# IUST at ClimateActivism 2024: Towards Optimal Stance Detection: A Systematic Study of Architectural Choices and Data Cleaning Techniques

**Ghazaleh Mahmoudi** and **Sauleh Eetemadi**
School of Computer Engineering, Iran University of Science and Technology, Iran
gh_mahmoodi@comp.iust.ac.ir, sauleh@iust.ac.ir

**Abstract**

This paper describes the IUST submission for sub-task C of the Climate Activism Shared Task at The 7th CASE workshop at EACL 2024. This work presents a systematic search of various model architecture configurations and data cleaning methods. The study evaluates the impact of data cleaning methods on the obtained results. Additionally, we demonstrate that a combination of CNN and Encoder-only models such as BERTweet outperforms FNNs. Moreover, by utilizing data augmentation, we are able to overcome the challenge of data imbalance. Our best system achieves 74.47% F1-Score on the unseen test set, outperforming the baseline by 19.97% and ranked $3^{th}$ among 19 participants.

## 1 Introduction

Climate change stands as one of the most critical challenges of our time, impacting ecosystems, economies, and communities worldwide. At the same time, understanding the public stance towards this pivotal issue is increasingly vital. Leveraging NLP techniques to gauge public stance on climate change, especially from Twitter data, provides an innovative means to comprehend diverse perspectives and sentiments in real time. To advance research in this domain, the ClimateActivism 2024 Shared Task[1] proposes three sub-tasks focused on Stance and Hate Event Detection (Thapa et al., 2024).
Sub-Task C is about Stance detection (also known as stance classification) which is a problem related to social media analysis, and natural language processing, which aims to determine the position of a person from a piece of text they produce, towards a target (a concept, idea, event, etc.) either explicitly specified in the text or implied only (Küçük and Can, 2022).

Our work focuses on exploring various model architectures and data cleaning methods to improve the performance of stance detection models on Twitter data related to climate change. We also investigate the impact of data imbalance on model performance and propose a solution using data augmentation techniques.

Our best approach utilizes a combination of Convolutional Neural Networks (CNN) and BERTweet to capture both local and global context information in the input text with Weighted Cross Entropy as loss function. Our experiments show that a combination of CNN and BERTweet outperforms Feedforward Neural Networks (FNNs) in stance detection on climate change related tweets. We also demonstrate that data augmentation can address the challenge of data imbalance, resulting improvements in model performance. We also experiment with different data cleaning methods. Moreover, the best results in the data cleaning type are achieved by removing URLs and usernames, and all experiments of this method have yielded better results compared to other data cleaning methods. Code and results are publicly available on *https://github.com/ghazaleh-mahmoodi/Climate_Activism_Stance_Detection*.

## 2 Data

The Sub-Task C (Stance Detection) dataset is part of the Multi-Aspect Twitter Dataset (Shiwakoti et al., 2024). The data was collected from tweets posted between January 1, 2022, and December 30, 2022. The selection criteria involved hashtags such as #climatecrisis, #climatechange, #ClimateEmergency, #ClimateTalk, #globalwarming, as well as activist-oriented hashtags like #FridaysForFuture, #climatestrike, etc. The dataset distribution is illustrated in Table 1.

[1] https://emw.ku.edu.tr/case-2024/

| Split | % | Support | Neutral | Against |
|---|---|---|---|---|
| Train | 70% | 4328 | 2256 | 700 |
| Dev | 15% | 897 | 511 | 153 |
| Test | 15% | 921 | 500 | 141 |
| All | 100% | 6146 | 3276 | 994 |

Table 1: Class distribution of stance detection dataset

### 2.1 Data Pre-processing

As the text data is sourced from Twitter, it is necessary to carry out pre-processing to enhance the extractable features and ensure the cleanliness of the text. When it comes to cleaning data, the principle is to not throw away any data. However, given that our data is limited and if we don't remove some noise, the model may be inaccurate (in limited data), so we need to perform a certain level of data cleaning. However, based on the assumptions we will explain below, we have examine a limited number of data cleaning methods. The defined methods will involve increasing levels of text input cleaning, from the least to the most aggressive.

