Custom Classifier Head Architectures for Transformer-based Hate Speech Detection

Julius Voß, Kevin Saiger, Hind Dafallah, Natalia Timokhova, and Masooma Masooma

***Abstract*— Hate speech detection has emerged as a critical task in the field of natural language processing (NLP), especially in social media moderation. This paper presents a comparative analysis of transformer-based language models including BERT, RoBERTa and XLNet, applied to the HASOC 2019 dataset. The dataset was preprocessed to clean and normalize text content. Evaluation was conducted using accuracy, F1 score, and confusion matrices. Among the models tested, the fine-tuned RoBERTa achieved the highest performance with an F1 score of 0.86. This study demonstrates the effectiveness of transformer architectures and highlights the performance benefits of fine-tuning.**

I. INTRODUCTION

With the increasing use of online platforms, hate speech detection has become a vital application of NLP. The HASOC 2019 dataset provides annotated social media posts for identifying offensive content in multiple languages. This study investigates and compares the performance of powerful transformer-based models: RoBERTa (frozen and fine-tuned), BERT, and XLNet (frozen and partial fine-tuning). The primary objective is to build robust classifiers capable of detecting offensive language using deep contextual embeddings. We also analyze the impact of fine-tuning versus using frozen embeddings on classification accuracy and robustness.

II. METHODOLOGY

A. Dataset & Preprocessing

In this work, we use the HASOC dataset, suitable for training models to detect hate speech. We train, validate, and test all presented models on the HASOC 2019 english dataset [1]. The dataset consists of three tasks:

- **Task 1** focuses on hate speech and offensive language offered in multiple languages and classifies text into either *Hate and Offensive* (HOF) or *Non- Hate and offensive* (NOT).
- **Task 2** can be used for more fine-grained classification. It extends Task 1 by classifying offensive texts in *Hate speech*, *Offensive*, *Profane* or none.
- **Task 3** checks the type of the offense. *Hate and Offensive* texts are organized in the categories *Targeted Insult* and *Untargeted*. The first contains posts insulting an individual, group, or others, while the *Untargeted* contains posts with offensive or profane words but not targeted at someone.

We don't directly feed our model with the given dataset as we preprocess it first. In the preprocessing, we remove Twitter handles (i.e., usernames followed by the @ symbol), URLs, hashtags, and special characters (except basic punctuations). Additionally, we convert all texts to lowercase and remove extra whitespaces.

The following figures illustrate the data visualisation and class balances:

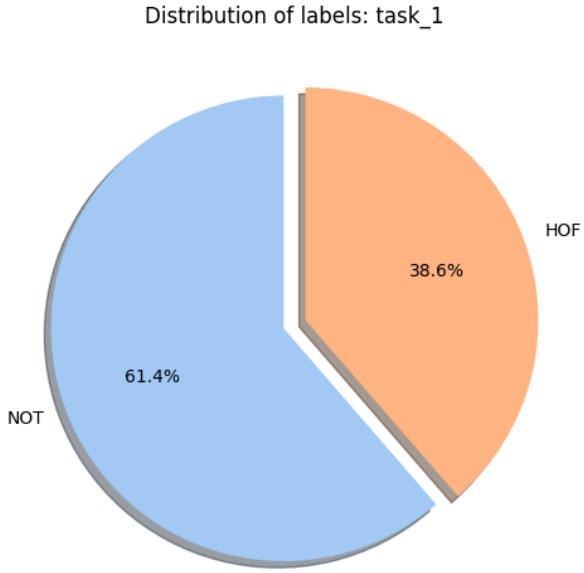

Fig. 1. Label distribution of the HASOC 2019 english dataset for Task 1.

To address the class imbalance present in the HASOC dataset (see Figure 1), we experimented with several techniques. First, we applied class weighting, where we adjusted the loss function to penalize misclassification of underrepresented classes more heavily. Additionally, we explored undersampling of the majority class and oversampling of the minority class to balance the training data. Each method was evaluated in terms of its impact on model performance, particularly with respect the F1-score for the minority class. Among these, class weighting consistently led to the best overall performance and was therefore used in our final model.

