Analysis of supernova gravitational waves with machine learning.

Dias Suleimenov
Nazarbayev University
dias.suleimenov@nu.edu.kz

February 2, 2026

Abstract

This report evaluates a Convolutional Neural Network (CNN) designed for supernova gravitational wave classification to determine its suitability for transfer learning. The study involved replicating the baseline model and then conducting a deep analysis of its feature representations using t-SNE, benchmarking it against a simpler MLP, and testing its robustness to time-shifted data. The analysis revealed that the dataset was intrinsically simple, allowing a basic MLP to outperform the CNN, while the CNN itself learned brittle, position-dependent features and suffered from significant filter underutilization. It is concluded that the CNN fails to learn the general-purpose, robust features required for transfer learning, making it an unsustainable approach for this specific model and dataset.

1 Introduction

Objectives of project is to evaluate applicability of transfer learning to the CNN model from the article of the Abylkairov et al. [1]. This report first details the attempt to replicate the baseline model's performance. It then presents a deep analysis of the model's internal feature representations and robustness. Based on these findings, it evaluates the suitability of simpler architectures and finally draws a conclusion on the viability of transfer learning for this specific problem.

2 Baseline Model Replication and Performance

The initial objective was to establish a performance baseline by replicating the CNN architecture, shown below in Table 1, from the source article using the PyTorch framework. The experimental setup, including the early-stopping strategy, data normalization, and train/test/validation splits, was duplicated to match the original study. The model was trained using a Cross-Entropy loss function and an Adam optimizer on full GR simulations. As the source paper did not specify the hyperparameters such as learning rate, batch size, or weight decay, standard values of 1×10^{-3}, 32, and 1×10^{-5}, respectively, were used for the initial attempt.

This initial implementation achieved a mean accuracy of $96.2\% \pm 2.7\%$ over 100 random test splits (as illustrated in Figure 1). Which was slightly lower than the $97.4\% \pm 2.0\%$ reported in the original paper.

To determine if this discrepancy was caused by suboptimal hyperparameters, an automated tuning process was performed using the Optuna framework. A search of 100 trials was conducted to optimize the learning rate, batch size, and weight decay, with the objective of maximizing the mean accuracy over 10 splits. Then the best model was evaluated over 100 splits. The optimized model reached a maximum mean accuracy of $96.7\% \pm 2.5\%$.

Since extensive hyperparameter tuning did not fully close the performance gap, the remaining minor difference is likely attributable to implementation difference, such as the use of

Table 1: Architecture of the Baseline CNN Model.

Layer	**Transformation**
Conv Block 1	1D Conv (1→32, k=3) + ReLU + MaxPool(2)
Conv Block 2	1D Conv (32→64, k=3) + ReLU + MaxPool(2)
Conv Block 3	1D Conv (64→128, k=3) + ReLU + MaxPool(2)
Flatten	Flatten → 1024 features
Hidden 1	Linear Layer + ReLU (1024 → 512)
Hidden 2	Linear Layer + ReLU (512 → 256)
Output	Linear Layer (256 → 4)

PyTorch in this project versus the TensorFlow framework used in the original article. Therefore it was concluded, that the implementation is closely enough replicated the article's core results and was established as the performance baseline for all subsequent analysis.

