CS-433 : Machine Learning Project II

Riccardo Carpineto, Elias Naha, Kaan Uçar
Swiss Plasma Center, EPFL, Switzerland

***Abstract*—In this study, we address the challenge of distinguishing various types of disruptive events occurring in a tokamak during experimental operations. We analyze and process the time-series data of different features through multiple machine learning techniques. The most effective method we identified employs a dual-model strategy. This approach combines a fully-connected neural network, which is responsible for identifying the specific time windows of the events, with a Residual Neural Network tasked with assigning the appropriate labels.**

I. Introduction

EPFL's Swiss Plasma Center is one of the world leaders in fusion research. Fusion is based on the principle that powers the Sun. This fusion center especially excels in self-automatising its actions, where deep learning models are deeply rooted in their functioning.

The experiences in the Tokamak involves heating hydrogen gas to extremely high temperatures until it forms a plasma. The primary challenge in this endeavor is achieving and maintaining the necessary extreme temperatures and containment to allow fusion to occur, as the conditions required are even more extreme than those in the sun's core.

Our project 'Characterization and automatic differentiation between minor and major disruptions' focuses on studying these experiments and the behavior of plasma. For each conducted experiment, we receive a document with a time-series of the experiment and labels indicating when disruptions occur. This includes as well the physical values observed through the sensors of the Tokamak, such as electron temperature, plasma current, and internal inductance. During each of these experiments, either minor or major 'plasma events' occur, leading to the loss of plasma. Our task is to train a model to classify them.

II. Data preprocessing and Feature engineering

A. Overview

The data is composed of 137 shots, each shot is composed of windows of 20ms (1ms sampling frequency so 20 time points) right-centered on an event which is either major (2), minor(1), no event(0). The data is a time series with 8 columns, representing the physical recordings.
Time series data are not optimal for all the Machine Learning models, thus we did two different data preparations.

B. Time series preparation

1) **Cleaning**: We have noticed that many event windows overlap, resulting in several milliseconds of our experiments' time-series having different labels, corresponding to different events. Particularly, when major events occur, minor events have already started a few milliseconds earlier, creating a sort of chain reaction. These duplicates are managed as follows: the label of the higher class takes precedence. If there are duplicates between a major label (2) and a minor label (1), the minor ones are deleted, same process for minor label (1) and no-event label(0). Additionally, completely identical duplicates also appear within the classes, having the same physical values and times of occurrence, so we have also removed these. We finally shortened the window to 15ms instead of 20ms to keep relevant information.
2) **Columns added** :
- *IDistance*: The two first columns of our data frame consists of "IPLA", the current of the plasma, and "IPref", the theoretical value of the plasma's current inferred from the initial physical values of the Tokamak. The latter is set prior to the experience and serves as a reference. One of our new columns would then be the absolute distance between "IPLA" and "IPref", the further we are from the current of reference, the more likely the plasma is getting unstable.
- *Instability* : After talking about the plasma events with our mentors at the Swiss Plasma Center, we were told that after an event occurred, another one would be more likely to happen. So we added a "Instability" column that increases each time we encounter an event, this helps for classifying future events.
- *Time-Series columns* : We added various columns useful for time-series data like the fourier transforms, auto-correlations, partial-correlations and many more you can find more details in the website from V. Shkulov [1].

C. Window prepation

1) **Cleaning** : No cleaning was needed as the whole window gets a label instead of individual time points which resolves the problem of duplicates. We also shortened the window from 20ms to 15ms.
2) **Columns added** :
- *IDistance & Instability* : These two columns were also added in the window dataset.
- *Derivatives & means* : As the data is composed of windows of time series, instead of having a column for each time point and each physical measure, a much more efficient way is to compute the average or derivatives to keep the temporal dimension while reducing the columns. To this aim, we kept the mean of all the columns that were correlated to the label and that showed a significant difference between the 3 labels. Looking at the plots of each measure we realized that right before the disruption event occurred we could see different increase in these measures. To capture this slope we took the derivatives computed on the last three data-points (yields two derivatives). We finally also

kept values at certain time points which seemed highly informative (e.g. VLoop at time point 14)

D. Transformation

After feature selection we can proceed to standardize our data to get the most balanced weights. We standardize all the features using the Z-score standardization $Z = \frac{X-\mu}{\sigma}$ with μ being the mean and σ the standard deviation. This was done using the Scikit learn Standard Scaler function.

