POLITECNICO
MILANO 1863

AN2DL - First Challenge Report
ANNtonio

Francesco Caracciolo, Antonino Ciancimino, Francesco Mazzola, Nicola Tummolo

francescocaracciolo2, antoninociancimino, framazzola14, nicolatummolo

274340, 286703, 280678, 286946

November 17, 2025

1 Introduction

We are given a dataset composed of samples of multivariate time series. Each sample is a patient and contains time series features coming from several sensors applied to the patient's body. The length of each time series is equal and fixed to 160 instants.

The features are the following:

- *pain_survey_1 to 4*: Categorical data (0, 1, 2) coming from pain-perceiving sensor.
- *n_legs, n_hands, n_eyes*: Number of legs, hands and eyes.
- *joint_00 to joint_30*: Numerical real-valued data coming from sensors measuring joint inclinations with respect to time.

The target labels are three: *no_pain*, *low_pain*, *high_pain*, representing the pain perceived by patients given the its series of joint inclinations and pain surveys.

Therefore, our task is to create a model aimed at solving a multivariate time series classification problem.

2 Problem Analysis

We list here several observations made on the dataset, given its peculiarities and quirks. More can be found in the notebook attached to this report.

- Column *joint_30* is constant, thus it carries no discriminative information, we decided to drop it.
- Columns *n_legs*, *n_hands*, *n_eyes* are constant, as expected, except for six samples having unique values for each of them (e.g. *one_leg+peg_leg*), these samples refer to pirates. These samples show idiosyncrasies we are not sure how to handle, we try detrending some columns showing a clear trend.
- **Columns *joint_13* to *joint_29* show utterly small values** (order of 10^{-5}), with exceptional peaks. This poses the problem of assessing whether the useful information is in the peaks or is comprised in the underlying small signal, or in both.
- The distribution of each feature on the whole dataset highlights that columns from *joint_00* to *joint_12* are normal-looking, with some having a heavy left tail (many slightly small values) and others are multimodal. This probably means that many of these joints were recorded while being mostly in one position (normal) or two different positions (multimodal).
- The correlation heatmaps before and after outlier removal show that a correlation between columns *joint_13* to *joint_25* could have been discovered, while the high correlation between the couple *joint_15*, *joint_16* and the correlation between the couple *joint_21*, *joint_29* could have been wrongly reduced.
- In the *AutoCorrelation Function* (ACF) plot of

each column we see that a correlation is very likely till lag 10, while it's probable till lag 25 (we think). Removing (winsorizing) the outliers slightly changes the *ACF* plots of *joint_13* to *joint_29*.

- The **training set is highly unbalanced** in term of target labels, the *no_pain* class amounts to $\sim 77\%$ of total target labels.

The main challenges we try to tackle is (1) to force our classifiers to **pay attention to underrepresented samples** in terms of target labels, i.e. those belonging to classes *low_pain* and *high_pain*, either by changing our model, changing our loss function, or oversampling minority classes, (2) augment our training data to prevent overfitting.

3 Method

The pre-processing steps we apply on the dataset are listed here briefly.

1. Removal of column *joint_30*.
2. Removal of rows having minority values in columns *n_legs*, *n_hands*, *n_eyes*, remove these three columns too. They represent 0.9% of the dataset, hence dropping them is admissible.
3. Outlier *winsorizing* using *IQR* with quantile 05-95 or 25-75.
4. Encoding of pain survey features in one-hot.
5. Integration of manually-engineered features velocity and acceleration of joint features (this was tried few times and never again, since the dataset would be too big).
6. Train-test-validation *stratified* splitting of the data, with 70-15-15 percentages, training the best models on the whole dataset.
7. *Max-Min normalization* of joint features in the training split with its maximum and minimum; normalization in validation and test splits with the training split maximum and minimum, to avoid *data leakage*.
8. ***Random OverSampling*** (Hayaty et al., 2021) (ROS) to oversample the minority target classes in the training split (to avoid data leakage) and make it balanced.
9. ***Principal Component Analysis*** (PCA) on joint features, retaining 95% of total variance (22 dimensions) on the training split, and with its mean and standard deviation done on validation and test splits too (to avoid data leakage).
10. **Data augmentation through *sliding window*** (Lashgari et al., 2020) of *ACF* lag size.

