AN2DL - First Challenge Report
thenegatives

Edoardo Burchini, Davide Collovigh, Gabriele Corti, Nicolò Ravasio

edoardoburchinii, davidecollovigh, gabrielecorti02, nicolravasio

304662, 249818, 286479, 280092

November 17, 2025

1 Introduction

This project focuses on Time series Classification using **Neural Networks** techniques: the main goal is to assign each time series to one of the three possible classes, by capturing both temporal dependencies and relationship across multiple input classes.

2 Problem Analysis

2.1 Dataset characteristics

The data set contains **multivariate time series data**, captured from both ordinary folk and pirates over repeated observations in time. Each sample collects the temporal dynamics of the body joints and pain perception, with the goal of predicting the true state of pain.

Each record represents a time step within the recording of a subject, identified by *sample_index* and *time*. The data set includes several groups of features:

- *pain_survey_1* to *pain_survey_4*: these columns represent aggregations of sensor rules that estimate perceived pain.
- *n_legs*, *n_hands*, *n_eyes*: these columns represent the physical characteristics of the subject.
- *joint_00* to *joint_30*: are continuous measurements of body joint angles (neck, elbow, knee, etc.) over time.

Finally, the data set has a *label* column that can be one of three values:

- *no_pain*
- *low_pain*
- *high_pain*

2.2 Main challenges

We found that there is an **uneven distribution** of data among the three pain classes:

Table 1: Class distributions

no_pain	low_pain	high_pain
78%	14%	8%

2.3 Initial assumptions

We assumed that some of the features were not relevant. This was confirmed by a **Principal component analysis**.

So we excluded these features:

- *n_legs*

- *n_hands*
- *n_eyes*
- *time*

3 Method

Our approach is structured around a **progressive refinement of the classification pipeline**, starting from a simple neural model and incrementally integrating **pre-processing strategies**, **regularization techniques** and **ensemble learning**.

Before training any classifier, all numerical features were normalized using **min–max scaling** to ensure a consistent range across inputs and to stabilise gradient-based optimization. A preliminary Principal Component Analysis confirmed that *n_legs, n_hands, and n_eyes* did not meaningfully contribute to variance; therefore, these features were excluded.

As shown in *table 1* (1), the data set presents a **severe imbalance** between the three pain classes. To avoid this issue during training, we used the **Synthetic Minority Oversampling Technique (SMOTE)**, introduced in the Data Mining course, that generates synthetic positive instances to ensure a more balanced training distribution.

Given the multiclass nature of the problem, we adopted **cross-entropy** as the training loss, while optimization was performed with **AdamW**, allowing us to incorporate an explicit regularization directly into the parameter update step (see equation 1), to prevent uncontrolled weight growth without interfering with Adam's adaptive learning rates.

$$\theta_{t+1} = \theta_t - \eta \left(\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} + \lambda \theta_t \right) \tag{1}$$

Dropout further contributes to model robustness by **preventing co-adaptation** between neurons, as shown in equation 2:

$$\hat{\mathbf{w}}_j = \begin{cases} \mathbf{w}_j, & \text{with } P(c) \\ \mathbf{0}, & \text{otherwise} \end{cases} \tag{2}$$

To choose the proper model, we used **stratified k-fold cross-validation**. This protocol was applied to all neural configurations, enabling a fair comparison and guiding the selection of hyper parameters such as hidden layer size, dropout probability, number of epochs, and early stopping.

The last addition made to the final model is an **ensemble method**: specifically, eXtreme Gradient Boosting (XGBoost) was chosen. Such methods were introduced to the team in the machine learning course and in the data mining course: they are useful because they aggregate forecasts made by various predictors, thus increasing the accuracy of the final model. We decided to opt for XGBoost as it is an incredibly effective boosting technique due to the **continuous estimation of the log loss function's gradient**, as shown in equation 3:

$$\hat{y}_i = \hat{y}_i + \alpha \nabla LogLoss(y, \hat{y}) \tag{3}$$

and the use of both Lasso and Ridge regularization techniques to prevent overfitting.

4 Experiments

During our experiments, five main models were developed.

The first model (**M1**) consists of a **basic two layer neural network with static train-validation split**. This model has subsequently been used as a baseline model. The second model (**M2**) was obtained by fine-tuning M1, specifically by increasing the number of epochs and adding an **early stopping mechanism** during training.

From model three (**M3**) onward, the neural network was enriched by adopting a stratified **6-fold cross-validation approach**, as well as further fine tuning, this time regarding the network layers' size, which were increased, and by adding a **dropout parameter** to prevent overfitting. The fourth model (**M4**) is characterized by an **oversampling method** to deal with the unbalanced nature of the classes present in the data set. In particular, the SMOTE class provided by the imbalanced learning library in Python was used to deal with the uneven nature of the set.

Finally, the fith model (**M5**) saw the implementation of the **XGBoost method paired with the neural network**: the probabilities provided by the two methods are weighted and averaged to provide the final prediction of each label of the test set. Furthermore, cross validation's **folds were increased to 10** to improve the stability of the evaluation metrics.

