POLITECNICO
MILANO 1863

AN2DL – First Challenge Report

Backprop. Brothers

Daniele Uras, Filippo Saccomano, Jonny Arbex Marques, Gabriele Carta

Kaggle: danieleuras2, filipposaccomano, jonnyarbexmarques, gabrielecarta28

11094104, 10771424, 10764433, 11097253

November 17, 2025

1 Introduction

This report presents a complete study on a multivariate time–series classification problem based on the *Pirate Pain Dataset*. The goal is to assign each subject to one of three pain levels: **no pain**, **low pain**, or **high pain**. The available data contain heterogeneous temporal signals with different behaviours, so the task is approached as a sequence–classification problem.

The work follows a gradual and data–driven strategy. We begin with a careful exploration of the dataset, highlight critical challenges such as the strong class imbalance, and then refine both preprocessing and modelling through an iterative design process. The objective is to build a model that captures both short–term variability and long–range structure in the temporal dynamics.

2 Problem Analysis

The dataset includes 105,760 time steps from 661 subjects, each described by 38 features after removing indexing columns. These features incorporate discrete pain surveys, three categorical descriptors of body configuration, and a large set of joint–measurement signals representing movement over time.

A key difficulty is the heavy imbalance across the three classes [1], with *high pain* accounting for only about ten percent of the samples. This imbalance can distort learning if left untreated. For this reason, all features were preserved, but the modelling pipeline was built with balancing strategies, weighted losses, and a controlled windowing system aimed at stabilizing training.

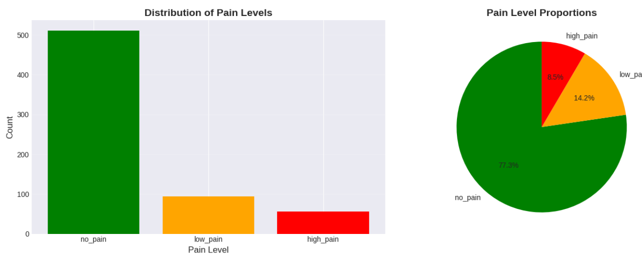

Figure 1: Plot of the Dataset distribution

The pipeline starts with an analysis of temporal dependencies. Autocorrelation curves of the joint signals were inspected to extract an effective temporal horizon. The median value of the first non–significant lag motivated the window length, with a minimum of forty samples. The stride was set to half of this value to maintain overlap without redundancy.

The categorical variables describing body configuration were converted into numerical form. Three additional time–encoding features were introduced to enrich each sequence with periodic information. Sliding windows were then applied to produce sequences of uniform length, padded as needed, and each window inherited the corresponding subject label.

A stratified 80/20 split ensured a consistent class ratio between training and validation. Given the imbalance, class weighting was integrated into the loss function. Dataloaders were finally defined with a batch size of sixteen, random shuffling, and consistent device–side optimization, forming a stable basis for the training stage.

3 Model and Training Strategy

Several neural architectures were compared under identical experimental conditions, including pure recurrent networks, convolution–only models, and hybrid architectures. The most robust and consistent results were delivered by a **Conv1D + BiLSTM** design.

Layer	**Input**	**Output**	**Params**
Conv1d (41→128)	(41,40)	(128,40)	15 872
ReLU	(128,40)	(128,40)	0
BatchNorm1d	(128,40)	(128,40)	256
MaxPool1d	(128,40)	(128,20)	0
Dropout	(128,20)	(128,20)	0
Conv1d (128→128)	(128,20)	(128,20)	49 280
ReLU	(128,20)	(128,20)	0
BatchNorm1d	(128,20)	(128,20)	256
MaxPool1d	(128,20)	(128,10)	0
Dropout	(128,10)	(128,10)	0
BiLSTM (128→128)	(10,128)	(10,256)	264 192
FC1 (256→64)	(256)	(64)	16 448
Dropout	(64)	(64)	0
FC2 (64→3)	(64)	(3)	195
Total			**346 499**

Table 1: Model architecture and number of parameters.

3.1 Conv1D + BiLSTM Architecture

The model processes each input window in two phases. The convolutional blocks extract local temporal patterns through convolution, normalization, pooling, and dropout. The refined representation is then passed to a bidirectional LSTM, which captures long–range dependencies by integrating information from both forward and backward directions.

The final hidden states are concatenated and fed to a compact dense classifier. This architecture proved particularly effective in balancing expressivity and training stability.

