AN2DL - First Challenge Report
The Termin-AI-tor

Giulia Di Virgilio, Georgia Manioudaki, Carlo Maria Porelli, Francesco Giuseppe Tagliabue

giuliadivii, giorgiamanioudaki, carlomariaporelli, francescotagliabue

271225, 271253, 225416, 271220

November 17, 2025

1 Introduction

This project addresses a multi-class time-series classification task using deep learning methods. The goal is to accurately classify subjects into their true pain status (no_pain, low_pain, or high_pain) categories based on temporal patterns in joint-movement data and perceived pain levels. To tackle this problem, we adopted a stepwise modeling strategy, beginning with a basic recurrent architecture and progressively increasing complexity to assess the relationship between model sophistication and performance. The **F1 score** was selected as the primary evaluation metric, as required.

2 Problem Analysis

The training dataset contains 661 subjects, each represented by a fixed-length time-series sequence from 31 motion sensors placed on various body joints. In addition to the sensor readings, four pain-survey–derived features were provided, summarizing rule-based sensor aggregations related to perceived discomfort, along with variables representing subject characteristics. Each sequence is time-stamped and associated with a label indicating the subject's true pain status. A key challenge is the presence of class imbalance: as can be seen in Figure 1, the no_pain class is substantially more represented than the low_pain and high_pain classes.

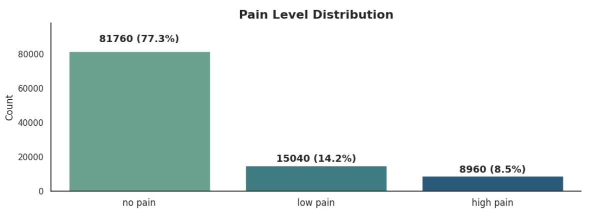

Figure 1: *Class imbalance in the training dataset.*

3 Methods

3.1 Data Preprocessing

The first step involved inspecting the dataset to understand the structure of the motion-sensor sequences and the temporal patterns relevant for pain status classification. After observing variations in joint sequence dynamics across all samples, we noticed that several of the 31 sensors displayed near-zero variance and therefore removed them to reduce the risk of overfitting. Additionally, not only did we find that the subject-characteristic features (n_[bodypart]) were 100% correlated and redundant, but we also considered them uninformative for our model. The final feature set consisted of the pain_survey variables and 16 joint sensors. After feature selection, the dataset was split into training and validation sets using an 80 : 20 ratio.

3.1.1 Normalization

To ensure numerical stability and uniform feature scaling, we applied min–max normalization, rescaling every feature to the [0,1] range. The minima and maxima were computed exclusively on data from the training set, and the resulting parameters were then applied to the validation and test sets. This type of implementation prevents data leakage and ensures a fair comparison across splits.

3.1.2 Windowing Strategy

Each motion sequence consists of 160 time steps. To capture meaningful local temporal patterns, we segmented each sequence into a series of small overlapping windows of **length 16** and **stride 4**. For window size selection, we adopted a data-driven rationale: small windows of relatively few time steps enable the model to capture localized fluctuations in the motion signals while avoiding dilution of relevant information across longer sequences.

3.2 Models

3.2.1 Baseline models: RNNs

Given the sequential nature of the time-series data, we initially developed a recurrent baseline architecture. Our first model consisted of a two-layer bidirectional **Gated Recurrent Unit (GRU)**, chosen for its ability to capture medium-range temporal dependencies while remaining computationally efficient compared to other recurrent variants[2]. As a second baseline, we implemented a two-layer **bi-Long Short-Term Memory (LSTM)** network in order to assess whether its more expressive gating mechanisms would yield an improved performance.

Table 1: *Training hyperparameters on the final model.*

Hyperparameter	Value
Learning rate	0.0001
Batch size	512
Epochs	1000
Dropout rate	0.5
Optimizer	AdamW
Weight decay	0.0001

3.2.2 Hybrid model: CNN + GRU

Building on the baseline models, we explored a hybrid architecture combining **convolutional** and **recurrent** components. A stack of 1D convolutional layers first extracts local temporal patterns from the sensor sequences. Before entering this block, each time index $t \in \{0, \dots, 159\}$ is mapped to an 8-dimensional learned embedding and concatenated with the input features to encode positional information[5]. Each convolutional block consists of a **1D convolution** (kernel size k and stride 1), ReLU, max-pooling of size p, and dropout 0.2, with channels increasing from 32 to 64. This module captures short-range motion dynamics before passing the reduced sequence to the recurrent stage[3]. The processed sequence is then fed to a **two-layer GRU** with hidden size 256 and inter-layer dropout probability of 0.5 (further information about the selected hyperparameters is listed in Table 1). Finally, a linear classification layer maps the GRU output to logits for the soft cross-entropy loss.