E. Separation of Dataset

The dataset was split into training and testing while keeping one shot out for validation to avoid overfitting and evaluate model performance on unseen data. For this we split the dataset into 80% Training and 20% Testing. Moreover, to ensure the model remains unbiased, we segregated the data based on individual experiments. In practice, this means that if the model is trained on data from a particular experiment, it is not subsequently tested on data from the same experiment. This division is crucial to prevent the model from using specific insights from an experiment during training to influence its predictions during testing, thereby maintaining the authenticity of its learning process.

III. Classifier models

Since the plasma is characterized by high dimensional data, we tried various models to explore different aspects of the time-series.

A. Recurrent Neural Network

Long Short-Term Memory (LSTM) layers are expertly tailored for handling time-series and sequential data. These layers are adept at discerning temporal dependencies and patterns, particularly in contexts where the sequence and timing of events are of paramount importance.

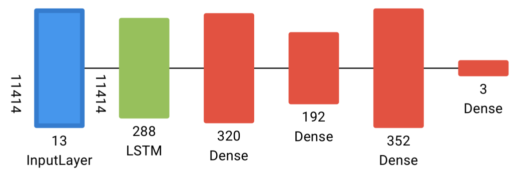

Figure 1. RNN architecture, made on Net2Vis

Our recurrent neural network (RNN) model is meticulously structured to optimize performance. It commences with an LSTM layer, which is the cornerstone of the architecture, consisting of 288 units. This layer employs a hyperbolic tangent activation function, adept at capturing the complexities of sequential data. Subsequent to the LSTM layer are three densely connected layers, each utilizing the ReLU activation function. Culminating the architecture is a dense output layer, integrated with a Softmax activation function. This final layer is crucial for class prediction, as it effectively translates the network's learned features into probabilistic outcomes.
For this model we used the categorical cross-entropy loss defined as follows : $L(y, \hat{y}) = -\sum_i y_i \log(\hat{y}_i)$

B. Neural Network

In addition to our advanced models, we developed a remarkably straightforward yet surprisingly effective neural network. Despite its simplicity, this network outperformed its more complex counterparts in terms of results. The architecture of this neural network is composed of just two layers. The first is a dense layer featuring 128 units, employing the ReLU activation function. This layer comes with L1 and L2 regularization, specifically implemented to combat the challenge of overfitting the training data. This function is renowned for its efficiency in handling non-linear data, making it an ideal choice for the foundational layer of the network. The second and final layer of the network is another dense layer, this time comprising 3 units. It utilizes a Softmax activation function, which is particularly effective for class prediction. This function converts the output into probability distributions, enabling the network to categorize inputs with a high degree of precision. The streamlined structure of this neural network demonstrates that sometimes, simpler approaches can yield surprisingly robust and accurate results. For this model, we observed a slight improvement using the Kullback-Leibler (KL) divergence loss defined as follows : $D_{\mathrm{KL}}(P \parallel Q) = \sum_i P(i) \log\left(\frac{P(i)}{Q(i)}\right)$

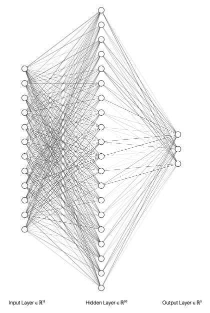

Figure 2. NN architecture, made on Net2Vis

It should be noted that this model was developed with the specific aim of serving as the initial component of our dual-model framework. Essentially, its purpose is to classify time-series data on a sample-by-sample basis. This initial classification is designed to facilitate the identification of pertinent windows of data, which will then be further analyzed by our second model in the framework.