I. **Original Tweet Text**: Without any changes in the text.
II. **Removing URL**: Considering that URLs are modified (e.g., ://t.co/rs1vhBp2ax), we assumed their presence in the data could cause errors.
III. **Removing username**: The existence of usernames without information about the person may create ambiguity.
IV. **Removing URL and username**: To determine the effect of removing the URL and username together.
V. **Removing URL and username and split hashtag**: For example `#FridaysForFuture` becomes `Fridays For Future`.
VI. **Removing URL and username, split hashtag, and convert all letters to lowercase**: Sometimes writing letters in capital form has a special meaning, which we want to observe its impact.
VII. **Complete cleaning**: Contains removing URL, username, stop words, punctuation, converting all letters to lowercase, and split hashtag.

### 2.2 Data Augmentation

One of the existing challenges is the imbalance of the dataset. In such conditions, the trained model tends to lean towards the class with more data. To address this issue, we generate additional data for minority class data. We use two different methods to generate data.

1. **Substitution**: We use synonym substitution as an augmentation method. We employ the method provided by python nlpaug library (Ma, 2019) based on RoBERTa (Liu et al., 2019a).

2. **Round-trip translation**: We translate the English texts to German and then back to generate extra data using python nlpaug library.

We generated 950 data points for the "oppose" class using the introduced data augmentation methods and added them to the training data . The class distribution of data before and after data augmentation can be observed in Figure 1.

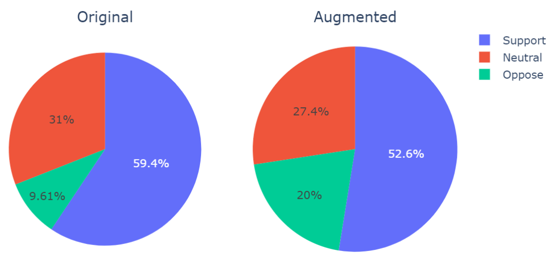

Figure 1: Train Set Class distribution

## 3 Methodology

We proposed a model comprising four modules, and to determine the most suitable parameters for each module, we conducted numerous experiments with various configurations, seeking the optimal values within the defined search space (Table 2). Using the Optuna library (Akiba et al., 2019), which employs a sampler using the TPE (Tree-structured Parzen Estimator) algorithm, we selected the optimal model configuration based on the Macro F1-score on the development set. In the following, we provide a brief explanation of the search space defined for each module.

1. **Embedding**: We are searching among several Encoder-only Language Models to determine which one to choose for extracting features from text. We chose Encoder-only models because they are more popular and efficient for text classification. The search space includes:
   - **BERT** (Devlin et al., 2019)
   - **RoBERTa**: Builds on BERT and modifies key hyperparameters, removing the

next-sentence pretraining objective and training with much larger mini-batches and learning rates.(Liu et al., 2019b).

- **BERTweet**: Trained based on the RoBERTa for English Tweets (Nguyen et al., 2020).
- **XLM-RoBERTa**: A multilingual pre-trained language model, trained on 2.5TB of filtered CommonCrawl data. (Ruder et al., 2019).
- **DEBERTA**: Improves the BERT and RoBERTa models using disentangled attention and enhanced mask decoder (He et al., 2021).

2. **Classifier**: There are two options.

   - **Fully Connected Neural Networks**. We use a three-layer network architecture with a linear layer, a ReLU activation function, and a dropout. Finally, we apply a softmax function to the output.
   - **Convolutional Neural Networks**. The architecture used is the same as the one introduced by Safaya et al. (2020), with the difference that instead of 4 last layers, we defined the search space and examined. In this architecture the embeddings are fed into parallel convolutional filters of five different sizes (768x1, 768x2, 768x3, 768x4, 768x5), with 32 filters for each size. Each Kernel utilizes the outputs from the preceding N last hidden layers[2] of Encoder-only (e.g., BERT) as separate channels and conducts a convolution operation. Following this, the resulting outputs undergo ReLU Activation and Global Max-Pooling processes. The pooled outputs are then concatenated, flattened, and fed through a dense layer and softmax function to obtain the final class.

3. **Optimizer**: The search space includes four well-known optimizers (Table 2) that have shown good performance.
4. **Loss Function**: Since we are dealing with the classification task and imbalanced data, we have chosen two loss functions that are suitable for our experiments.