III. MODEL ARCHITECTURE

To address the task of binary hate speech detection on the HASOC dataset (Task 1), we designed a modular classification architecture built upon pre-trained transformer-based language models. Our architecture aims to systematically explore the effects of various pooling strategies and classification heads on downstream performance (see Figure).

We hypothesize that relying solely on the standard [CLS] token representation may be insufficient for capturing the full semantic richness of a sequence, and therefore evaluate several alternative approaches.

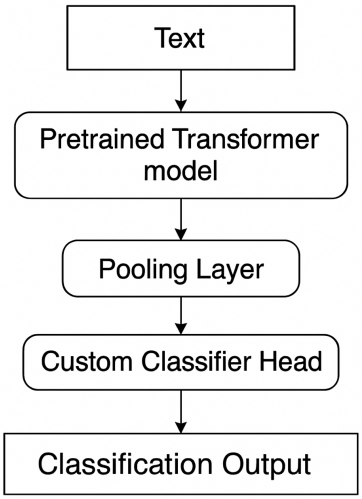

Fig. 2. Modular architecture for testing different model configurations.

A. Pretrained Models

We employ a range of transformer encoders as frozen feature extractors, including BERT-base-uncased, RoBERTa-base, RoBERTa-large, and a RoBERTa variant fine-tuned on Twitter sentiment analysis (cardiffnlp/twitter-roberta-base-sentiment-latest) from the Hugging Face library. We also include XLNet-base-cased as a frozen encoder, using its built-in sequence-summary vector in place of [CLS]. These models provide contextualized token embeddings which are subsequently aggregated and passed to task-specific classifier heads. In some configurations, we partially unfreeze the top 1–4 encoder layers of the base model, which we found beneficial to reduce overfitting and adapt the representations to our modified pooling and classification strategies.

B. Pooling Layer Mechanisms

To condense the variable-length sequence of token embeddings into a fixed-size representation, we experiment with four pooling mechanisms. Given the output of the last hidden layer of a Transformer model as a matrix

$$H \in \mathbb{R}^{T \times d},$$

where T is the sequence length and d the hidden size, and an attention mask

$$M \in \{0,1\}^T,$$

we describe the following pooling strategies used in our models:

- **CLS Pooling:** Standard approach using the [CLS] token embedding at position 0 [4]. CLS pooling simply extracts the embedding of the special classification token:

$$h_{\text{cls}} = H_0$$

assuming that the [CLS] token appears at position $t = 0$.

- **Mean Pooling:** Mean pooling computes the average of the token embeddings, weighted by the attention mask:

$$h_{\text{mean}} = \frac{\sum_{t=1}^{T} M_t \cdot H_t}{\sum_{t=1}^{T} M_t},$$

where $H_t \in \mathbb{R}^d$ is the embedding at time step t.

- **Max Pooling:** Max pooling selects the maximum value for each feature dimension across all tokens:

$$h_{\text{max}}^j = \max \left\{ H_t^j \;\middle|\; M_t = 1,\ t = 1, \ldots, T \right\}, \quad j = 1, \ldots, d$$

- **Attention Pooling:** A learned, context-sensitive mechanism that assigns attention weights to each token embedding, allowing the model to emphasize the most relevant parts of the input sequence [6]. We use a simple two-layer attention network to compute attention scores and a weighted sum of token embeddings.

a) Step 1: Compute intermediate representation:

$$u_t = \tanh(W_1 H_t + b_1)$$

b) Step 2: Compute raw attention scores:

$$s_t = w_2^\top u_t + b_2$$

c) Step 3: Apply masking and softmax with temperature τ:

$$s_t = \begin{cases} s_t, & \text{if } M_t = 1 \\ -\infty, & \text{if } M_t = 0 \end{cases}$$

$$\alpha_t = \frac{\exp(s_t/\tau)}{\sum_{i=1}^{T} \exp(s_i/\tau)}$$

We apply a temperature $\tau > 1$ to the softmax distribution to smooth the attention weights and prevent the model from overfitting to specific tokens.

d) Step 4: Compute attention-weighted sum:

$$h_{\text{attn}} = \sum_{t=1}^{T} \alpha_t \cdot H_t$$

While CLS pooling is commonly used, it is known to represent only a subset of the full input sequence, potentially discarding information spread across tokens [4]. Our attention-based pooling method offers a more expressive alternative by learning to dynamically reweight tokens based on their importance to the classification objective.