C. Residual Neural Network

After encountering the vanishing gradient issue in our previous two neural networks, we opted for a more sophisticated model. We chose a Residual Neural Network (ResNet), known for its effectiveness in addressing this problem. ResNets incorporate "shortcuts" to facilitate gradient computations in initial layers, as detailed in the paper H. Fawaz, G. Forestier, J. Weber et al., 2019,[2]. Our network consists of three residual blocks, each with three convolutions. The output from these convolutions is added back to the input of each block and then passed on. These blocks use 64 filters each, with ReLU activation and batch normalization. The filters have lengths of 8, 5, and 3 for the first, second, and third convolutions, respectively. Following the residual blocks, the network includes a Global Average Pooling

(GAP) layer and a softmax classifier, with the number of neurons matching the dataset's class count (3 in our case).

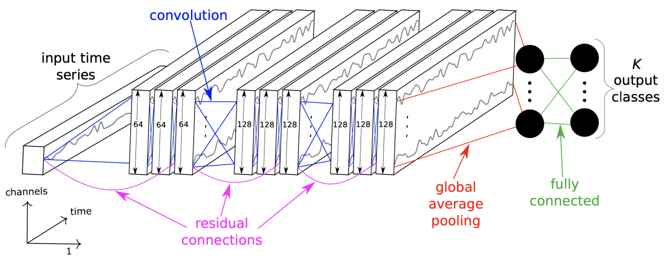

Figure 3. Residual Network architecture for time serie classification.[2]

D. Gaussian Mixture Models

In our research, we delved into the possibility that the data might possess a spatial representation amenable to separation through unsupervised learning techniques. Pursuing this line of inquiry, we implemented a Gaussian Mixture Model (GMM), a probabilistic model that assumes all the data points are generated from a mixture of several Gaussian distributions with unknown parameters.

To optimize the performance of the GMM, we experimented with various initialization methods. The choice of initialization can significantly impact the convergence and accuracy of the model, as it influences how the algorithm begins the process of iteratively refining the parameters of the Gaussian distributions. The implementation was done using the famous scikit-learn library [3].

Despite the fact that the k-means initialization method provided a reasonable approximation of the actual data, it still fell short of accurately capturing the true representation (Fig.8). This model is also a potential contender to act as the inaugural segment in our dual-model framework.

E. Support Vector Machine

Support Vector Machine (SVM) is a powerful supervised machine learning algorithm for classification. It works by finding the hyperplane that best divides a dataset into classes, with the support vectors being the data points that are closest to the hyperplane. SVM is particularly useful for classification because it is effective in high-dimensional spaces, and is versatile due to the mutliple possible kernels, making it suitable for a wide range of datasets, including those where the classes are not linearly separable.

SVM was applied on the window dataset to classify the labels based on the engineered features. We leveraged the versatility of the SVMs by varying the kernel, not knowing the data distribution. Therefore, we used the linear, polynomial, radial basis function (RBF) and sigmoid kernels. The implementation was done using the Scikit learn SVM function. A first implementation gave rise to results showing very good performances of linear and RBF.

F. Random Forest

Random Forest is a widely used classification supervised machine learning algorithm. It constructs a multitude of decision trees at training time and outputs the class that is the majority of the predicted class of the individual trees. The latter allows Random Forest to avoid overfitting while still maintaining high accuracy. The built-in process of combining features helps it perform well in complex datasets with many variables, and it's especially effective in our case where different categories overlap.

As done with SVM, Random Forest was applied to the window dataset to determine the class labels. The Random Forest algorithm was implemented thanks to the Scikit learn library as well.

G. Hyperparameter tuning

Hyperparameter tuning is an essential part of the Machine Learning pipeline, allowing us to optimize as much as possible our performance. To achieve efficient tuning, we performed a grid search, testing a variety of parameters. In this process, we utilized a 5-fold cross-validation approach, where the training dataset was divided into five subsets. In each validation cycle, four subsets were used for training and one subset for testing, this ensures that every point would be trained on. The best outputted model of the cross validation was then fitted to the initial test dataset. This method of cross-validation is crucial as it provides a more accurate measure of the model's performance on unseen data, reducing the risk of biased evaluations and helping to prevent overfitting by verifying the model's ability to generalize to new datasets.