   - **Focal Loss**: This loss addresses class imbalance by down-weighting easy well-classified examples during the training stage. It puts more emphasis on hard examples o improve overall performance (Lin et al., 2017).
   - **Weighted Cross Entropy**: This loss is a variant of the standard Cross-Entropy loss function that assigns different weights to individual class predictions. Class weight can be calculated for each class as the inverse of its proportion in the training data. This is commonly achieved by dividing the total number of samples by the number of samples in each class, thereby obtaining the weight to be assigned to that particular class.

| Parameter | Search Space |
|---|---|
| Classifier | [FNN, CNN] |
| N_last_layer | [1, 2, 3, 4, 5] |
| Optimizer | [Adam, AdamW, RMSprop, SGD] |
| Loss | [Cross Entropy, Focal] |

Table 2: Architecture search space

### 3.1 Hyperparameter Tuning

Hyper-parameters used in training stages are selected via tuning using the Optuna library. We choose the optimal hyperparameters by the Macro F1-score on the development set. The search space defined for hyper-parameters is present in the Table 4.

## 4 Experiments and Results

To evaluate the results, we used the Marco F1-score as the main metric and also reported Precision, Recall, and Accuracy. The hardware used in experiments is a GPU.1080Ti.xlarge with 31.3GB RAM. Each training epoch lasts 2–5 minutes on average.

In section 2.1, we introduced seven modes for data cleaning. Experiments are repeated for the mode without or with data augmentation. Therefore, we tested 14 configurations in total, including 7 modes for data cleaning and 2 modes for input data. For each configuration, we selected model

[2]This variable chooses in search space.

[3]CNN with last 5 layers of BERTTweet, Data Augmentation, Weight Cross Entropy as loss function and SGD as optimizer. Removing URL and username as data cleaning approach.

[4]CNN with last 3 layers of XLM-RoBERTa,Focal as loss function and SGD as optimizer. Removing URL and username as data cleaning approach.

| Cleaning | Aug | Embedding | Classifier | Loss | Optimizer | F1-Score | Recall | Precision | Accuracy |
|---|---|---|---|---|---|---|---|---|---|
| C1 | - | RoBERTa | CNN(N=1) | WCE | SGD | 71.74 | 69.94 | 74.83 | 68.82 |
| C2 | - | XLM-RoBERTa | CNN(N=3) | WCE | AdamW | 69.80 | 69.38 | 74.16 | 64.91 |
| C2 | - | BERT | CNN(N=2) | WCE | SGD | 70.28 | 68.64 | 73.82 | 66.00 |
| C2 | ✓ | BERT | CNN(N=3) | WCE | SGD | 68.75 | 66.27 | 75.26 | 70.16 |
| C3 | - | RoBERTa | FNN | WCE | SGD | 71.89 | 68.89 | 79.55 | 73.81 |
| C3 | ✓ | XLM-RoBERTa | CNN(N=3) | F(g=4) | SGD | 68.59 | 68.51 | 75.93 | 69.84 |
| C4 | - | XLM-RoBERTa | CNN (N=4) | WCE | RMSprop | 71.82 | 69.56 | 74.80 | 69.52 |
| C4 | - | BERT | CNN(N=5) | WCE | SGD | 72.82 | 69.56 | 74.80 | 72.85 |
| C4 | - | XLM-RoBERTa | CNN(N=3) | F(g=1) | SGD | **73.97** | **70.91** | **78.17** | **72.59** |
| C4 | - | RoBERTa | FNN | WCE | RMSprop | 71.52 | 68.31 | 78.17 | 70.06 |
| C4 | - | XLM-RoBERTa | CNN(N=4) | WCE | SGD | 72.72 | 69.38 | 78.85 | 73.17 |
| C4 | ✓ | **BERTweet** | **CNN(N=5)** | **WCE** | **SGD** | **74.47** | **70.31** | **79.31** | **73.11** |
| C4 | ✓ | BERT | CNN(N=4) | WCE | SGD | 70.64 | 67.75 | 75.63 | 70.01 |
| C5 | - | DEBERTA | FNN | WCE | Adam | 71.33 | 68.78 | 75.43 | 67.73 |
| C5 | - | XLM-RoBERTa | CNN(N=2) | WCE | SGD | 72.70 | 69.63 | 77.18 | 72.15 |
| C5 | - | BERT | FNN | F(g=1) | SGD | 72.01 | 68.62 | 77.44 | 71.75 |
| C5 | ✓ | DEBERTA | FNN | WCE | AdamW | 71.20 | 68.51 | 77.68 | 72.72 |
| C5 | ✓ | BERT | CNN(N=3) | WCE | SGD | 70.85 | 70.01 | 73.41 | 67.22 |
| C6 | - | BERT | FNN | F(g=2) | SGD | 71.83 | 68.38 | 74.48 | 72.21 |
| C6 | - | BERT | FNN | WCE | RMSProp | 70.13 | 67.18 | 74.85 | 69.65 |
| C6 | ✓ | XLM-RoBERTa | CNN(N=4) | WCE | SGD | 72.70 | 69.43 | 78.56 | 72.85 |
| C7 | - | BERT | CNN(N=5) | F(g=1) | SGD | 71.68 | 68.77 | 76.53 | 71.76 |
| C7 | ✓ | BERTweet | CNN(N=2) | F(g=4) | AdamW | 69.36 | 66.56 | 74.09 | 69.06 |