C. Costum Classifier Heads

We explore three distinct classifier head architectures attached to the pooled outputs:

- A **Convolutional Neural Network (CNN)** head, which applies 1D convolutions over the sequence of token embeddings (before pooling), followed by global max pooling. This design aims to capture local n-gram patterns and reduce feature dimensionality.

- A **shallow feed-forward network**, consisting of a linear layer followed by a nonlinearity and dropout, designed to serve as a lightweight classifier.
- A **deeper feed-forward network**, which uses multiple hidden layers and higher capacity to model complex decision boundaries.

Each classifier head is trained on top of the frozen (or partially unfrozen) base model, allowing us to isolate and evaluate the effects of the pooling strategy and head architecture independently.

IV. Training Details

All models were implemented using PyTorch and Hugging Face's transformers library. For classification, a dropout layer was followed by a final dense layer with a softmax activation.

We experimented with multiple configurations for each transformer model, including varying the number of training epochs and learning rates to determine the optimal hyperparameters. In addition, we explored two different training strategies for the RoBERTa model: one with frozen encoder layers where only the classification head was trained, and another with full fine-tuning of all model layers. In addition, we applied the same workflow to XLNet in two configurations. In the first configuration (partial fine-tuning), we unfroze the top 2 of 12 transformer layers, used a maximum sequence length of 128 tokens, set the learning rate to 2×10^{-5}, and trained for 20 epochs with class weighting. In the second configuration (full fine-tuning), we unfroze all 12 layers, used a maximum sequence length of 256 tokens, set the learning rate to 2×10^{-6}, and trained for 20 epochs with class weighting. These two XLNet variants allowed us to compare how partial versus full adaptation of the pretrained encoder affects hate-speech classification performance.

Training and evaluation were performed using an 80/20 stratified train-test split to preserve label distribution. To manage and track our experiments efficiently, we used Weights & Biases (WandB), a tool for experiment tracking and visualization. WandB was used to run sweeps across different hyperparameter combinations and to log key metrics such as training/validation loss, accuracy, F1 scores, and confusion matrices. The platform enabled a streamlined comparison between models and configurations, helping us identify the best-performing setup.

V. Results

A. Roberta results

Fig 1 and 2 shows the validation accuracy and loss for the Roberta model., when it is fine tuned to get the best results. The table shows that we achieved F1 equals 0.89. which is pretty good.

TABLE I
ROBERTA TEST RESULTS

F1	accuracy
0.89	0.83

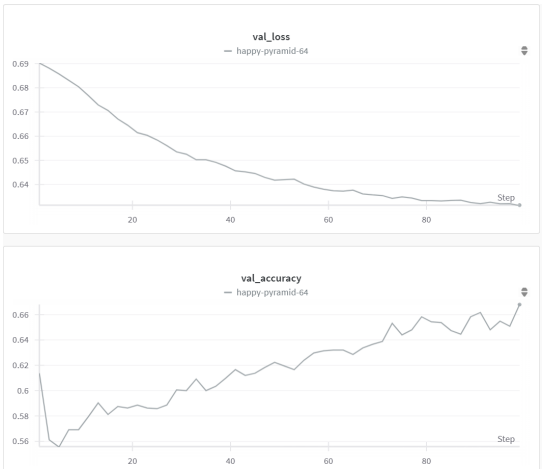

Fig. 3. Validation accuracy and loss for the Roberta transformer.

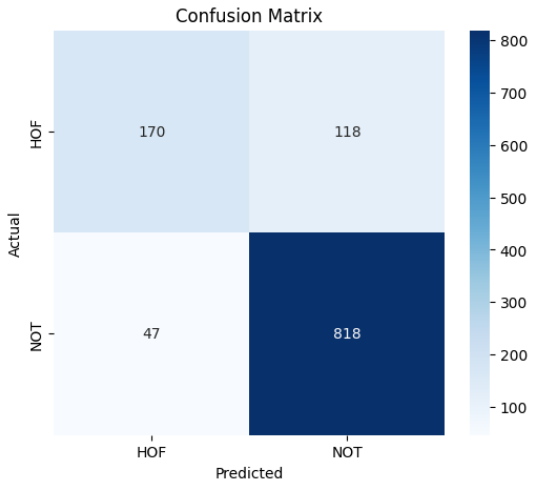

Fig. 4. Confusion matrix of the RoBERTa base model after finetuning.