For SVM, the parameters tuned are the regularization parameter 'C', 'gamma' value and degree (for polynomial). 'C' controls the trade-off between achieving a low error on the training data and minimizing the model complexity, while 'gamma' defines how much a single piece of training data influences the model. If 'gamma' is low, then each data point has a broad, but weak influence. If 'gamma' is high, then each data point has a strong, but narrow influence. After testing with the 3 kernels, RBF achieved highest performances. The optimal parameters for the model include `'C'` of 94, and `'gamma'` of 0.0005.

For RF, four parameters can be tuned. The number of trees, the maximum depth of the trees, the minimum samples required to split an internal node and the minimum number of samples required to be at a leaf node. The last two help controlling the size and complexity of the tree. The optimum parameters for the model include `'max_depth'` of 10, `'min_samples_leaf'` of 1, `'min_samples_split'` of 2, and `'n_estimators'` of 500.

For a neural network, it involves numerous factors, such as the number of layers, their types (e.g., Dense, LSTM, Residual), layer sizes, filter and kernel dimensions, regularization methods, learning rate, and the number of epochs. To streamline this process, we utilized Keras-Tuner as detailed in the paper . O'Malley, E. Bursztein, J. Long, F. Chollet, H. Jin, L. In-vernizzi et al., [4], a well-known Keras framework, which significantly improved our F1 scores. However, it's worth noting that our computational resources were a limiting factor, suggesting that with more powerful hardware, even better performance might be achievable.

IV. Results

A. Model Comparison

To compare our model's performance we focused on the F1 score and accuracy primarily. F1 score is particularly useful to measure performance of classification tasks where a significant class imbalance is present, such as here. F1 score was calculated using the weighted parameter of Scikit-learn package to consider class imbalance and give more importance to underrepresented classes.

Table I
F1 score from each model for each class on the test set

Classification Model	Accuracy (%)	F1 Score (%)
Residual Neural Network	96.91	96.84
Recurrent Neural Network	95.56	83.25
Dense Neural Network	99.44	99.25
Gaussian Mixture Model	67.24	74.87
Support Vector Machine	96.61	96.55
Random Forest	96.61	96.37

From the results we can deduct that a combination of our Dense and Residual neural networks would yield the best dual-model as they both are the best models for their respective tasks. Visualizing the learning curve in Fig.4 we always have a small difference never exceeding a threshold of 7% between the training and test losses, proving no overfitting of the training set. Similarly, the overall score is higher than 95% meaning that the model does not underfit.

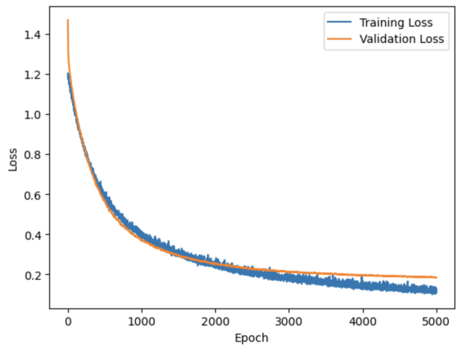

Figure 4. Learning curve

B. Ablation study

To explore feature importance, we can use permutation importance which assesses how shuffling individual feature values impacts the model's accuracy. This can be done by using the permutation function of Scikit on the RF and SVM. Both yield consisting results saying that the instability and the Vloop values are the most important features.

Parameter tuning is also essential to perform well in this classification task. Plotting our grid search (Fig.5) showcases the variation of accuracy with respect to number of estimators and max depth of each tree which are only two out of the four parameters tuned.

