MNLP Homework: Multiclass Classification of Cultural Items

Valerio Massimo Carioti
Sapienza University of Rome
carioti.1983063@studenti.uniroma1.it

Federico Urbini
Sapienza University of Rome
urbini.2007465@studenti.uniroma1.it

4 May 2025

Introduction

This project addresses the challenge of classifying cultural items to understand how much they are related to a culture, assigning them to one of three classes: *cultural agnostic*, *cultural representative*, or *cultural exclusive*. To do so, we implemented two different models: a transformer-based classifier and a non-LM approach based on multilingual features and graphs.

The first one relies only on textual descriptions, to capture cultural signals directly from language without the need for structured data, while the second uses graphs to represent connections across Wikipedia versions, capturing how the same item is described differently and helping determine its cultural specificity.

Methodology non-LM based

Feature Extraction and Graph Construction

Starting from the original *.csv* dataset, both global and local features were extracted by querying Wikidata and Wikipedia.

We selected the 10 Wikipedia editions that appeared most frequently as *sitelinks* among the items in the dataset. This number was chosen because using fewer resulted in too little data, while including more led to excessive missing information from the less frequent language versions. For each item, we collected the corresponding Wikipedia pages (when available) and extracted local features such as *page length*, *number of images*, *templates*, *interwiki links*, *links to the same page in different languages*, *redirects* and *categories*. Features like these help estimate how widely known or culturally shared an item is.

Then, we built an undirected and unweighted graph, with nodes representing different Wikipedia versions (typically languages), and edges added based on how often two versions co-occur across items. The similarity criteria used is Intersection over Union (IoU) over the binary presence of pages (see Appendix for the formula). This graph structure is independent of the feature set, ensuring stability and enabling the GNN to propagate cultural informations across languages even when some are missing. Each node was populated with the page features from the corresponding language. One graph per item was constructed, all sharing the same shape, and used as GNN input. Those informations were combined with global ones, such as the total number of sitelinks.

To reduce training time and avoid extraction delays, we focused on features that were both informative and quick to collect. This ensures fast prediction at inference time, also in an online environment.

Model Architecture and Optimization

The model combines two types of information: global signals describing the item as a whole, and local features extracted from different versions of Wikipedia. The global features are processed using a fully connected neural network (MLP), while the local ones are handled by a Graph Neural Network (GNN) based on GINConv layers, introduced in the paper *How Powerful Are Graph Neural Networks?* by Xu et al. (2019), enhanced with pre-activation residual connections as proposed in *All You Need to Train Deeper GCNs* by Li, Xiong, Thabet and Ghanem (2020).

In this architecture, each item is represented by a graph and a set of numerical attributes. While the MLP focuses on aggregated data, the GNN captures the structural relationships between Wikipedia editions through message passing across the graph. After being independently processed, the two representations are combined, so that the model can use them together to make better predictions.

This fusion improves the model's ability to detect cultural distinctions, even when some sources are missing or noisy, by combining information from different editions and making use of the content they share.

Experiments non-LM based

We trained the GNN on graphs composed of 10 nodes, each corresponding to a different version of Wikipedia. Given their limited size, we adopted the `GINConv` architecture, known for its effectiveness in small graphs due to its ability to capture subtle structural differences. Initial tests with `GCNConv` produced weaker results, likely because of its lower capacity to model such small variations.

To address potential class imbalance and label noise, we tested several loss functions, including focal loss, symmetrical loss and weighted cross-entropy. However, none outperformed standard cross-entropy, possibly due to label noise in the dataset.

We also experimented with defining edges based on the correlation between Wikipedia page lengths. However, this produced less informative graphs, as most editions closely follow the English Wikipedia, leading to disproportionately high correlation with "enwiki" compared to other editions.

Finally we tried `GINEConv` that extends `GINConv` by including edge features, but it led to no improvement.

Results non-LM based

The custom GraphNet model demonstrated strong and reliable performance, reaching a validation F1 score of 0.739 and an accuracy of 74.4%. The confusion matrix highlights difficulties in correctly identifying the *cultural representative* class, in contrast with the other two classes.