Table 3: Experiment configuration and result on climate stance detection test data.
**Data Cleaning Approuch**(C1:Original Tweet Text, C2:Removing URL, C3:Removing username, C4:Removing URL and username, C5:Removing URL and username and split hashtag, C6:Removing URL and username, split hashtag, and convert all letters to lowercase, C7:Complete cleaning). **Classifier**(CNN: Convolutional Neural Networks, FNN: Fully Connected Neural Networks). **Loss Function**(WCE:Weighted Cross Entropy Loss, F:Focal Loss, g:Gamma parameter in focal loss).

| Parameter | Search Space |
|---|---|
| Dropout | $[0.1 : 0.5]$ |
| Learning Rate | $[1e^{-5} : 1e^{-2}]$ |
| Batch Size | [4, 8] |
| Focal_gamma | [1, 2, 3, 4, 5] |

Table 4: Hyperparameters search space

| Model | ACC | F1 |
|---|---|---|
| BERTTweet[3] | **73.11** | **74.47** |
| XLM-RoBERTa[4] | 72.59 | 73.97 |
| ClimateBERT (Baseline)* | 65.1 | 54.5 |

Table 5: climate stance detection Accuracy and macro F1-Score result.* from Shiwakoti et al. (2024) report.

parameters and hyperparameters using Optuna and performed fine-tuning for 20 trials. In each trial, the parameters are selected using the sampling method TPE (Tree-structured Parzen Estimator), based on the defined search space. Additionally, a mechanism for pruning unsuccessful trials is also included by default in Optuna. Finally, the results with F1-Macro greater than 0.68 on the development set are present in Table 3 (Since we only included results F1 scores greater than 0.68, it is possible that the results for some cleaning methods may not be available for a specific classifier, such as FNN).

The experimental results indicate that the cleaning method, which removes URLs and usernames (C4), performs better compared to other methods. The complete cleaning and original text methods, on the other hand, yielded weaker results than other approaches. Additionally, it can be said that maintaining hashtags and not converting to lowercase is a better cleaning approach because sometimes writing all letters in capital letters indicates intensity of anger or opposition.

Furthermore, in general, BERT embeddings perform better in complete cleaning, while RoBERTa and XLM-RoBERTa models are more commonly used in other cleaning methods and yield better results and the best result is obtained with BERTweet.

Regarding the classifier type, usually a CNN with 4-5 last layers achieves better results. Evidence suggests that the defined CCN architecture, due to its use of different filters sizes and consideration of neighborhoods, has been able to achieve better results compared to FNN. Additionally, typically, RoBERTa and XLM-RoBERTa embeddings are used with CNN, while BERT is paired with FNN for better performance. Analyzing the experiments as a whole, it can be concluded that the best results were obtained by optimizing SGD and using Weighted Cross Entropy as loss function. Comparison of our results and the baseline illustrate in Table 5.

| Parameter | Value |
|---|---|
| Epoch | 8 |
| Batch Size | 4 |
| Dropout | 0.5 |
| Learning Rate | 0.007903 |
| Learning schedule | Linear Schedule With Warmup |
| Embedding | BERTweet |
| Classifier | CNN |
| N_last_layer | 5 |
| Optimizer | SGD |
| Loss Function | Weighted Cross Entropy |

Table 6: Best model configuration and hyperparameters.