V. Discussion

The ResNet classifier excels in analyzing summaries from 20-frame windows, enhanced by comprehensive feature en-

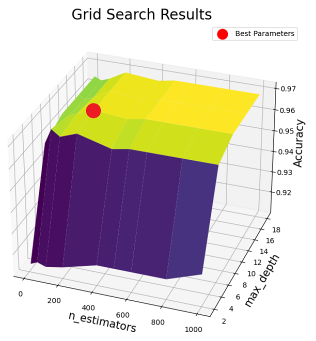

Figure 5. Grid-search over number of estimators and max depth

gineering, for assigning one of three labels. The data undergoes pre-optimization, aligning windows with events to facilitate easier extraction of characteristics. While adding new columns could be advantageous, it may not be feasible for novel, unexplored experiments. We have improved our method by incorporating a neural network into our workflow, which first identifies events before carrying out classification. This neural network examines each frame, and if most frames within a window agree on a label, that data is marked for further analysis using ResNet Architecture. Additionally, SVM and Random Forest classifiers utilize the "Instability" feature, which is beneficial to the model but challenging to compute in raw experiments. ResNet does not currently use this feature, indicating potential for enhanced efficiency in the pipeline if implemented effectively.

Upon implementing our dual-model system, which incorporates a fundamental algorithm to establish windows transferred from the initial to the subsequent network, we observed specific outcomes for a particular experiment. This experiment was not part of the training dataset but was included in the testing phase. Interestingly, the model successfully identified the designated windows during testing. However, it struggled to pinpoint the major disruption (label 2) when we manually constructed the windows. This limitation raises uncertainties about the model's overall accuracy, as our expertise does not extend to independently verifying the labels. It's plausible to suggest that the discrepancies observed, particularly the model's efficacy in the test set versus its performance with our self-created window decompositions, could be attributed to potential inadequacies in our window decomposition algorithm.

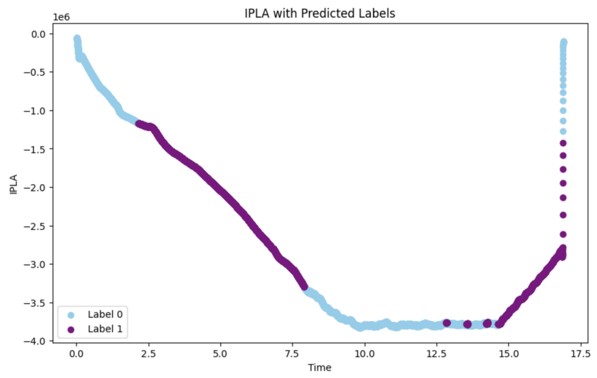

Figure 6. Dual-model performance evaluation on random experiment

VI. Ethics

The development of our machine learning algorithm, aimed at classifying disruption events in plasma during nuclear fusion, has been assessed for ethical risks, and none have been identified. This conclusion comes after a rigorous process that carefully considered both direct and indirect stakeholders, including nuclear fusion researchers, technicians, energy consumers, environmental advocates, and future generations who may benefit from the advancements in clean energy.

1) Stakeholder Consideration: In evaluating ethical risks, we considered two main categories of stakeholders. The first includes the direct users of our technology—fusion scientists and technicians whose safety and efficiency in operating nuclear fusion reactors are paramount. The second category encompasses indirect stakeholders, such as the general public and the environment, which would ultimately benefit from the clean and sustainable energy produced by nuclear fusion.

2) Risk Evaluation Process: To rule out ethical risk, we considered what harm it could do. This algorithm is meant to be used offline, thus not implicating any safety issues with plasma handling. Moreover, fusion aims to provide a universally accessible power source, designed to bridge rather than widen socioeconomic disparities, ensuring equitable benefits across diverse populations.

3) Positive Contributions: Our algorithm contributes positively to the world by improving nuclear fusion research's efficiency, thereby advancing a form of energy that promises to be revolutionary in its cleanliness and sustainability. By automating the labor-intensive process of labeling disruption events, it enhances research productivity, potentially accelerating the development of fusion energy technologies.

