AN2DL - First Challenge Report
The Gradient Descenders

Lorenzo Bardelli, Lorenzo Moretti, Luca Zani

lorenzobardelli, lorenzo4moretti, zanilucapolimi

279912, 275124, 273824

November 17, 2025

1 Introduction

In this challenge, our task was to tackle a **classification** problem on a time-series dataset.

Each sample captures the temporal dynamics of the body joints and pain perception, with the goal of predicting the true pain status of the subject between `no_pain`, `low_pain`, and `high_pain`.

2 Problem Analysis

After a first examination of the dataset, we noticed a very unbalanced distribution of classes in the train set: `low_pain` and `high_pain` individuals, where no more than 10% of the entire dataset each.

Another pattern we noticed was that `n_legs`, `n_legs` and `n_eyes` were completely correlated: an individual having `one+peg_leg` as `n_legs` would also have `one+hook_hand` and `one+eye_patch` in the other 2 fields; allowing us to aggregate the information of these 3 columns in just one. Finally, we found that `joint_30` was constant throughout the entire train set, therefore, it could not provide any useful information for training and could be discarded.

This unbalance in class frequency and useless features poses as the main threat for overfitting, and needs to be addressed during all the model training procedures.

The main challenge was to create a model that could reliably predict all classes, taking into account the characteristics of this data set.

3 Method

3.1 The normie

Our initial approach consisted of an **RNN** model based on a **bidirectional GRU** using a plain **Cross-Entropy** loss function. Our training applied **data normalization** for the `joint_XX` columns and **hyper-parameter tuning** using grid search.

To select the window size, we analyzed the autocorrelation between feature samples, which showed a rather fast decay across all features in all pirates. In this way, we identified a suitable value of 40 time steps.

Furthermore, we added a verification step to ensure that the training, validation, and test splits preserved the same class distribution as the original dataset. This was particularly important due to the dataset's imbalance. Ensuring consistent proportions across the splits allowed us to obtain a more reliable performance estimate in our local environment and reduced the risk of misleading validation scores.

3.2 Differential Pain

Since we noted a lot of confusion between `high_pain` and `no_pain` predictions, we tried to investigate on the data what was causing this. We noted in particular that the values of the pain surveys could be misleading (similar distribution for both classes). Therefore, we tried to use the difference between pain surveys of every two consecutive time instances instead of the raw value for each sample, because we expected that the variations in the pain surveys would be more representative of the actual pain of the pirate.

We also tried to do something similar with the joints, training also using the approximated velocities and accelerations of the most important ones, since we thought that rapidly moving joints could be a denoting sign of pain in the patient.

3.3 The Layered Pain

A single model might not always represent the optimal solution for our task.

For this reason, we also experimented with a *two-stage screening approach*. In the first stage, a lightweight classifier determines whether the subject is experiencing pain (i.e., a binary decision: *pain* vs. *no pain*). Only if the subject is classified as being in pain, we proceed on to the second stage, where a dedicated model estimates the intensity of pain by distinguishing between `low_pain` and `high_pain`.

This strategy offers several potential advantages:

- It reduces the burden on the second-stage model, which only processes samples that are likely to contain meaningful pain-related information.
- It may lead to improved overall accuracy, since each model specializes in a simpler and more clearly defined task.
- It allows the use of different architectures or hyperparameters for the two stages, potentially increasing flexibility and robustness.

The only concern with this approach was that for training to classify between `low_pain` and `high_pain`, we would have a very small amount of data.

3.4 The 5 Folds

The small amount of data for `low_pain` and `high_pain` classes led us into considering the use of k-fold cross-validation. We split the data into 5 equal groups and trained 5 models, each on a different combination of 4 folds, validating them on the remaining one. Then we merged the predictions of the models by using majority voting, to get a final classification. in this approach we also used a **Weighted Cross-Entropy** loss fuction, based on the frequency of the pain class. We considered this approach because in this way, we could still validate the model locally (each one of the 5 models gets validated), and also use all the data for training, so we don't lose any precious value for training.

3.5 Jury of 5

The previous experiment gave us the idea of possibly a voting mechanism between more diverse models. We decided to train an ensemble of five relatively simple models with different size and parameters, using both **BiGRU** and **BiLSTM** to try and achieve heterogeneity of votes. To prevent overfitting, we introduced various mitigation techniques:

- **Data Augmentation**: Gaussian noise is added to joint angle features during training to improve model robustness and generalization.
- **Diverse Training Data**: Each model is trained on a different balanced subset of training data, using varying undersampling ratios for the majority class `no_pain`
- **Diverse Model Parameters**: varied both number of layers, layer size, loss weights, dropout and regularization parameters for each model
- **Soft Voting**: Final predictions are made by averaging probability distributions across all five models, leveraging the wisdom of all models to reduce variance and overfitting.

The ensemble consists of models with different configurations to hopefully achieve a result that is flexible enough and can generalize well on real world data.

4 Experiments

Although relatively simple, our **Normie** model initially achieved very strong performances, reaching an **F1-score of 0.981** on our local validation set. However, the score obtained on Kaggle was lower (**0.94**). Further analysis revealed that the model was **overfitting** the training and validation data, consolidating its lack of generalization. We attempted to mitigate overfitting by increasing both dropout and regularization, but these interventions did not produce significant improvements.

The **Differential Pain** approach was quickly abandoned because adding derivatives hindered our performance: generating too many columns making the model lose focus.

The **Layered Pain** approach showed promising results when classifying between `pain` and `no_pain` but ultimately showed to be **overfitting** especially in the last classification layer that discerns between `low_pain` and `high_pain`.

The **Five Fold** configuration helped us validate the model we started with, but was too resource-heavy to scale. It gave us the idea of trying to differentiate the models used for voting, allowing to use simpler diverse models for faster training and better generalization.

Experimenting with the **Jury of 5** we found that it was our most stable and general model, even when using smaller models that did not overfit. our F1 scores were always more consistent than other approaches with the Kaggle score showing promising generalization.

Table 1: individual model performance (local and public)

Model	Local F1	Kaggle Score
The Normie	0.9810	0.9403
The Layered Pain	0.9542	0.9419
Jury of 5	0.9938	**0.9474**

5 Results

The **Jury of 5** approach with smaller models always generalized well with Kaggle scores, often higher than local, but still not high enough. We ultimately settled on larger model that were slightly overfitting but together reached our best score.

6 Discussion

Despite our various attempts, we were not able to fully overcome overfitting, and the limited amount of data proved to be a significant constraint on model performance.

We couldn't generalize well enough on real world data to reach higher scores, even though some techniques like adding small gaussian noises or using an ensemble of smaller models focused on different characteristics of the provided data seemed promising.

The fact that our local validation score were so high didn't help at all in the process, not allowing us to spot signs of improvement until the Kaggle submission.

7 Conclusions

During this challenge we learned a lot about neural networks in practice, especially how tricky they can be. Every time we seemed to have a good idea, we always ended up struggling to obtain very little improvements.

The limited time of the challenge also didn't help, we couldn't experiment as much as we wanted and had to change our goals quickly to try and pursue a better result, instead of following a steady but slower path.

Some possible future improvements could definitely involve using hybrid approaches, mixing some of the ideas we described or improve some of the models we presented.

The **Layered Pain** model for example, seemed promising but needed better tuning . Maybe applying techniques like oversampling with a reasonable augmentations to the data could have helped us achieving a better performing model. Also, for the **Jury of 5** model, we would have liked to experiment more on creating more diverse models to specialize on different features of the dataset.