PPO-CIS: A Deep Reinforcement Learning Framework for Real-Time Toxicity Detection in Social Media

Arezo Bodaghi, Benjamin C. M. Fung, Ketra A. Schmitt

[a]*Concordia Institute for Information Systems Engineering, Concordia University, Montreal, Canada, H3G 1M8*
[b]*School of Information Studies, McGill University, Montreal, Canada, H3A 1X1*
[c]*Centre for Engineering in Society, Concordia University, Montreal, Canada, H3G 1M8*

Abstract

Online platforms face growing challenges in moderating harmful user-generated content due to the large volume and rapid pace of interactions. In existing moderation systems, automated tools assist human moderators, yet they often struggle to balance processing efficiency and reliable classification. When moderation fails to detect harmful content quickly and accurately, platforms risk user harm and noncompliance with content safety regulations. This paper proposes an adaptive moderation method named the Proximal Policy Optimization-based Cascaded Inference System (PPO-CIS). The method integrates multiple toxicity classifiers into a cascaded decision architecture guided by deep reinforcement learning. At each step, PPO-CIS selects the next classifier according to the content difficulty and the expected gain in accuracy relative to computational cost. The system enables rapid filtering of benign content and only activates high-capacity models for uncertain cases. PPO-CIS is the first toxicity detection framework to employ Proximal Policy Optimization for real-time optimization of classifier cascades. Experiments on the AugmenToxic and ToxiGen datasets show that PPO-CIS improves detection accuracy by 2.10 percent while increasing processing speed from 42.74 to 384 samples per second compared with static cascade designs. The findings show that adaptive model selection can better shield users from exposure to harmful content while lowering moderation costs. PPO-CIS provides a practical solution for deploying scalable and timely content moderation in fast-moving online environments.

Keywords: Toxic Language Detection, Cascaded Classifiers, Dynamic Inference System, Real-Time Content Moderation

1. Introduction

Online platforms rely heavily on automated systems to moderate toxic content at scale, yet effective moderation remains difficult due to the volume, speed, and linguistic diversity of online communication (Gorwa et al., 2020; Waseem et al., 2017; Bodaghi et al., 2023). Automated classifiers frequently misclassify content, either removing benign content or failing to detect harmful content. These errors increase the workload of human moderators and can undermine user trust in platform governance (Ganesh and Bright, 2020; Dong et al., 2020). Regulatory frameworks such as the German NetzDG and the EU Code of Conduct on Hate Speech further intensify these pressures by imposing strict requirements on the timely removal of illegal or harmful content (Gorwa et al., 2020).

Recent advances in large language models (LLMs) have improved the contextual understanding of toxicity; however, deploying LLMs in large-scale, real-time moderation pipelines remains problematic due to their computational cost, inference latency, and limited interpretability (Cirillo et al., 2025). Cascaded and ensemble classifiers have been proposed as a way to address this trade-off by combining multiple models with different computational costs (Polikar, 2012; Chen et al., 2012; Viola and Jones, 2001; Paisitkriangkrai, 2011). In cascaded architectures, inexpensive classifiers process most inputs, while more complex models are reserved for uncertain or high-risk cases. This approach can significantly reduce computational overhead while preserving detection performance, making it particularly suitable for real-time toxicity detection (Viola and Jones, 2001; Zhang et al., 2016). However, most existing cascaded systems rely on static decision rules that do not adapt to shifting data distributions, evolving toxicity patterns, or changing computational constraints. Beyond these computational and operational issues, privacy concerns are increasingly tied to content moderation, as personal data shared on social platforms can be misused to fuel toxic interactions (Cirillo et al., 2023; Cerruto et al., 2022). Addressing toxicity detection therefore requires not only accurate classifiers, but also adaptive systems that can balance accuracy, latency, and cost under real-world conditions. To address these challenges, we propose PPO-CIS, a Proximal Policy Optimization-based Cascaded Inference System for toxicity detection. PPO-CIS formulates classifier selection as a sequential decision-making problem and uses deep reinforcement learning

(DRL) to dynamically choose which model in a cascade should process each input. High-throughput classifiers are applied first to handle straightforward cases, while more accurate but computationally expensive models are invoked only when classification confidence is low. A custom reward function jointly penalizes misclassification and inference latency, enabling the system to optimize detection quality while respecting computational constraints. By learning from ongoing feedback, PPO-CIS adapts its inference policy to changing content characteristics and operational conditions.

To the best of our knowledge, PPO-CIS is the first toxicity detection framework to apply Proximal Policy Optimization (Schulman et al., 2017b) to dynamic cascade optimization. Experimental results on the AugmenToxic and ToxiGen datasets show that PPO-CIS improves detection accuracy by approximately 2.10% and increases throughput from 42.74 to 384 samples per second compared with static cascaded baselines. When evaluated against CETRA, a state-of-the-art reinforcement learning-based cascaded inference system originally designed for malware detection (Lavie et al., 2023), PPO-CIS achieves a 0.37% improvement in accuracy and more than a 107% increase in throughput. These results demonstrate that adaptive classifier selection can substantially reduce moderation costs while maintaining strong safety guarantees and limiting user exposure to harmful content.