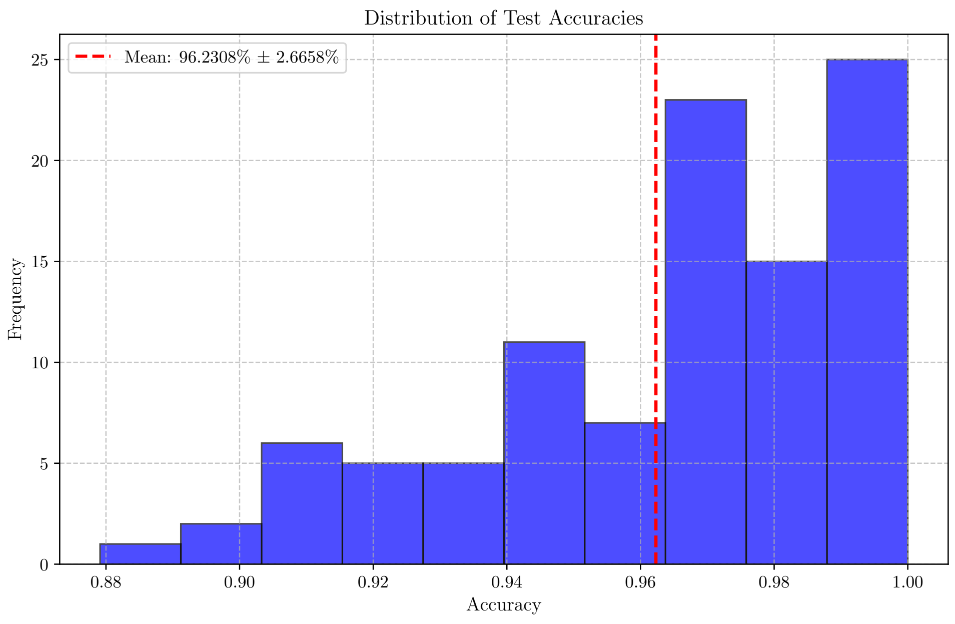


Figure 1: Histogram of CNN accuracy before optimization over 100 splits.

3 Analysis of Model and Data Characteristics

Before evaluating transfer learning, it is essential to determine if the baseline model learns meaningful, transferable features. This section investigates this question by analyzing the model's internal feature representations with t-SNE, benchmarking its performance against a simpler architecture, and testing its robustness to data perturbations.

3.1 Feature Separability Analysis with t-SNE

Given the small dataset size (452 samples), a key hypothesis was that the CNN's convolutional layers might not be learning complex, meaningful features. To test this, the internal feature representations were analyzed using t-distributed stochastic neighbor embedding (t-SNE), a method for visualizing high-dimensional data. By comparing the t-SNE plot of the raw input data to the output of the final convolutional layer (Figure 2), one can observe the degree to which the model improves class separability.

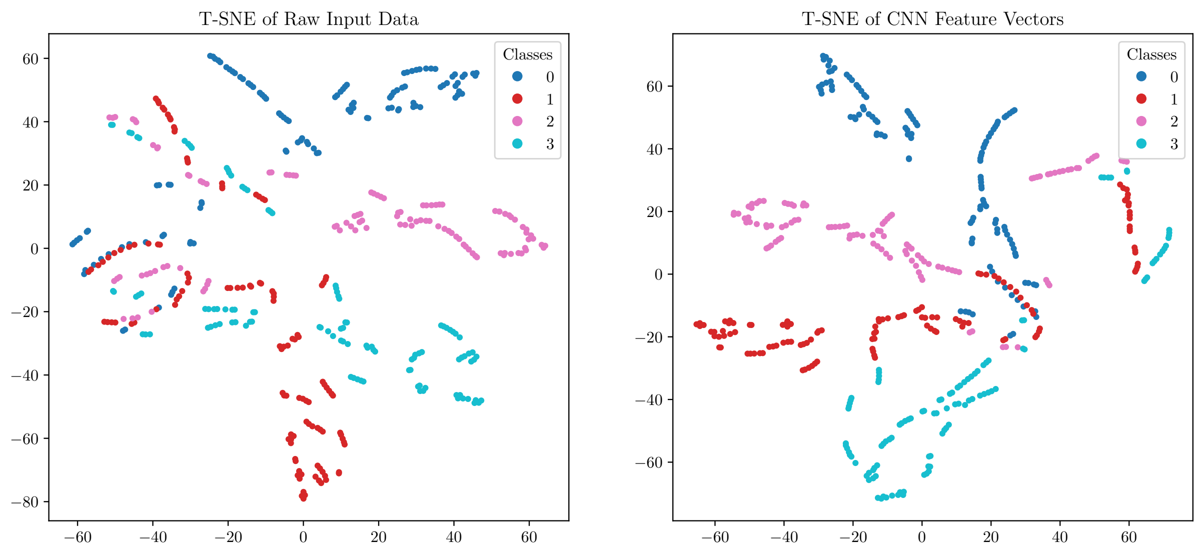


Figure 2: t-SNE visualization of the dataset's feature space. (Left) Raw input data. (Right) Output of the final convolutional layer. The figure illustrates the high degree of class separation already present in the input data, with minimal improvement from the convolutional layers.