In conclusion, the algorithm's development has been guided by a commitment to promoting societal and environmental well-being, adhering to ethical principles, and contributing to the transformative impact of fusion energy as a clean power revolution.

References

[1] V. Shkulov, "Advanced techniques for time series data feature engineering," *HackerNoon*, 2023. [Online]. Available: https://hackernoon.com/advanced-techniques-for-time-series-data-feature-engineering

[2] I. H. Fawaz, G. Forestier, J. Weber *et al.*, "Deep learning for time series classification: a review," *Data Mining and Knowledge Discovery*, vol. 33, no. 4, pp. 917–963, 2019. [Online]. Available: https://doi.org/10.1007/s10618-019-00619-1

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[4] T. O'Malley, E. Bursztein, J. Long, F. Chollet, H. Jin, L. Invernizzi *et al.*, "Kerastuner," https://github.com/keras-team/keras-tuner, 2019.

Appendix

A. *Gaussian Mixture Model*

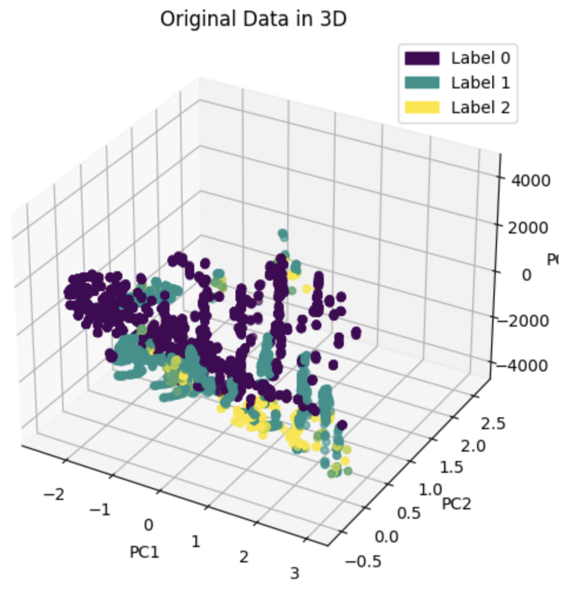

Figure 7. Original data plotted in 3D using PCA

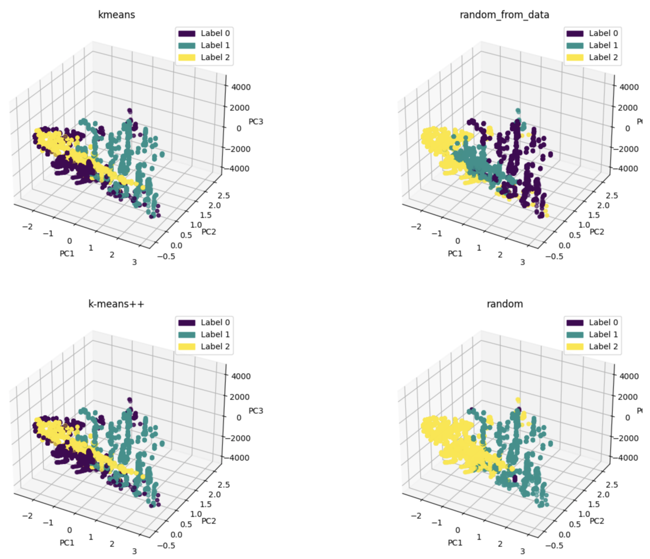

Figure 8. GMM with different initialisations

Using Principal Component Analysis (PCA), we have rendered a three-dimensional plot to observe the spatial distribution of our labels. Initially, the data appeared to lack discernible patterns, leading to skepticism about a model's ability to identify the original distribution. However, to our surprise, the Gaussian Mixture Model (GMM) with k-means initialization demonstrated a reasonably accurate understanding of the underlying label distribution. Despite this, its precision fell short of the requirements for integration into our dual-model framework.

B. *Feature engineering details*

In the initial phase of our project, our focus was directed towards examining the data to discern potential patterns that might enhance the performance of our models. See the figures presented below for illustration. We specifically selected two features for plotting, believing them to be relevant to the observed patterns.

