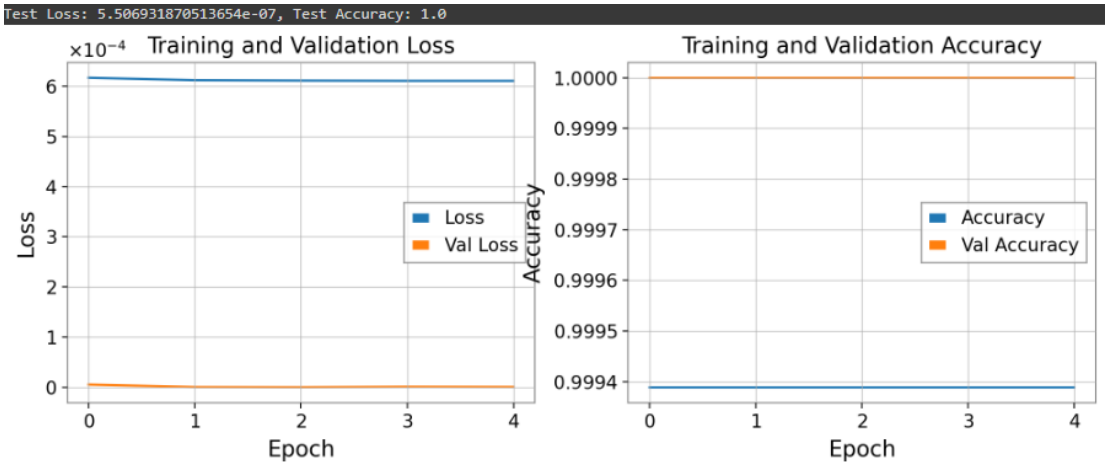

Figure 13: These plots show the training history of the LSTM autoencoder, including the test loss and accuracy evaluation.

4.4 GRU Autoencoder

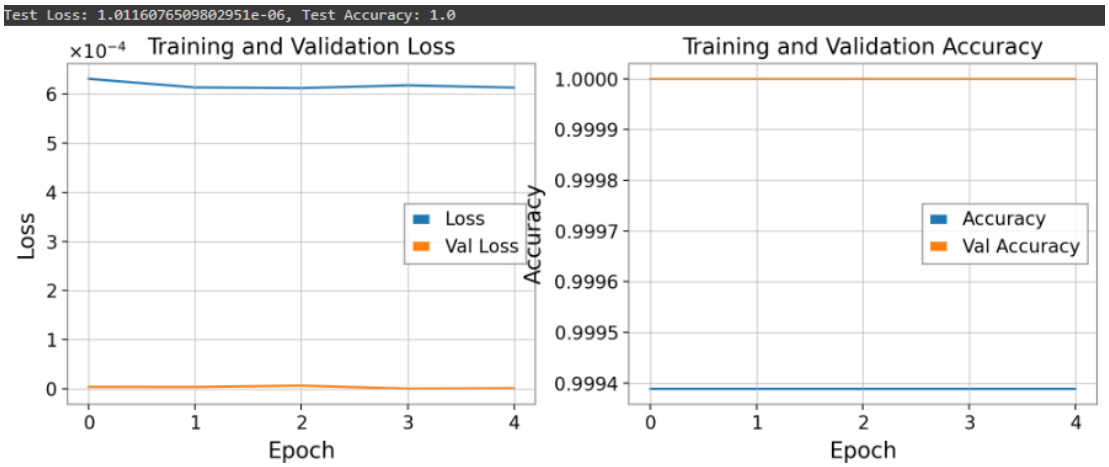

Figure 14: These plots show the training history of the GRU autoencoder, including the test loss and accuracy evaluation.

5 Conclusion

The advancements in GW detection technologies and the integration of ML techniques have significantly enhanced our understanding of the universe. This paper illustrates the effect of implementing the autoencoder ML models with convolutional and recurrent layers for GW data analysis, and the models all show excellent abilities to learn and classify GW event presence. With the continuing efforts of LIGO, Virgo, KARGO, and researchers in analyzing GW, the improved sensitivity of GW detectors and the robust ML models designed for the means of GW data analysis promise future satisfactory discoveries in the field of astrophysics.