As you can see on Figure 2 input data is already highly separated and CNN layer does not improve significantly separability. Furthermore, an analysis of filter activation revealed that a significant portion of the model's capacity was underutilized. Across 100 random data splits, an average of 72 out of 128 filters in the final convolutional layer never activated. This observation, combined with the t-SNE results, suggests that the CNN architecture is overly complex for this dataset.

3.2 Architectural Simplification: MLP as a Benchmark

Based on the evidence that the CNN was overly complex, a simpler Multi-Layer Perceptron (MLP) was implemented for perfomance comparison. The MLP's architecture, shown below in the Table 2, uses the same classifier head as the original CNN but operates directly on the flattened input data, omitting the convolutional layers entirely.

The performance of this model is higher than with baseline CNN, achieving 98.4% ± 1.5%, as shown in Figure 3. Such results further strengthen argument that CNN is overly complex for the given task.

Table 2: Architecture of the MLP Benchmark Model.

Layer	**Transformation**
Input	Flatten $\rightarrow$ 81 features
Hidden 1	Linear Layer + ReLU ($81 \rightarrow 512$)
Hidden 2	Linear Layer + ReLU ($512 \rightarrow 256$)
Output	Linear Layer ($256 \rightarrow 4$)

3.3 Evaluation of Model Robustness

A key characteristic of effective CNNs is the ability to learn shift-invariant features. Since the training data was not augmented with random time shifts, a stress test was performed to evaluate the model's robustness to such perturbations. The test involved shifting the input signals by k timesteps (where 1 timestep = 0.1 ms) and measuring the resulting model accuracy.

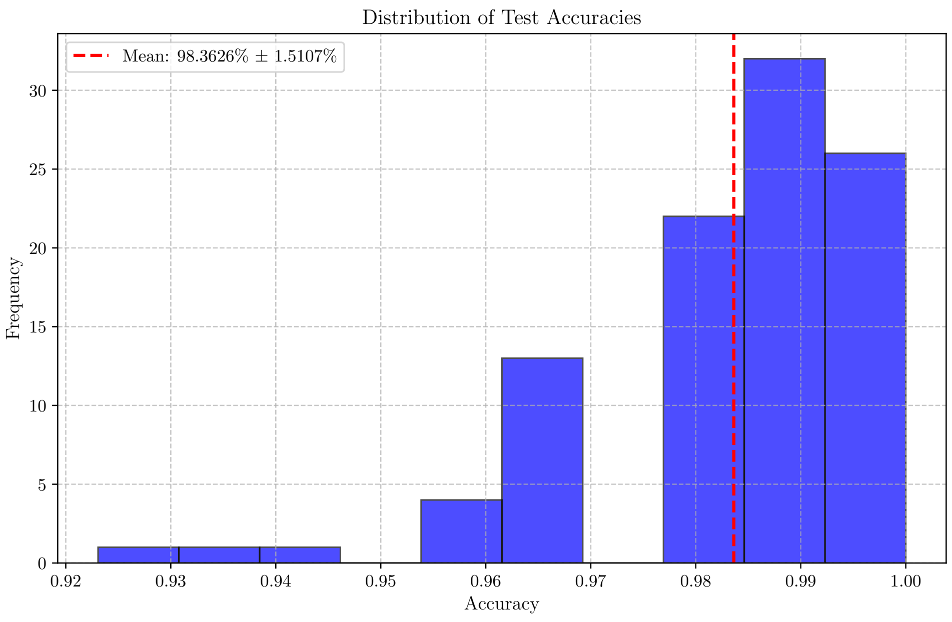
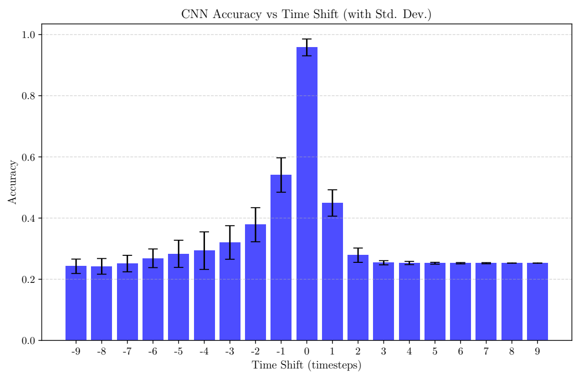
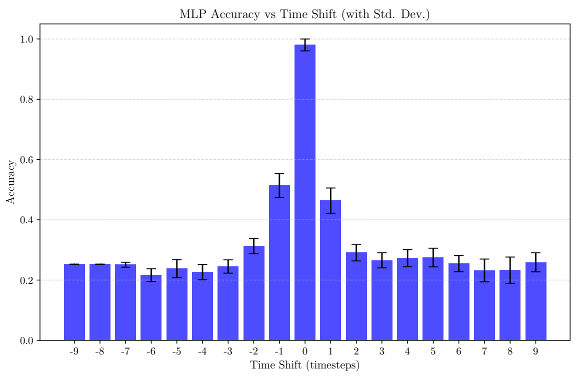