Hyperparameter tuning with Optuna highlighted the learning rate as the most influential factor.

Notably, despite its simplicity and lack of pre-training, the non-LM model performed competitively, only slightly below the Transformer-based approach in both F1 and accuracy. This confirms the effectiveness of combining global and local informations to capture both structural patterns across languages and the cultural context of each item.

Methodology – LM-based

We fine-tuned a Transformer-based model using a concatenation of the name, description and item's English Wikipedia extract, separated by special tokens (<name>, <description>, <enwiki>). This combination provided a richer linguistic context and improved classification performances compared to using each field in isolation. We chose `google/electra-small-discriminator`, which follows the same encoder architecture as BERT but adopts a more efficient pretraining strategy. While standard BERT models rely on masked token prediction, ELECTRA is trained to detect replaced tokens, allowing it to learn from all the positions in the input. This resulted in better performances in our setting, without requiring large-scale computations.

Experiments – LM-based

We initially tested several pre-trained models including `XLM-RoBERTa` and `BERT` using only the raw description field, reaching at most 71% validation accuracy. To improve performances, we switched to a concatenation of the item name, short description, and the English Wikipedia extract.

We noticed that small distilled models like DistilBert generally performed as good as the bigger ones, except for the required training time. However, too small models like `TinyBert` and `microsoft/MiniLM-L12-H384` were, generally, scoring a bit lower. In this scenario, `google/electra-small-discriminator` offered a solution that allowed us to use an efficient model while not losing performances.

We also tried to concatenate the extracts from more languages, but to process this kind of input a large model (such as *BigBird* or *Longformer* that accept as input longer sequences of tokens) is required and this was not computationally practical given our constraints.

Results – LM-based

The final Transformer-based model achieved strong performances, with a validation F1 score of 0.8 and an accuracy of 80.7%, outperforming all previously tested models. These results confirm the effectiveness of combining the Wikipedia extract with the item name and description, as well as the suitability of *ELECTRA-small* for tasks with limited computational resources.

Although the input was limited to 512 tokens, this proved sufficient for capturing cultural signals.

The confusion matrix revealed that most errors occurred between the *representative* and *exclusive* classes, reflecting their conceptual overlap.