3.3 Loss Function

To reduce overconfidence on the dominant `no_pain` class while keeping the minority classes well defined, we employed a simple class-dependent variant of soft cross-entropy. Instead of smoothing all classes uniformly, we applied a slightly *softened* target only to the majority class. Importantly, we did not enforce the softened target to sum to one. In practice, cross-entropy can be used with non–one-hot or class-weighted targets, since its formulation only requires per-class weighting coefficients rather than a normalized probability distribution[4]. This makes it a flexible objective for incorporating class-dependent smoothing.

3.4 Inference

At test time, we adopted an inference procedure consistent with the preprocessing used during training. For each window, the model produces a logit vector $z_i \in R^C$, which is then converted into class probabilities p_i using the softmax function. All window-level probability vectors belonging to the same sample are collected and aggregated. To obtain a single prediction per sequence, we computed the **mean probability vector** across all windows associated with the same sample. The final pre-

Table 2: *Comparison of validation metrics across different seeds and model implementation choices.*

Model	Precision	Recall	F1 Score
LSTM	0.9405 ± 0.0066	0.9403 ± 0.0085	0.9395 ± 0.0077
GRU	0.9372 ± 0.0095	0.9383 ± 0.0093	0.9370 ± 0.0095
CNN+LSTM	0.9346 ± 0.0103	0.9344 ± 0.0116	0.9336 ± 0.0111
CNN+GRU	**0.9420 ± 0.0087**	**0.9424 ± 0.0091**	**0.9412 ± 0.0091**

dicted class is then obtained by selecting the class with maximum mean probability. Such probability-averaging strategy has been implemented due to the provided robust predictions, obtained by smoothing over local fluctuations that may occur in individual windows.

4 Experiments

The experimental process involved evaluating each model's behaviour over the validation dataset. We proceeded by comparing the baseline LSTM and GRU models with the hybrid convolutional ones to understand if the latter were able to achieve steadier performances. The results are shown in Table 2.

5 Results

Implementing a more elaborate model, made up of a one-dimensional CNN followed by the GRU, was able to outperform the baseline GRU and generally give more consistent results (Figure 2). Overall, the performance of GRUs and LSTMs has been found to be quite comparable with each other; however, it appears that the former behave in a more stable manner after the deployment of the convolutional layers.

6 Discussion

While all models performed similarly on the validation set, the small differences fell within normal seed variability, so validation F1 alone could not identify a clear winner. We selected the hybrid CNN+GRU model because its architecture matches the structure of the data: the convolutional layers extract short local fluctuations, while GRU captures medium to long-range temporal patterns, making it more suitable than a pure RNN[1]. Although it was not used for model selection, the hybrid model also achieved the best score on the Kaggle test set, indicating good generalization. The targeted label smoothing reduced bias toward the majority `no_pain` class, and the window-level probability averaging stabilized predictions across noisy windows. Despite these strengths, performance remains limited by the small dataset and class imbalance.

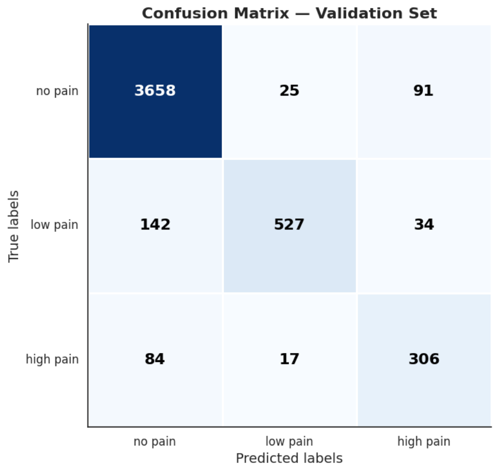

Figure 2: *Validation confusion matrix of the final model.*

7 Conclusion

This project explored several deep learning architectures for multi-class pain classification from time-series joint-movement data. After evaluating multiple recurrent and hybrid models, the CNN+GRU architecture proved to be the most suitable for capturing both local and long-range temporal patterns, and showed strong generalization in the test set. Careful preprocessing, targeted label smoothing, and probability averaging further contributed to robust performance despite class imbalance and limited data. Overall, the results demonstrate that combining convolutional layers with a GRU provides the most effective representation of sensor data, while also highlighting opportunities for improvement through better imbalance handling and model interpretability.

References

[1] H. Abdullah, N. Ali, and N. Abdullah. Evaluating the performance and behavior of cnn, lstm, and gru for classification and prediction tasks. *Iraqi Journal of Science*, 65(3):1741–1751, 2024.

[2] D. Guimarães da Silva and A. Alvarenga de Moura Meneses. Comparing long short-term memory (lstm) and bidirectional lstm deep neural networks for power consumption prediction. *Energy Reports*, 10:3315–3334, 2023.

[3] F. Karim, S. Majumdar, and H. Darabi. Insights into lstm fully convolutional networks for time series classification. *arXiv preprint arXiv:1902.10756*, 2019.

[4] G. Pereyra, G. Tucker, J. Chorowski, Ł. Kaiser, and G. Hinton. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*, 2017.

[5] H. Sigurthorsdottir, J. V. Zaen, R. Delgado-Gonzalo, and M. Lemay. Ecg classification with a convolutional recurrent neural network. *arXiv preprint*, 2020.