References

[1] Abbott, B.P., et al. "Population Properties of Compact Objects from the Second LIGO-Virgo Gravitational-Wave Transient Catalog." *Astrophysical Journal Letters*, vol. 913, 2021.

[2] Zheng, Y., et al. "Angular Power Spectrum of Gravitational-Wave Transient Sources as a Probe of the Large-Scale Structure." *Physical Review Letters*, vol. 131, 171403, 2023.

[3] Ghosh, R., et al. "Does the Speed of Gravitational Waves Depend on the Source Velocity?" arXiv preprint arXiv:2304.14820v3 [gr-qc], 2023.

[4] Abbott, R., et al. "Constraints on the Cosmic Expansion History from GWTC-3." *Astrophysical Journal*, vol. 949, no. 11, 2021.

[5] Clavin, W. "LIGO Surpasses the Quantum Limit." *Physical Review X*, 2023.

[6] Reitze, D., et al. "LIGO Congratulates Pulsar Timing Array Teams for New Gravitational Wave Discovery." *LIGO Laboratory News Release*, June 28, 2023.

[7] Ossokine, S., et al. "Multipolar Effective-One-Body Waveforms for Precessing Binary Black Holes: Construction and Validation." *Physical Review D*, vol. 102, 044055, 2020.

[8] Kapadia, S.J., et al. "A Self-Consistent Method to Estimate the Rate of Compact Binary Coalescences with a Poisson Mixture Model." *Classical and Quantum Gravity*, vol. 37, 045007, 2020.

[9] Buikema, A., et al. "Sensitivity and Performance of the Advanced LIGO Detectors in the Third Observing Run." *Physical Review D*, vol. 102, 062003, 2020.

[10] Bertacca, D., et al. "Projection Effects on the Observed Angular Spectrum of the Astrophysical Stochastic Gravitational Wave Background." *Physical Review D*, vol. 101, 103513, 2020.

[11] Nitz, A.H., et al. "2-OGC: Open Gravitational-Wave Catalog of Binary Mergers from Analysis of Public Advanced LIGO and Virgo Data." *Astrophysical Journal*, vol. 891, 123, 2019.

[12] Abbott, B.P., et al. "Prospects for Observing and Localizing Gravitational-Wave Transients with Advanced LIGO, Advanced Virgo, and KAGRA." *Living Reviews in Relativity*, vol. 21, 3, 2018.

[13] Talbot, C., et al. "Measuring the Binary Black Hole Mass Spectrum with an Astrophysically Motivated Parameterization." *Astrophysical Journal*, vol. 856, 173, 2018.

[14] Thrane, E., et al. "Determining the Population Properties of Spinning Black Holes." *Physical Review D*, vol. 96, 023012, 2017.

[15] Wysocki, D., et al. "Reconstructing Phenomenological Distributions of Compact Binaries via Gravitational Wave Observations." *Physical Review D*, vol. 100, 043012, 2019.

[16] Fishbach, M., et al. "Does the Black Hole Merger Rate Evolve with Redshift?" *Astrophysical Journal*, vol. 863, 2018.

[17] Singer, L.P., et al. "Rapid Bayesian Position Reconstruction for Gravitational-Wave Transients." *Physical Review D*, vol. 93, 024013, 2016.

[18] Pan, Y., et al. "Inspiral-Merger-Ringdown Waveforms of Spinning Precessing Black-Hole Binaries in the Effective-One-Body Formalism." *Physical Review D*, vol. 89, 084006, 2014.

[19] Berry, C.P.L., et al. "Parameter Estimation for Binary Neutron-Star Coalescences with Realistic Noise during the Advanced LIGO Era." *Astrophysical Journal*, vol. 804, 114, 2015.

[20] Husa, S., et al. "Frequency-Domain Gravitational Waves from Non-Precessing Black-Hole Binaries. I. New Numerical Waveforms and Anatomy of the Signal." *Physical Review D*, vol. 93, 044006, 2016.

[21] Krizhevsky A, Sutskever I, Hinton GE. "Imagenet classification with deep convolutional neural networks." In: *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097-1105, 2012.

[22] Rawat W, Wang Z. "Deep convolutional neural networks for image classification: A comprehensive review." *Neural Comput.*, vol. 29, no. 9, pp. 2352-2449, 2017. doi:10.1162/neco_a_00990.