Figure 3: Histogram of MLP accuracy over 100 splits.

The results for both the CNN and MLP are shown in Figures 4a and 4b. Both models exhibit a radical drop in accuracy with even minor time shifts, indicating that their learned features are not robust and are highly dependent on the signal's absolute position. This presents a major drawback for real-world applications where data is not perfectly centered.



(a) Baseline CNN Performance



(b) MLP Performance

Figure 4: Mean accuracy (with standard deviation) of the (a) CNN and (b) MLP models evaluated on time-shifted signals over 100 splits. The x-axis represents the signal shift, k, in timesteps.

4 Discussion

The analysis made in Section 3 demonstrates that despite achieving high accuracy on the specific task, the current convolutional neural network (CNN) model is unsuitable for transfer learning. This conclusion is primarily supported by two lines of evidence: the model's excessive capacity for the given task and its learning of highly non-generalizable, dataset-specific features.

A primary argument against the model's suitability for transfer learning is the mismatch between its architectural complexity and the simplicity of the dataset. As illustrated by the t-SNE visualization in Section 3.1, the high separability of the input data suggests a redundancy in the convolutional layers. Further evidence shows, an average of 72 out of 128 filters never activated on the dataset, indicating significant model redundancy. Crucially, employing a simpler Multi-Layer Perceptron (MLP) model yielded a higher and more stable result of $98.4\% \pm 1.5\%$. This superior performance from a less complex model strongly suggests that the convolutional layers are unnecessary for this specific task.

Beyond architectural concerns, the nature of the model's learned features renders it ineffective for transfer learning. Effective transfer learning relies on the acquisition of robust, general-purpose features. However, as depicted in Figure 4, the features learned by this model exhibit high sensitivity to minor shifts in the input signal. This indicates that the model has not learned robust shift-invariant features, significantly limiting its applicability to new datasets or tasks. Instead of abstracting generalizable patterns, the model appears to have memorized simple, highly position-specific features inherent to the current dataset.

In summary, the baseline CNN model, while accurate for the current dataset, does not learn abstract, generalizable features due to the dataset's inherent simplicity. It instead focuses on memorizing specific, highly localized patterns. While this approach yields high accuracy on the current dataset, it compromises the model's utility for transfer learning to other tasks and datasets.

5 Conclusion

In conclusion, this report finds that the CNN model from the source article is not a suitable candidate for transfer learning when applied to the given dataset. This finding is supported by a combined evidence showing a fundamental mismatch between the model's complexity and the problem's simplicity. Specifically, the analysis revealed that the input data was already highly separable, a simpler MLP architecture achieved superior performance, and the CNN failed to learn robust, shift-invariant features. Future work towards creating more rigorous and generalizable CNN model should therefore focus on developing more complex datasets and implementing data augmentation techniques.

References

[1] Y. S. Abylkairov, M. C. Edwards, D. Orel, A. Mitra, B. Shukirgaliyev, and E. Abdikamalov, “Evaluating machine learning models for supernova gravitational wave signal classification,” *Machine Learning: Science and Technology*, vol. 5, no. 4, p. 045 077, 2025.