The main contributions of this paper are summarized as follows:

- An adaptive multi-stage inference system that dynamically selects classifiers based on prediction confidence and processing cost.
- A PPO-based optimization strategy that guides inference within cascaded classifiers for efficient decision-making.
- A custom DRL reward function that integrates latency and misclassification penalties for cost-aware optimization.
- Extensive evaluations on two benchmark datasets demonstrating superior accuracy and throughput over state-of-the-art baselines.
- A scalable and cost-effective solution for automated toxicity detection that reduces human workload and enhances user safety.

The remainder of this article is organized as follows. Section 2 reviews related literature and prior studies. The proposed methodology is described in Section 3. Section 4 details the experimental setup, and Section 5 presents and analyzes the findings. Finally, Section 6 discusses the implications of the results, Section 7 outlines future work, and Section 8 concludes the paper.

2. Background and Related Work

The rapid increase in user-generated online content has led to extensive research on automated toxicity detection. Existing work spans several methodological directions that vary in accuracy, scalability, and adaptability to changing online behavior. In this section, we review four major approaches: (i) deep learning–based models, (ii) ensemble learning strategies, (iii) reinforcement learning methods applied directly to toxicity detection, and (iv) reinforcement learning frameworks designed for adaptive model management. These approaches provide the conceptual foundation for our proposed system while also revealing performance and efficiency gaps that motivate our work.

2.1. Deep Learning-based Strategies

Deep learning methods, including neural networks and transformer-based architectures, have demonstrated strong capabilities in toxic language detection by capturing nuanced linguistic patterns and contextual dependencies (Androcec, 2020). These models reduce reliance on manual feature engineering by automatically learning representations from raw text, enhancing detection accuracy and robustness across varied datasets (Maslej-Krešňáková et al., 2020; Museng et al., 2022). A wide range of deep learning architectures has been evaluated for toxicity detection (Jahan and Oussalah, 2023), including Convolutional Neural Networks (CNNs) (O'Shea and Nash, 2015), Long Short-Term Memory networks (LSTMs) (Hochreiter and Schmidhuber, 1997), Bidirectional LSTMs (Bi-LSTMs) (Graves and Schmidhuber, 2005), Gated Recurrent Units (GRUs) (Cho et al., 2014), and transformer-based models such as BERT (Devlin et al., 2019). Among these, CNNs paired with character-level embeddings have achieved notable performance, outperforming word-level counterparts in several cases (Malik et al., 2021). Recent work has also explored hybrid deep learning models that integrate multiple architectures to improve accuracy and efficiency (Alkomah and Ma, 2022; Beniwal and Maurya, 2021). While these approaches often yield promising results, they may face challenges in generalizing to small or imbalanced datasets, indicating that architectural complexity does not always guarantee better performance (Museng et al., 2022).

2.2. Ensemble Learning-based Strategies

Ensemble learning improves classification by integrating multiple models, often surpassing single classifiers in performance, generalizability, and robustness, especially in complex tasks such as toxicity detection (Poojitha et al., 2023). Common ensemble strategies such as bagging (Breiman, 1996), boosting (Schapire, 1990), and stacking combine learners to reduce variance and increase predictive stability.

A range of ensemble approaches has been applied to toxic content detection. These include Recurrent Neural Network (RNN)-based ensembles that incorporate user and content features (Pitsilis et al., 2018), word2vec-based classifiers combined via logistic regression meta-learners (Aljero and Dimililer, 2021), and multi-view stacked models that address unintended gender bias in classification (Nascimento et al., 2022). Other techniques, such as parallel bagging, A-stacking, and random subspace methods, have shown promise for enhancing throughput and fault tolerance in large-scale data processing (Agarwal et al., 2023). Frameworks such as DeL-haTE combine CNN-GRU layers to tackle class imbalance (Melton et al., 2020), while Stacked Weighted Ensembles (SWE) have demonstrated the benefits of blending deep learners (Kokatnoor and Krishnan, 2020). Despite strong results on benchmarks, most methods apply the full ensemble to every sample, leading to high computational cost and poor scalability for real-time systems (Lavie et al., 2023). Many models prioritize accuracy but overlook throughput, which is an essential metric in moderation pipelines. Prior studies comparing BERT-based and CNN-based classifiers confirm that while transformers yield better accuracy, they struggle with throughput on high-volume platforms (Bodaghi et al., 2025). Moreover, platforms including Facebook and X (formerly Twitter) continue to lack real-time moderation, often responding to toxicity only after it has been published (Anjum and Katarya, 2024). This emphasizes the need for systems that balance accuracy with high-speed processing. Several works highlight the importance of integrating machine learning models with human moderators. Research shows that even highly accurate models do not always perform optimally when used in isolation, especially in content moderation scenarios where human oversight plays a critical role (Bansal et al., 2021; Kivlichan et al., 2021). Collaboration between humans and AI, particularly when guided by model uncertainty, can significantly enhance moderation outcomes while reducing cognitive load.