Appendix

$$\mathrm{IoU}(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

IoU formula

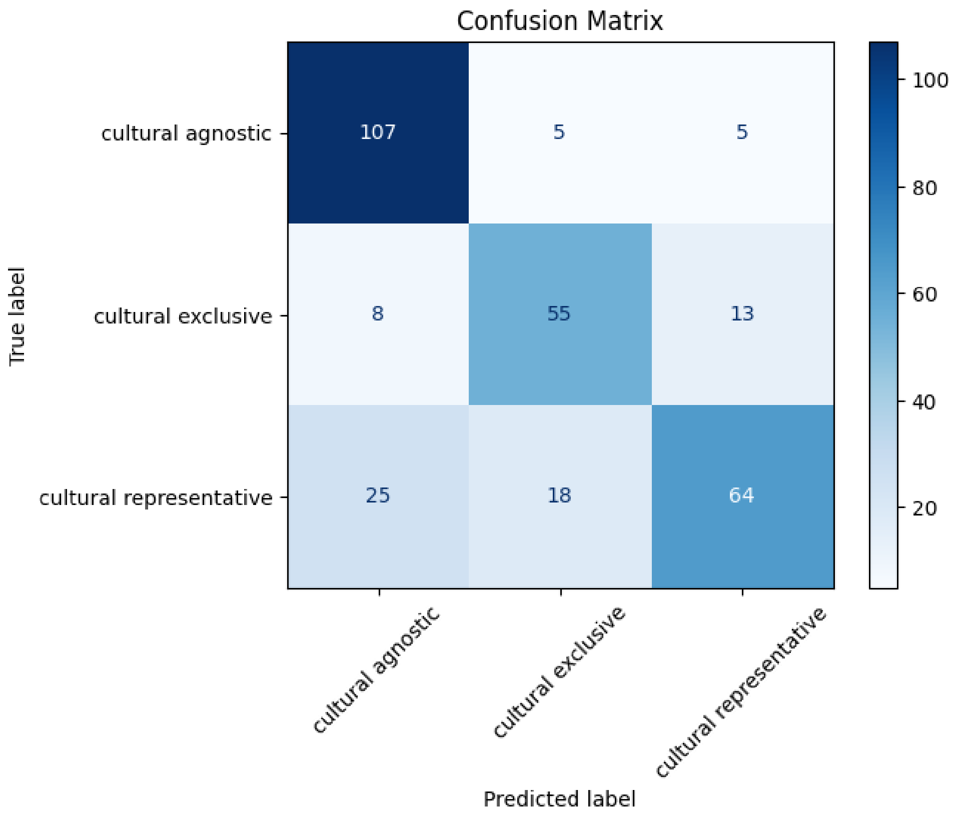

Figure 1: Confusion matrix (Non-LM)

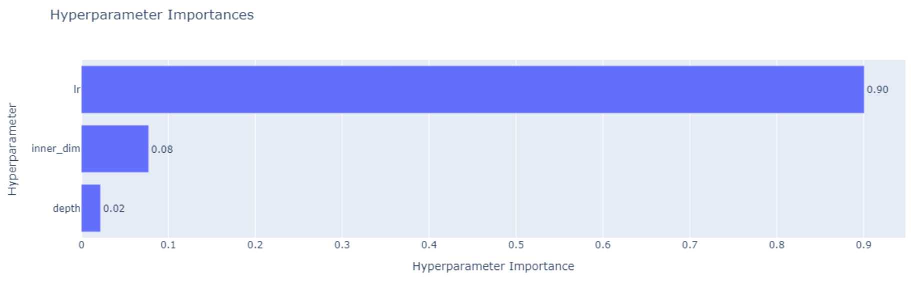

Figure 2: Hyperparameters Importance (Non-LM)