**To determine the impact of data cleaning** on the results obtained, we repeated experiments with the best configuration (as shown in Table 6). In these experiments, hyperparameters and model architecture were kept identical, with the only variation being the method of cleaning data. For each cleaning technique, we repeated the experiments 10 times for 8 epochs. The results obtained are illustrated in the Table 7. The results indicate that C3(Removing username) and C4 (Removing URL and username) are significantly better than C1(Original Tweet Text) and C7(Complete cleaning). Thus, the influence of data cleaning methods on the final results is clearly evident.

| Cleaning | F1-Score |
|---|---|
| C1 | $73.98 \pm 0.0012$* |
| C2 | $73.92 \pm 0.0017$* |
| C3 | $74.35 \pm 0.0015$*† |
| C4 | $74.11 \pm 0.0029$*† |
| C5 | $73.76 \pm 0.0014$* |
| C6 | $73.72 \pm 0.0009$* |
| C7 | $72.42 \pm 0.0020$ |

Table 7: Experiment with Best Model Configuration and hyperparameter. † indicates significance ($p < 0.005$) comparing to C1. $*$ indicates significance ($p < 0.005$) comparing to C7.

By repeating the experiment with the best configuration (Table 6), and only changed the classifier, it demonstrated the superiority of CNN over FNN. the results of which are illustrated in Table 8.

| Classifier | Cleaning | F1-Score |
|---|---|---|
| CNN | C3 | $74.35 \pm 0.0015$ |
| | C4 | $74.11 \pm 0.0029$ |
| FNN | C3 | $73.73 \pm 0.0058$ |
| | C4 | $73.91 \pm 0.0045$ |

Table 8: Classifier impact

## 5 Error Analysis

By analyzing the model errors, it can be concluded that as expected, the model struggles with detecting the oppose class. In addition to the low number of data points in this class, the presence of sarcasm and irony in the data makes it harder for the model to fully comprehend the situation and make accurate predictions. It is evident that in parts of the text where there is sarcasm, the probability of model error significantly increases. Consider the tweet `#FridaysForFuture #ClimateChange #ExtinctionRebellion #GlobalWarming What are we saving?`. Since some of the hashtags are used to collect data, they are present in all three classes.

## 6 Conclusion

This work involved a systematic exploration of model architecture and data cleaning methods. We find that the optimal configuration combining BERTweet and CNN with Weighted Cross Entropy and SGD, along with data augmentation, led to achieving an impressive Macro F1-Score of 0.7447.

## 7 Limitation

In our research, we encountered GPU limitations, which affected the scale and speed of our model training and experimentation. Despite our efforts to optimize code efficiency and parallel processing, these limitations restricted the size of our model architectures and the volume of data we could effectively process within a reasonable timeframe. Also, we confronted limitations stemming from insufficient labeled data and imbalanced class distributions. Despite employing data augmentation techniques to mitigate the imbalance, the inadequacy of labeled data impeded the depth and robustness of our model's learning, affecting its overall performance and generalization capabilities.

## 8 Acknowledgements

We'd like to thank the organizers for introducing this task. We are glad that we had the opportunity to engage more in the challenges.

## References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 2623–2631, New York, NY, USA. Association for Computing Machinery.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*.

Dilek Küçük and Fazli Can. 2022. A tutorial on stance detection. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, WSDM '22, page 1626–1628, New York, NY, USA. Association for Computing Machinery.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019a. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Edward Ma. 2019. Nlp augmentation. https://github.com/makcedward/nlpaug.

Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. 2020. BERTweet: A pre-trained language model for English Tweets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 9–14.

Sebastian Ruder, Anders Søgaard, and Ivan Vulić. 2019. Unsupervised cross-lingual representation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, pages 31–38, Florence, Italy. Association for Computational Linguistics.

Ali Safaya, Moutasem Abdullatif, and Deniz Yuret. 2020. KUISAIL at SemEval-2020 task 12: BERT-CNN for offensive speech identification in social media. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 2054–2059, Barcelona (online). International Committee for Computational Linguistics.

Shuvam Shiwakoti, Surendrabikram Thapa, Kritesh Rauniyar, Akshyat Shah, Aashish Bhandari, and Usman Naseem. 2024. Analyzing the dynamics of climate change discourse on twitter: A new annotated corpus and multi-aspect classification. *Preprint*.

Surendrabikram Thapa, Kritesh Rauniyar, Farhan Ahmad Jafri, Shuvam Shiwakoti, Hariram Veeramani, Raghav Jain, Guneet Singh Kohli, Ali Hürriyetoğlu, and Usman Naseem. 2024. Stance and hate event detection in tweets related to climate activism - shared task at case 2024. In *Proceedings of the 7th Workshop on Challenges and Applications of Automated Extraction of Socio-political Events from Text (CASE)*.