B. Bert results

Fig 3 and 4 shows the BERT results , when it is fine tuned to achieve the best results , the performance is good but slightly lower than Roberta model results.

TABLE II
BERT TEST RESULTS

F1	accuracy
0.72	0.77

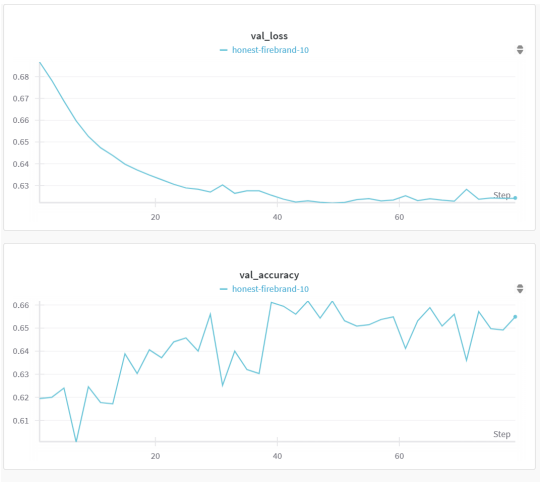

Fig. 5. Validation accuracy and loss for the BERT transformer.

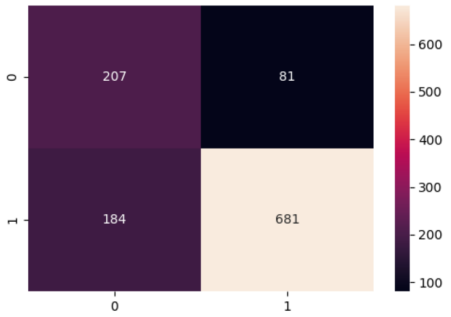

Fig. 6. Validation accuracy and loss for the BERT transformer.

C. XLNet results

Figure 7 plots the validation accuracy curves for both XLNet variants, with the green line indicating the stronger configuration. Although XLNet underperforms RoBERTa in terms of macro-F1 (see Table III), the accompanying confusion matrices (Figures 8, 9) reveal a more nuanced breakdown of its predictions - highlighting which classes the model handles well and where most errors occur.

TABLE III
XLNET TEST RESULTS FOR PARTIAL VS. FULL FINE–TUNING

Configuration	F1	Accuracy
Partial FT (top 2/12 layers, max_len=128)	0.6512	0.6982
Full FT (all 12/12 layers, max_len=256)	0.5277	0.5317

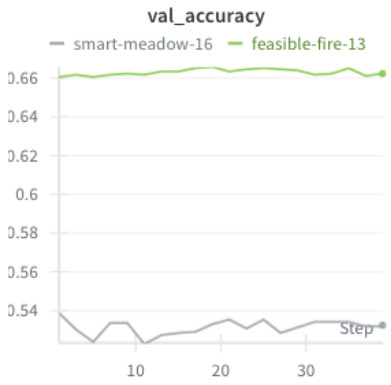

Fig. 7. Validation accuracy for the XLNet transformer.

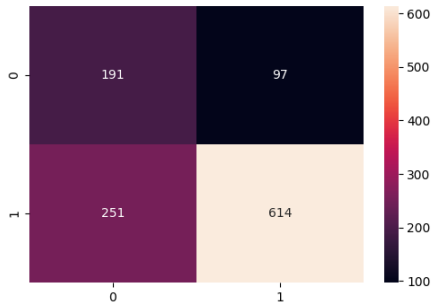

Fig. 8. Confusion matrix for XLNet partial fine-tuning (top 2 layers).

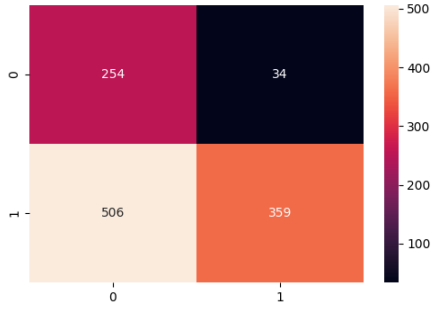

Fig. 9. Confusion matrix for XLNet full fine-tuning (all layers).