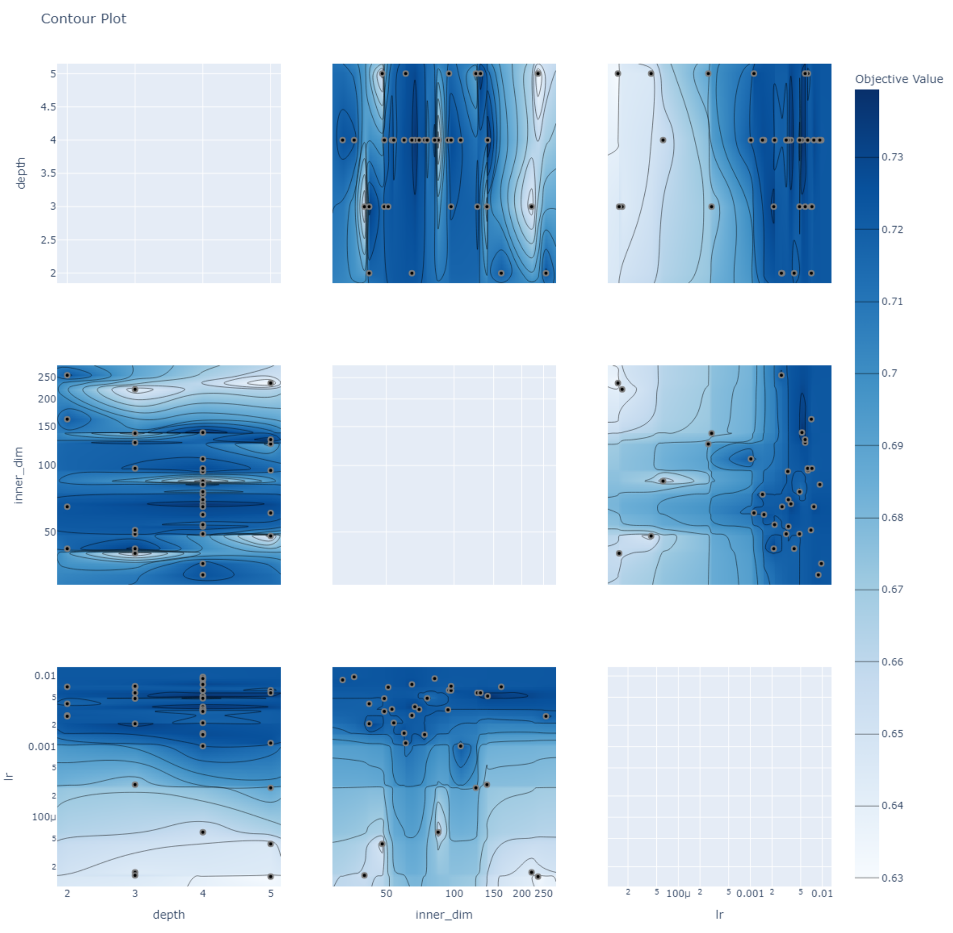

Figure 3: Hyperparameter Interaction (Non-LM)

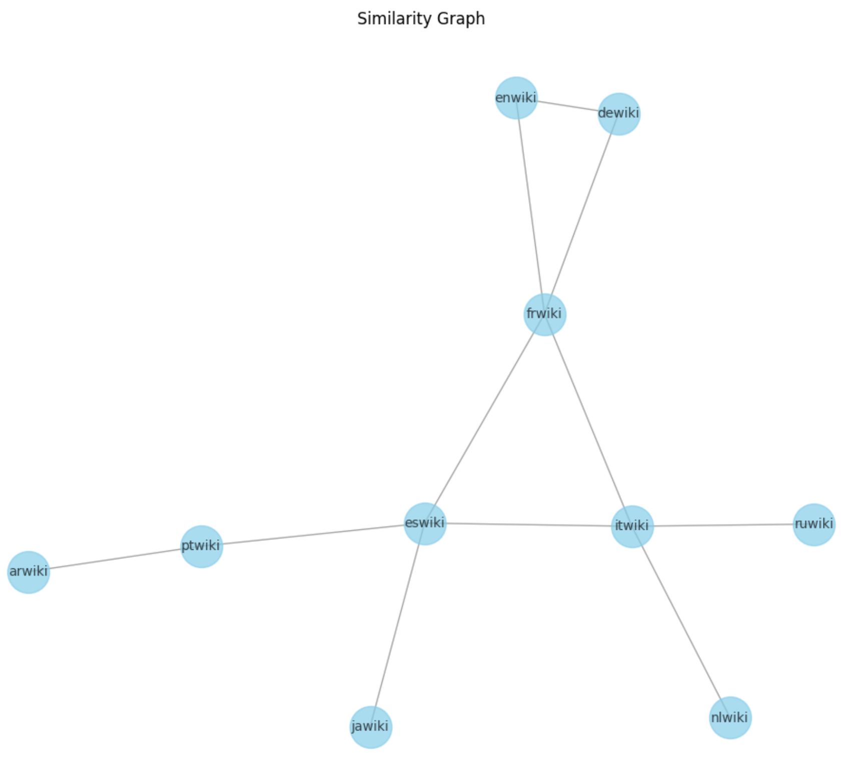

Figure 4: Similarity Graph Based on IoU

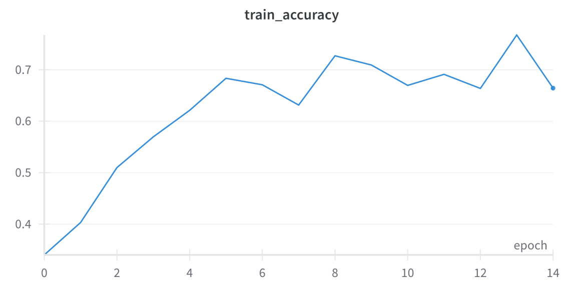

Figure 5: Training accuracy (Non-LM)

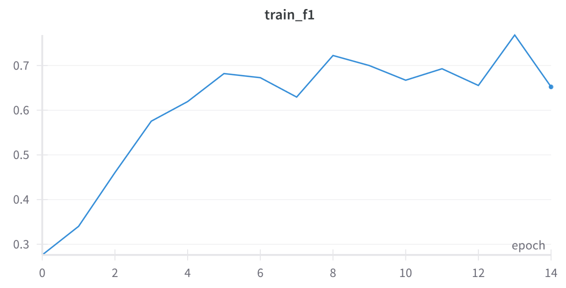

Figure 6: Training F1 score (Non-LM)