3.2 Loss Function

To address the class imbalance, a weighted cross–entropy loss was used. The class weights penalize errors on the minority classes more strongly, improving recall and preventing collapse into majority predictions. Label smoothing was introduced to avoid overconfidence and produced consistently smoother learning dynamics.

3.3 Training Procedure

Training adopted the AdamW optimizer with weight decay and gradient clipping to avoid instabilities in the recurrent layers. Adaptive learning rate reduction was driven by the validation F1 score through a ReduceLROnPlateau scheduler. Hyperparameters were optimized with Optuna, which effectively played the role of early stopping by selecting the configuration achieving the best macro–F1.

The optimal model included two convolutional blocks with 128 filters, a bidirectional LSTM with 128 units, a dropout of 0.11, a learning rate of 5×10^{-4}, and a weight–decay coefficient of 10^{-6}. The model exhibited smooth training curves, with the validation F1 improving steadily and no signs of overfitting [2].

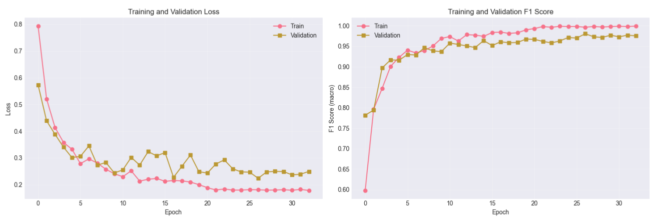

Figure 2: Those two plots represent the Loss and F1 values for training and validation through epochs.

Model	Macro-F1
Conv1D + BiLSTM	**0.9592**
ENS9 (ensemble of 9 models)	0.9544
Conv1D + BiLSTM + ATT	0.9483
Conv1D + BiGRU	0.9388

Table 2: Performance comparison of the main architectures on the validation set.

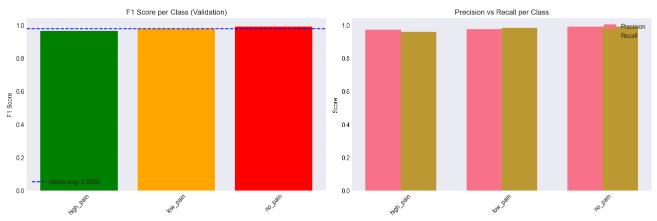

Figure 3: Those plot represent the MacroF1, Precision and Recall across the three classes.

4 Discussion

The Conv1D + BiLSTM model clearly stands out among the tested architectures. It benefits from both local filtering and long–range dependency modelling, which appear essential for the temporal structure of the joint signals.

The nine–model ensemble achieves strong performance but remains slightly below the best single model, suggesting that the chosen architecture already captures the relevant discriminative features. Attention–based variants performed stably but not better, indicating that the temporal information available in each window might already be sufficiently concentrated without the need for explicit weighting mechanisms. Replacing LSTM units with GRU cells reduced performance, reinforcing the fact that LSTMs manage the temporal variability of the dataset more effectively.

5 Conclusions

The study shows that combining convolutional layers with a bidirectional LSTM results in a reliable and efficient solution for the Pirate Pain dataset. The final model reaches a macro–F1 of 0.9551, outperforming several alternatives, including attention–based designs and multi–model ensembles. Although future work could explore advanced attention or subject–level ensembling, the current architecture already demonstrates excellent generalization and computational efficiency.

References

1. S. Kiranyaz, T. Ince, M. Gabbouj, “1D Convolutional Neural Networks and Applications: A Survey,” *Neurocomputing*, 2021.

2. T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, “Optuna: A Next-generation Hyperparameter Optimization Framework,” *KDD*, 2019.

3. L. Prechelt, “Early Stopping – But When?,” in *Neural Networks: Tricks of the Trade*, Springer, 1998.

4. M. A. Ganaie et al., “Ensemble Deep Learning: A Review,” arXiv:2104.02395, 2021.

5. C. C. Aggarwal, “Ensemble Methods,” in *Neural Networks and Deep Learning*, Springer, 2018.

6. C. N. dos Santos et al., “Attentive Pooling Networks,” arXiv:1602.03609, 2016.

7. Y. Wang et al., “Self-Attentive Pooling for Efficient Deep Learning,” arXiv:2209.07659, 2022.

8. S. Hochreiter, J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation*, 1997.

9. D. P. Kingma, J. Ba, “Adam: A Method for Stochastic Optimization,” arXiv:1412.6980, 2014.

10. I. Loshchilov, F. Hutter, “Decoupled Weight Decay Regularization,” *ICLR*, 2019.