From now on these steps will be addressed numerically (e.g. pre-processing step 2 as *pp2*).

We investigate the following loss functions to deal with dataset imbalance (C is the number of target classes).

- **Weighted Cross-Entropy Loss**: *CE* loss, where class components are weighted with $\boldsymbol{w}$.

$$L_{\text{WCE}} = -\sum_{i=1}^{C} w_i \, y_i \log p_i$$

- **Focal loss** (Lin et al., 2017): *CE* loss variant penalizing less errors on well-predicted samples and more on badly-predicted ones, with $\boldsymbol{w}$ tuning the penalization and $\boldsymbol{\alpha}$ weighting target classes.

$$L_{\text{FL}} = -\sum_{i=1}^{C} \alpha_i \, y_i \, (1 - p_i)^{\gamma} \log(p_i)$$

The weights ($\boldsymbol{w}$, $\boldsymbol{\alpha}$) of these losses are chosen with the goal of penalizing errors on minority target classes.

- **Inverse of relative frequency**:

$$w_i = \frac{1}{freq(i)} \quad \forall i \in \{1, ..., C\}$$

- ***Soft* inverse of relative frequency**:

$$w_i = \frac{1}{\sqrt{freq(i)}} \quad \forall i \in \{1, ..., C\}$$

Both can be normalized to one, $w_i = \frac{w_i}{\sum_j^C w_j}$. Label smoothing (Szegedy et al., 2015) is always applied.

The neural network models we explore are these below, shown in order of trial and performance.

- Feed Forward Neural Network (FFNN): We didn't do *pp4* (to keep feature space small) and *pp7*, *pp8*, *pp9* (we hadn't explored them yet). We used FFNNs with input size equal to the series size times the number of features (1088), to feed them entire sequences. The number of hidden layers and neurons was tuned manually. The approach is naive, nonetheless we reached a decent F1 score.
- Recurrent Neural Network (RNN, LSTM, GRU): The whole pre-processing pipeline was tried, avoiding some steps as test. The better performing were always LSTMs and GRUs without *pp4*. The models seemed **able to learn**

Table 1: Best performing models

Model characteristics	Precision	Recall	F1	Public Kaggle F1
FFNN[1]	90.72 ± 0.38	91.05 ± 0.15	90.55 ± 0.22	90.97
CNN-GRU[2]	93.02 ± 0.20	93.05 ± 0.55	93.03 ± 0.29	93.75
LSTM[3]	93.44 ± 0.25	93.71 ± 0.21	93.59 ± 0.23	94.16
CNN-GRU-kEns[4]	**94.93 ± 1.08**	**95.91 ± 0.10**	**95.42 ± 0.54**	**95.37**

from pain survey features without one-hot encoding them. RNN performed worse, as expected because of vanishing gradients.

- LSTM/GRU with Embedding (LSTM-E, GRU-E): We briefly explored putting an embedding layer to LSTMs and GRUs to project pain survey categorical one-hot encoded features to a dense space. It didn't result in better performance, most probably because the *dictionary* size was too small (three unique values).
- Convolutional-RNN hybrid (CNN-RNN): Before the usual LSTMs and GRUs, we put a single or double 1D convolutional layer (Conv1d-BatchNorm1d-Conv1d-BatchNorm1d-MaxPool1d-ReLU). The intended result was to let the model extract significant local temporal pattern along each series, while the LSTMs or GRUs learn long-term sequences of patterns. The results were good.
- **CNN-RNN k-fold cross-validation ensemble** (CNN-RNN-kEns): This architecture is unusual, as it takes a CNN-RNN model (bidirectional GRU was the best performing one) and does k-fold-cross-validation on the training set (*pp6* not done). The **k model obtained are then ensembled, and predictions on unseen examples are made according to *soft majority vote*** (mean of softmax logits of each model, predicted class is argmax) on these models (Isensee et al., 2018). This is the architecture yielding better F1 performances.