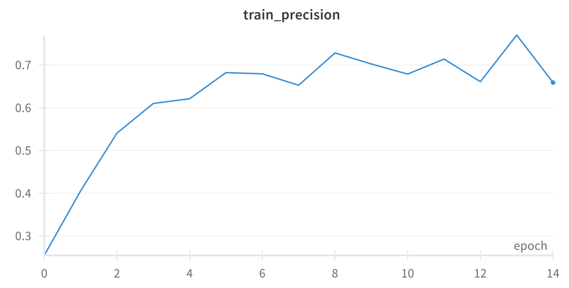

Figure 7: Training precision (Non-LM)

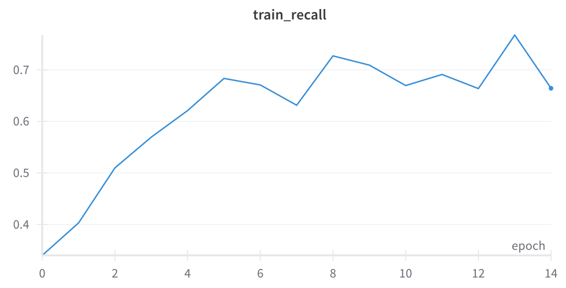

Figure 8: Training recall (Non-LM)

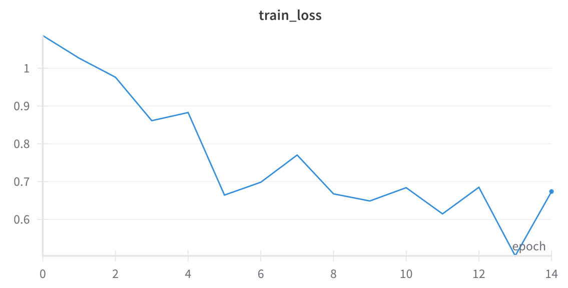

Figure 9: Training loss (Non-LM)

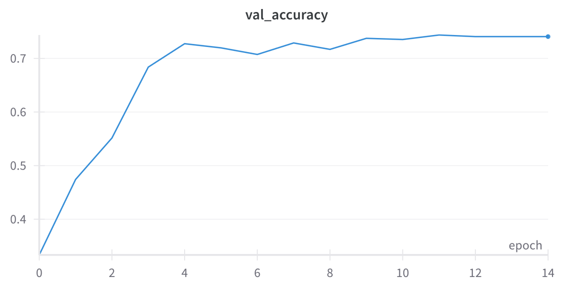

Figure 10: Validation accuracy (Non-LM)

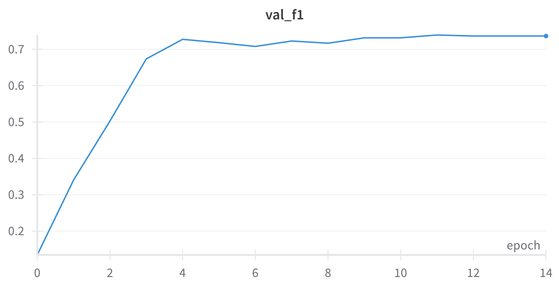

Figure 11: Validation F1 score (Non-LM)

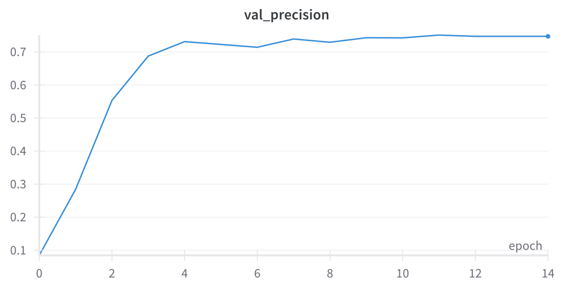

Figure 12: Validation precision (Non-LM)