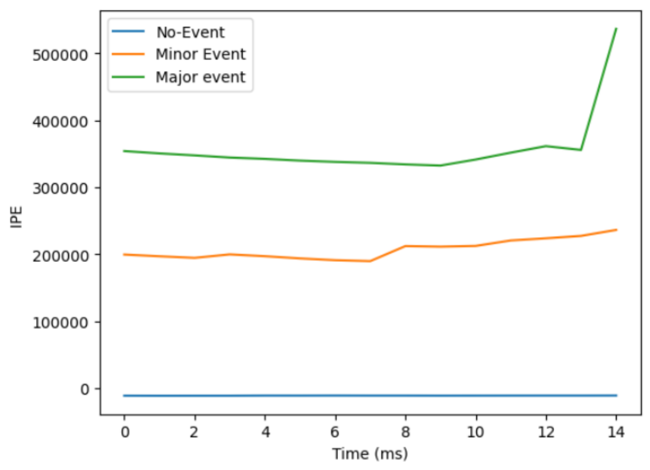

Figure 9. IPE feature average on all experiments for each label

The Internal Plasma Error (IPE) metric measures the discrepancy between the current observed in the tokamak during the experiment and the predefined reference current (IPLA - IPref). The results align remarkably with the labeled events (Fig.9). For instance, during instances with no significant events, the IPE shows minimal deviation, whereas the disparity becomes more pronounced during minor and major events.

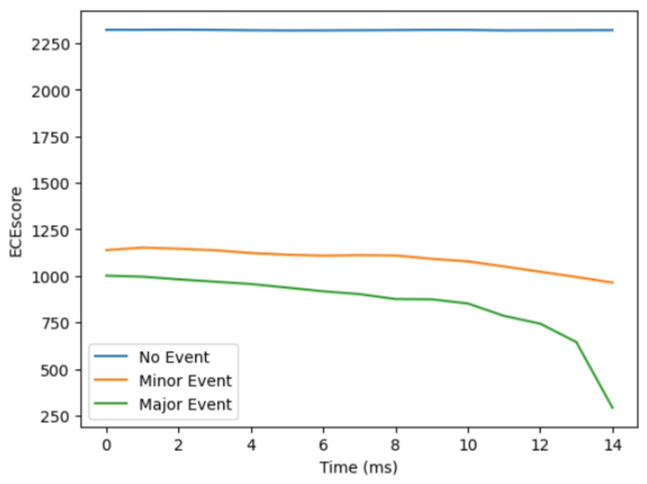

Figure 10. ECEcore feature average on all experiments for each label

The "Electron Cyclotron Emission Core" (ECEcore) feature, representing the electron temperature within the tokamak during experiments, exhibits distinct trends across different labels. For instances labeled as no-event and minor events, the ECEcore values remain relatively stable, albeit with substantial variations (differences reaching up to 1000). Conversely, in the case of major events, there is a notable decrease in the ECEcore measurements.

In our analysis, we took a broader approach to identify the significance of each feature by evaluating the correlations of each feature across all frames. This method allowed us to more accurately determine relevant sections of the data. Our findings suggest that the first five frames may not hold substantial importance, as indicated by their consistent values (Fig.11).

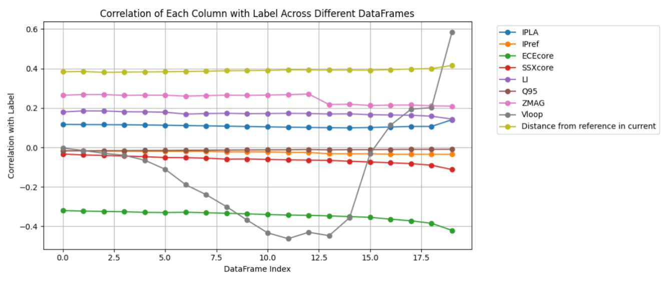

Figure 11. Feature correlations per frame across all windows with labels