4 Experiments and Results

Hand-out validation and test was performed while experimenting, cross-validation (changing the preprocessing to avoid data leakage) was carried out finally on best models. Hyperparameters tuning was mostly accomplished manually, few times *grid-search* and *Optuna* were applied. See *table 1* to check performances of best models and significant ones tried in the process of finding our best model.

We find surprising that **FFNN can perform time series classification** with decent performances. We are surprised even more knowing that k-fold cross-validation could be used to generate an ensemble model, in fact able to surpass all the other classifiers; notice that it's an approach somewhat similar to Bagging (Bootstrap Aggregating), being both able to **reduce variance without increasing bias** of the ensembled model.

5 Conclusions

After scrupulously cleaning and pre-processing data, we obtained a model showing remarkable performances in a task – we reckon – not trivial.

We propose to divide the pirate samples from the others and train a model just on them, given their quirks with respect to normal samples; then predict unseen pirate examples with it. The model should be simple enough not to overfit such a small dataset.

[01] FFNN, no sliding window augmenting, no oversampling, outlier winsorizing of *joint_13* to *joint_29* to *IQR* 25-95, no PCA, weighted CE, normal inverse frequency weights, 3 hidden layers, 512 hidden neurons per layer, dropout 0.2, batch size 512, l2-normalization $5 \cdot 10^{-3}$, learning rate 10^{-3}.

[2] CNN-GRU bidirectional with two convolutional layers, sliding window augmenting, oversampling of minority classes, outlier winsorizing of *joint_13* to *joint_29* to *IQR* 05-95, no PCA, weighted CE, soft weights, 5 hidden layers, 128 hidden neurons per layer, dropout 0.2, batch size 128, learning rate 10^{-3}.

[3] LSTM bidirectional, sliding window augmenting, oversampling of minority classes, outlier winsorizing of *joint_13* to *joint_29* to *IQR* 05-95, PCA done, weighted CE, soft weights, 5 hidden layers, 128 hidden neurons per layer, dropout 0.2, batch size 512, learning rate 10^{-3}.

[4] k-Fold cross-validation ensemble of CNN-GRU bidirectional with two convolutional layers, sliding window augmenting, oversampling of minority classes, outlier winsorizing of *joint_13* to *joint_29* to *IQR* 05-95, no PCA, Focal loss, soft weights, 2 hidden layers, 128 hidden neurons per layer, dropout 0.2, batch size 128, learning rate 10^{-3}.

References

Isensee, F., Petersen, J., Klein, A., Zimmerer, D., Jaeger, P. F., Kohl, S., Wasserthal, J., Koehler, G., Norajitra, T., Wirkert, S., & Maier-Hein, K. H. (2018, September 27). *NNU-NET: Self-adapting Framework for U-Net-Based Medical Image Segmentation.* arXiv.org. https://arxiv.org/abs/1809.10486

Hayaty, M., Muthmainah, S., & Ghufran, S. M. (2021). *Random and Synthetic Over-Sampling approach to resolve data imbalance in classification.* International Journal of Artificial Intelligence Research, 4(2), 86. https://doi.org/10.29099/ijair.v4i2.152

Lashgari, E., Liang, D., & Maoz, U. (2020). *Data augmentation for deep-learning-based electroencephalography.* Journal of Neuroscience Methods, 346, 108885. https://doi.org/10.1016/j.jneumeth.2020.108885

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2015, December 2). *Rethinking the inception architecture for computer vision.* arXiv.org. https://arxiv.org/abs/1512.00567

Lin, T., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). *Focal loss for dense object detection.* arXiv (Cornell University). https://doi.org/10.48550/arxiv.1708.02002