The precision, accuracy, recall, and F1 score metrics obtained during the validation phases of the models can be visualized in *table 2* (2).

Table 2: Results obtained during validation phase. For models employing cross validation, the metrics refer to the best performing validation fold.

Model	Accuracy	Precision	Recall	F1 Score - Val	F1 Score - Test
M1	96.32	96.30	96.32	96.32	93.93
M2	98.12	98.11	98.12	98.11	96.17
M3	N/A	N/A	N/A	N/A	96.65
M4	95.46	95.46	95.46	95.46	96.79
M5	95.81	95.81	95.81	95.81	97.25

5 Results

The key takeaway from our experiments is how **overfitting affects the models**. This can be particularly seen with the validation results showcased at (2): early classifiers were prone to overfitting, since, despite performing brilliantly on the validation set, **they did not generalize as well on test data**. For this reason, we decided to enrich the model further and further.

Another observation made by the team is how some techniques that are theoretically useful to lower variance, such as batch normalization, did not yield the desired results, causing instead a worse performance on the test set; subsequently, they were pruned from our models.

6 Discussion

Overall, our model has **yielded satisfactory results**. Its main strengths are the **employment of various problem specific techniques**, such as SMOTE and XGBoosting that have proven to be very beneficial to the generalization abilities of our classifier. Another big strength of this model is its robustness with respect to overfitting, thanks to the employment of stratified cross validation, decoupled weight decay and dropout.

At the same time, this classifier presents some shortcomings. Specifically, our model **does not use Long Short-Term Memory**, which is a very common method to deal with time series classification problems. This might have hindered performance, as LSTM networks generally perform better than multi layer perceptron networks for time series classification.

7 Conclusions

In this project, we investigated the problem of multivariate time series classification for pain detection, **progressively developing a pipeline that combines data pre-processing, neural modeling, imbalance mitigation, and ensemble learning**. Starting from a simple baseline network, we systematically refined our approach by integrating stratified cross-validation, **dropout**, the **SMOTE oversampling** technique to address the strong class imbalance present in the dataset as well as XGBoost to enable stronger prediction capabilities.

Despite these promising outcomes, some limitations remain. The neural component of our classifier relies on a feed-forward architecture, which does not explicitly model the temporal dependencies of time series, potentially constraining performance compared to recurrent or attention-based models designed for sequence modelling.

Future work could explore several extensions. First, integrating **RNNs** or **LSTM** architectures may allow the classifier to **exploit sequential structure more effectively**. Second, further hyperparameter optimization of the XGBoost component, along with techniques such as learning-rate scheduling, could yield additional improvements. Finally, a **deeper analysis of feature importance**—particularly joint-angle trajectories—may guide more targeted modelling choices.

References

M. Matteucci, *From Perceptrons to Feed Forward Neural Networks*, slides from Artificial Neural Networks and Deep Learning, Politecnico di Milano, 2025.

M. Matteucci, *Neural Networks Training and*

Overfitting, slides from Artificial Neural Networks and Deep Learning, Politecnico di Milano, 2025.

Lecture 2: Overfitting and Regularisation, notebook from Artificial Neural Networks and Deep Learning, Politecnico di Milano, 2025.

Lecture 3: Cross Validation and Tuning, notebook from Artificial Neural Networks and Deep Learning, Politecnico di Milano, 2025.

P. Lanzi, D. Loiacono, *Classification*, slides from Data Mining, Politecnico di Milano, 2025.

P. Lanzi, D. Loiacono, *Ensemble Methods*, slides from Data Mining, Politecnico di Milano, 2025.

P.-N. Tan, M. Steinbach, A. Karpatne, and V. Kumar, *Introduction to Data Mining*, 2nd ed. Pearson, 2018. Ch. 3.6.2.

P.-N. Tan, M. Steinbach, A. Karpatne, and V. Kumar, *Introduction to Data Mining*, 2nd ed. Pearson, 2018. Ch. 6.10.5.

P.-N. Tan, M. Steinbach, A. Karpatne, and V. Kumar, *Introduction to Data Mining*, 2nd ed. Pearson, 2018. Ch. 6.11.

P.-N. Tan, M. Steinbach, A. Karpatne, and V. Kumar, *Introduction to Data Mining*, 2nd ed. Pearson, 2018. Ch. 6.12.

SMOTE - imbalanced-learn Documentation. Available at: `https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html`. Last access: 17 November 2025.

XGBClassifier - XGBoost Documentation. Available at: `https://xgboost.readthedocs.io/en/stable/python/python_api.html#xgboost.XGBClassifier`. Last access: 17 November 2025.

PyTorch Documentation - utils, cuda, optim. Available at: `https://pytorch.org/docs/stable/`. Last access: 17 November 2025.

StratifiedKFold - scikit-learn Documentation. Available at: `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html`. Last access: 17 November 2025.

Adam vs. AdamW: Understanding Weight Decay and Its Impact on Model Performance. Available at: URL. Last access: 17 November 2025.

Stratified K Fold Cross Validation. Available at: URL. Last access: 17 November 2025.