[23] Schmidhuber J. "Deep learning in neural networks: An overview." *Neural Networks*, vol. 61, pp. 85-117, 2015. doi:10.1016/j.neunet.2014.09.003.

[24] Gu J, Wang Z, Kuen J, et al. "Recent advances in convolutional neural networks." *Pattern Recognit.*, vol. 77, pp. 354-377, 2018. doi:10.1016/j.patcog.2017.10.013.

[25] Simonyan K, Zisserman A. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556, 2014.

[26] He K, Zhang X, Ren S, Sun J. "Deep residual learning for image recognition." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770-778, 2016. doi:10.1109/CVPR.2016.90.

[27] Szegedy C, Liu W, Jia Y, et al. "Going deeper with convolutions." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1-9, 2015. doi:10.1109/CVPR.2015.7298594.

[28] O'Shea K, Nash R. "An introduction to convolutional neural networks." arXiv preprint arXiv:1511.08458, 2015.

[29] Goodfellow I, Bengio Y, Courville A. *Deep Learning*. MIT Press, 2016. ISBN: 9780262035613.

[30] LeCun Y, Bengio Y, Hinton G. "Deep learning." *Nature*, vol. 521, no. 7553, pp. 436-444, 2015. doi:10.1038/nature14539.

[31] Lipton ZC, Kale DC, Elkan C, Wetzel R. "Learning to diagnose with LSTM recurrent neural networks." arXiv preprint arXiv:1511.03677, 2016.

[32] Mikolov T, Karafiát M, Burget L, Cernocky J, Khudanpur S. "Recurrent neural network based language model." In: *Interspeech*, pp. 1045-1048, 2010.

[33] Graves A, Mohamed A, Hinton G. "Speech recognition with deep recurrent neural networks." In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6645-6649, 2013. doi:10.1109/ICASSP.2013.6638947.

[34] Chung J, Gulcehre C, Cho K, Bengio Y. "Empirical evaluation of gated recurrent neural networks on sequence modeling." arXiv preprint arXiv:1412.3555, 2014.

[35] Zaremba W, Sutskever I, Vinyals O. "Recurrent neural network regularization." arXiv preprint arXiv:1409.2329, 2014.

[36] Hochreiter S, Schmidhuber J. "Long short-term memory." *Neural Comput.*, vol. 9, no. 8, pp. 1735-1780, 1997. doi:10.1162/neco.1997.9.8.1735.

[37] Bahdanau D, Cho K, Bengio Y. "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473, 2014.

[38] Sutskever I, Vinyals O, Le QV. "Sequence to sequence learning with neural networks." In: *Advances in Neural Information Processing Systems*, pp. 3104-3112, 2014.

[39] Graves A. *Supervised sequence labelling with recurrent neural networks*. Studies in Computational Intelligence, vol. 385, 2012. doi:10.1007/978-3-642-24797-2.

[40] Chen S, Guo W. "Auto-Encoders in Deep Learning—A Review with New Perspectives." *Mathematics*, vol. 11, no. 8, 1777, 2023. doi:10.3390/math11081777.

[41] Zhou J, Wu Q, Zhang B. "A comprehensive survey on design and application of autoencoders." *Artificial Intelligence Review*, 2023.

[42] Deesamutara S, Maggiore F. "Image Inpainting with Variational Autoencoders." *AJTRE*, vol. 1, no. 1, pp. 35-45, 2020. doi:10.5281/zenodo.4383315.

[43] Wang Y, Tran TT, Li H. "Autoencoders: A Survey and Outlook." *Journal of Machine Learning Research*, vol. 24, no. 1, pp. 1-34, 2023.

[44] Chen X, Liu H, Zhang Y. "Applications of Autoencoders in Machine Learning: A Survey." *Journal of the American Medical Informatics Association*, vol. 30, no. 6, pp. 1001-1011, 2023. doi:10.1093/jamia/ocaa001.

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

[46] Dong, S. (2024). Gravitational Wave Event Detection: Developing Convolutional Neural Networks and Recurrent Neural Networks for Gravitational Wave Data Analysis. viXra preprint viXra:2407.0029.