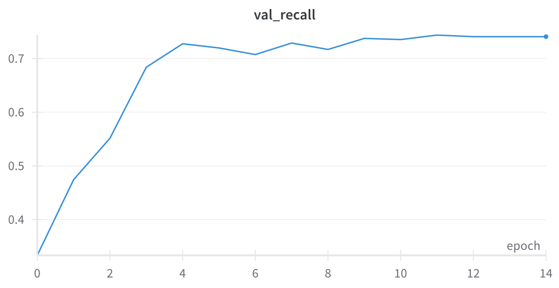

Figure 13: Validation recall (Non-LM)

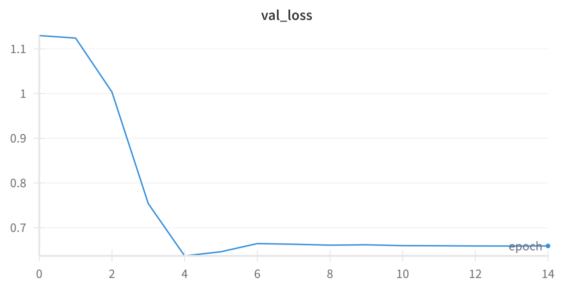

Figure 14: Validation loss (Non-LM)

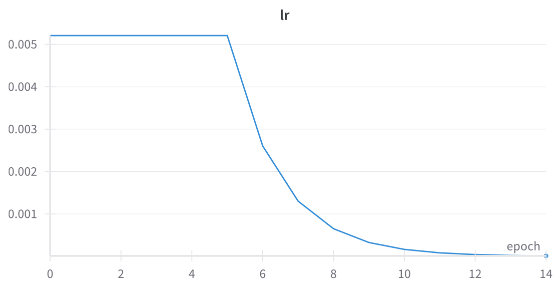

Figure 15: Learning rate (Non-LM)

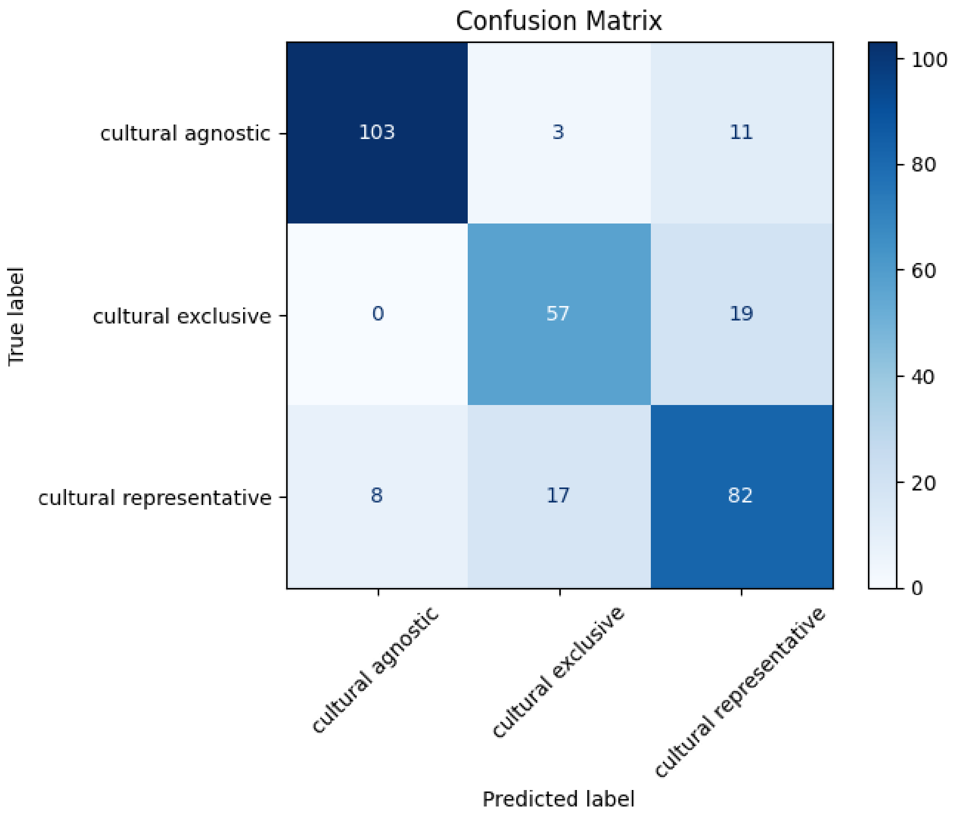

Figure 16: Confusion Matrix (Transformer)