VI. Discussion

In addition to our two RoBERTa variants, we also evaluated XLNet in both partial and full fine-tuning configurations. All four models were trained and tested on the same HASOC 2019 English Task 1 split, using class-weighted loss, identical data preprocessing, and evaluation via Accuracy, Precision, Recall and macro-F1.

Despite XLNet's more sophisticated pretraining, both of its configurations consistently underperformed compared to RoBERTa. The partial fine-tuning setup (only the top two layers unfrozen, 128-token inputs) failed to adapt most of XLNet's 110 M parameters to the hate-speech domain. The full fine-tuning run (all layers, 256-token inputs) suffered from an overly aggressive learning-rate schedule that decayed to near zero before convergence, and its custom CNN pooling head did not fully leverage XLNet's built-in sequence-summary output. As a result, XLNet's macro-F1 and overall accuracy remained below those of even the Twitter-specific RoBERTa model.

Consequently, we focus our detailed comparison on the two RoBERTa variants, which demonstrated the strongest performance on HASOC Task 1.

Evaluation Metrics Comparison

Metric	roberta-base	cardiffnlp/twitter-roberta-base-sentiment-latest
Loss	0.3989	0.4008
Accuracy	0.8569	0.8395
Precision	0.8739	0.8556
Recall	0.9457	0.9457
F1 Score	0.9084	0.8984

Fig. 10. Test set Accuracy and macro-F1 for the two RoBERTa variants.

As shown above, RoBERTa base slightly outperforms the Twitter-fine-tuned RoBERTa across all metrics, with identical recall for both models.

Confusion Matrix for RoBERTa Base

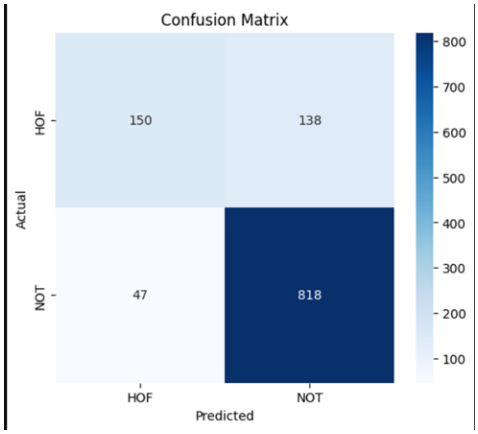

Fig. 11. Confusion matrix of the RoBERTa base model after fine-tuning.

VII. Conclusion

This study validates the effectiveness of modern transformer architectures for hate speech detection. Overall, fine-tuning transformer encoders significantly boosts F1 (exceeding 0.85 for RoBERTa), while frozen embeddings alone underperform. BERT remains competitive but below RoBERTa. XLNet's architectural advantages did not translate on this small, short-text task: partial freezing underfits, and full fine-tuning overfits under the chosen schedule. These findings underscore that model choice and hyperparameter tuning must align with dataset characteristics; for HASOC Task 1, RoBERTa proved the most robust and efficient.

References

[1] https://hasocfire.github.io/hasoc/2019/

[2] T. Mandl, S. Modha, C. Mandlia, D. Patel, A. Patel, and M. Dave, "Overview of the HASOC Track at FIRE 2019: Hate Speech and Offensive Content Identification in Indo-European Languages," in *Working Notes of FIRE 2019 - Forum for Information Retrieval Evaluation*, Kolkata, India, Dec. 2019.

[3] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," arXiv preprint arXiv:1907.11692, Jul. 2019.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proc. of the 2019 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, Minneapolis, MN, USA, Jun. 2019, pp. 4171–4186.

[5] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized Autoregressive Pretraining for Language Understanding," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, Vancouver, Canada, Dec. 2019, pp. 5753–5763.

[6] Z. Lin, M. Feng, C. N. dos Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A structured self-attentive sentence embedding," in *Proc. of the 5th International Conference on Learning Representations (ICLR)*, 2017. [Online]. Available: https://arxiv.org/abs/1703.03130