## A Appendix

We explore the visualization of Parallel Coordinate (Figure 3) and FS-Importance (Figure 2) of our search space by functionalities offered by the Optuna package, providing a comprehensive understanding of the hyperparameter optimization process in our FNN model.

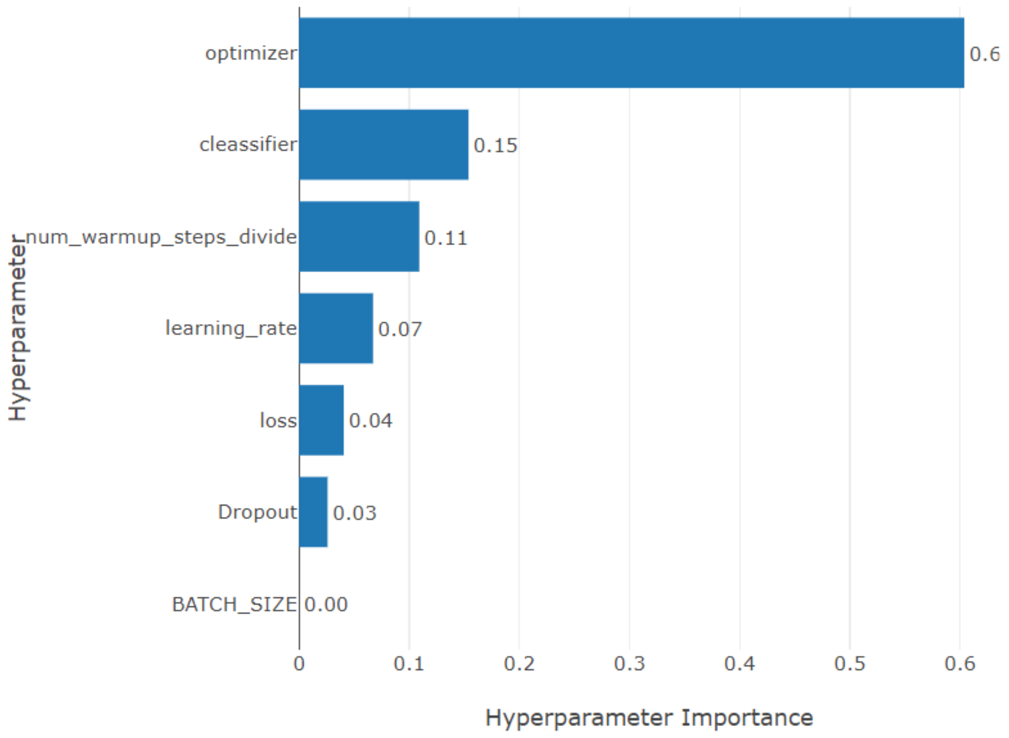

Figure 2: FS-Importanc

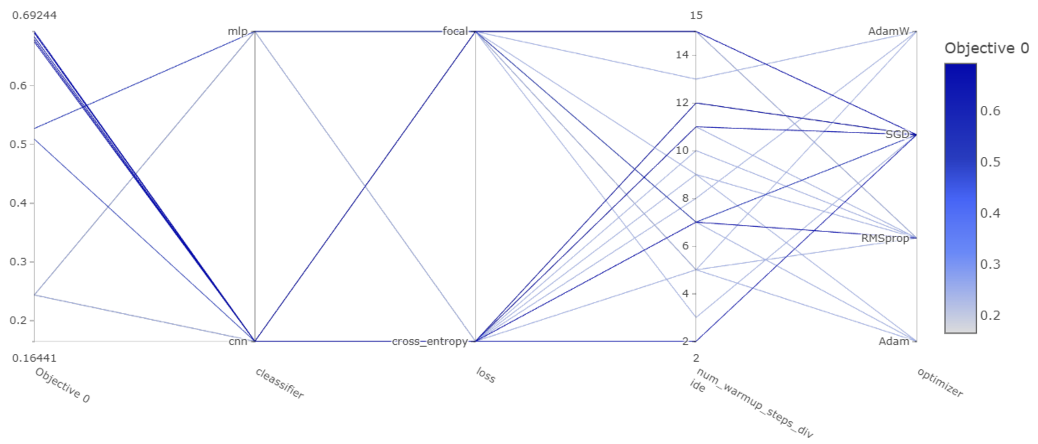

Figure 3: Parallel Coordinate