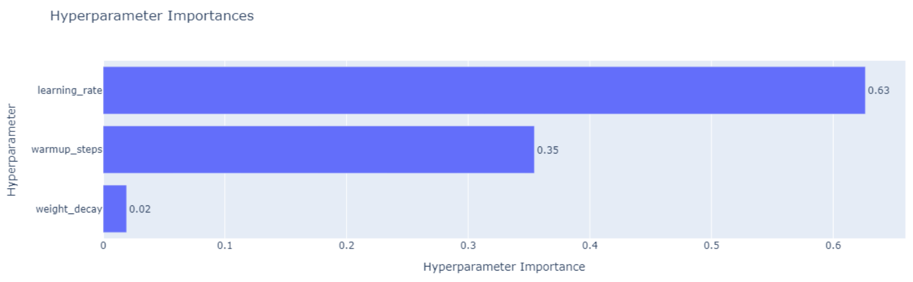

Figure 17: Hyperparameter Importance (Transformer)

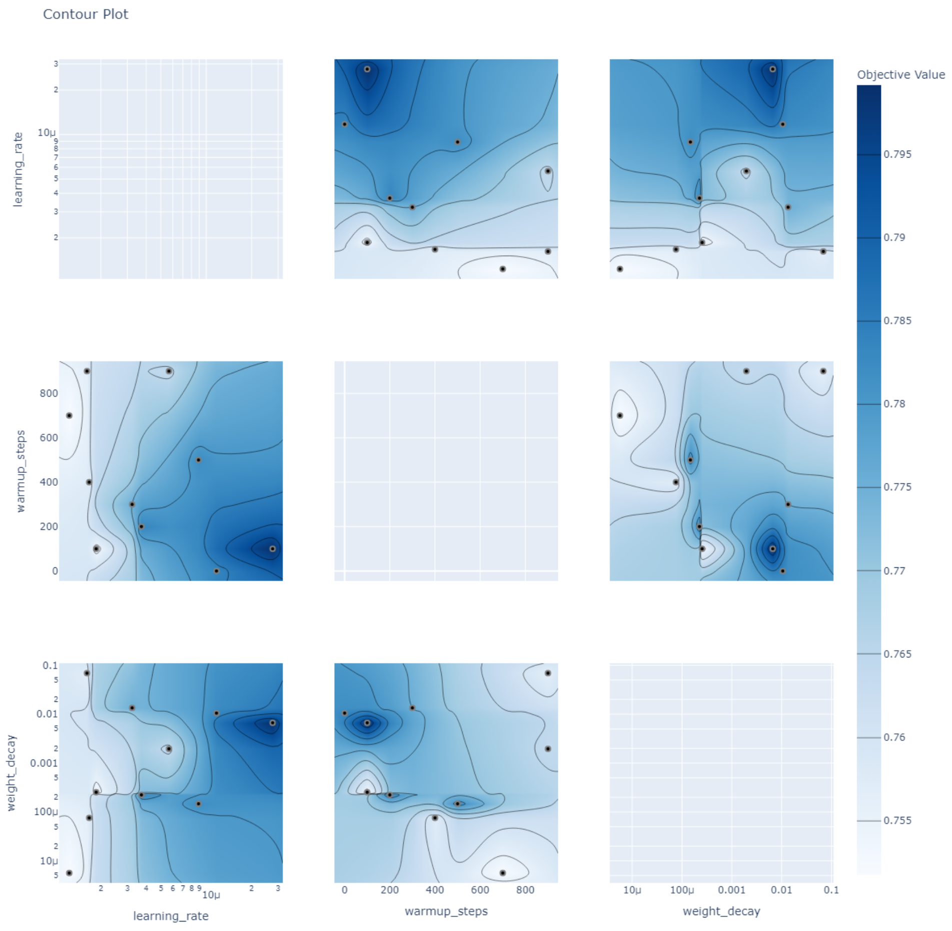

Figure 18: Hyperparameter Interaction (Transformer)

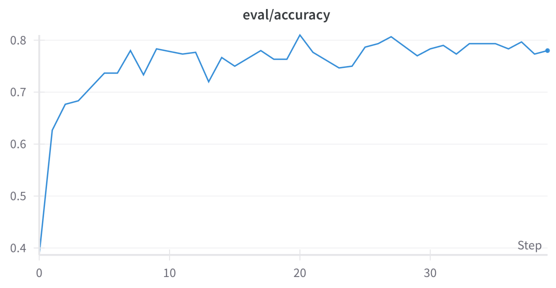

Figure 19: Validation accuracy (Transformer)

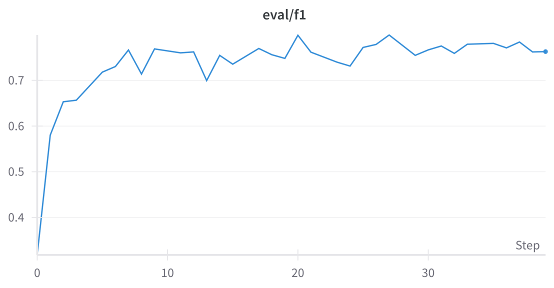

Figure 20: Validation F1 score (Transformer)

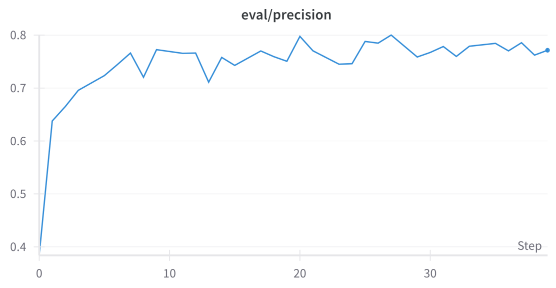

Figure 21: Validation precision (Transformer)

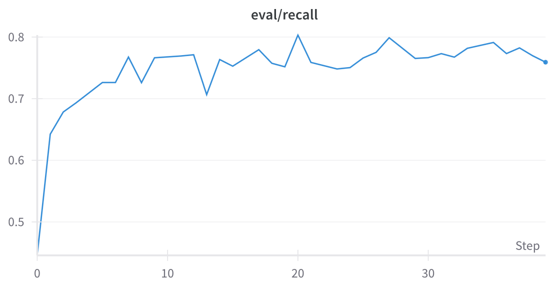

Figure 22: Validation recall (Transformer)

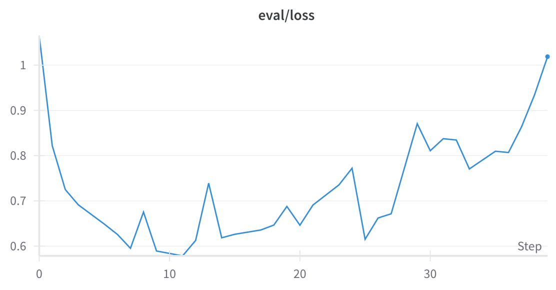

Figure 23: Validation loss (Transformer)

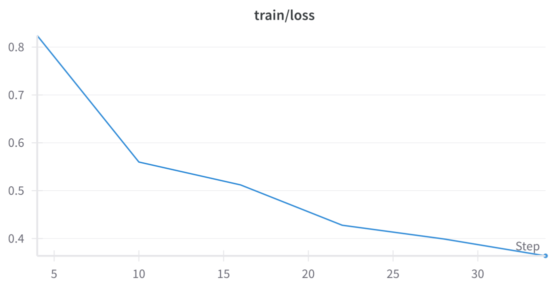

Figure 24: Training loss (Transformer)

References

[1] Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2019). *How Powerful Are Graph Neural Networks?* arXiv preprint arXiv:1810.00826.

[2] Li, G., Xiong, C., Thabet, A., & Ghanem, B. (2020). *DeeperGCN: All You Need to Train Deeper GCNs*. arXiv preprint arXiv:2006.07739.