The literature suggests that employing a tree of classifiers is an effective approach for achieving high-throughput and accurate data processing. A "one-class-at-a-time" approach (Singh et al., 2018), a multistage cascading classification technique, was proposed for triaging psychiatric patients using machine learning on textual patient records. This method classifies one class at a time and uses the most accurate classifier at each stage. It outperformed traditional multiclass classifiers, achieving overall high accuracy rates for individual classes. This approach helps reduce expert effort and serves as decision support for triaging psychiatric patients based on the severity of their condition. However, this approach is not high-throughput enough to process large volumes of data in seconds. Employing multiple learning models for a single task is a common practice across different fields, enhancing classification accuracy but also presenting challenges such as increased processing time, higher costs, and scalability issues (Birman et al., 2022). Different strategies have been proposed to address these issues, such as using an AdaBoost-based algorithm and a cascading approach for rapid and accurate visual object detection, which prioritizes computational resources on relevant areas while ignoring backgrounds (Viola and Jones, 2001).

Overall, ensemble methods remain valuable for improving performance in toxicity detection, but real-time applicability requires dynamic systems that optimize inference cost, accuracy, and response time.

2.3. Reinforcement Learning Approaches in Online Toxicity Detection

Reinforcement Learning (RL) is a computational approach for automating goal-directed learning through interaction with dynamic environments (Sutton, 2005; Wang et al., 2018). In this paradigm, an agent explores its environment, taking actions and receiving feedback in the form of rewards or penalties (Kaelbling et al., 1996). Through trial and error, the agent learns a policy that aims to maximize cumulative rewards over time (Arulkumaran et al., 2017; Sutton and Barto, 2018), effectively addressing complex sequential decision-making problems (Li et al., 2018).

When state and action spaces become large, exact reward estimation becomes computationally infeasible (Andriotis and Papakonstantinou, 2019). To address this, neural networks are often used to approximate value functions, enabling scalability in complex environments (Mnih et al., 2016; Silver et al., 2016). The integration of RL with deep learning, known as Deep Reinforcement Learning (DRL) has significantly expanded the applicability of RL in high-dimensional spaces (Goodfellow et al., 2016; LeCun et al., 2015; Arulkumaran et al., 2017). DRL has demonstrated success in diverse areas, including malware detection (Binxiang et al., 2019) and various Natural Language Processing (NLP) tasks such as dialogue systems, translation, and text generation (Dhingra et al., 2017; Bahdanau et al., 2017; Xia et al., 2016). In toxicity detection, DRL has recently been employed to optimize fine-tuned Large Language Models (LLMs) for generating or filtering toxic content (Faal et al., 2023; Hartvigsen et al., 2022). Earlier work explored Deep Q-Learning to detect toxicity in online conversations, achieving performance comparable to traditional models despite the novelty of the approach (Singh, 2020). Other efforts include Q-Bully, which combines RL and NLP to identify cyberbullying across platforms (Aind et al., 2020), and ConBERT-RL, which integrates BERT representations into a policy-driven framework for detecting homophobia and transphobia in transliterated low-resource languages (Raj et al., 2024). This latter approach notably enhances classification accuracy by capturing nuanced linguistic patterns specific to transliterated text.

These contributions highlight the growing relevance of RL for adaptive toxicity detection; foundational concepts of RL and DRL are explained in (Appendix A) to support readers seeking further technical context.

2.4. Reinforcement Learning for Adaptive Ensemble Model Management

Recent advances in DRL have shown promise for managing ensemble classifiers efficiently, particularly under real-time and resource-constrained conditions. Notable examples include Security Policy Implementation using Reinforcement Learning (SPIREL) (Birman et al., 2022) and Cost-Effective Transfer of Reinforcement Learning Policies (CETRA) (Lavie et al., 2023), both designed for malware detection, but applicable more broadly to adaptive classifier selection. SPIREL adopts an Actor–Critic architecture (explained in subsection 10.1), where the reward function accounts for both classification outcomes and computational cost (e.g., runtime). This design enables the system to learn policies that balance accuracy and efficiency, making it viable for deployment in high-throughput, real-time settings. Building on this, CETRA introduces customizable performance targets such as memory usage, inference time, or AUC, and dynamically adjusts its reward function to meet those goals. This flexibility allows CETRA to optimize performance across diverse operational constraints. Both frameworks exemplify how DRL

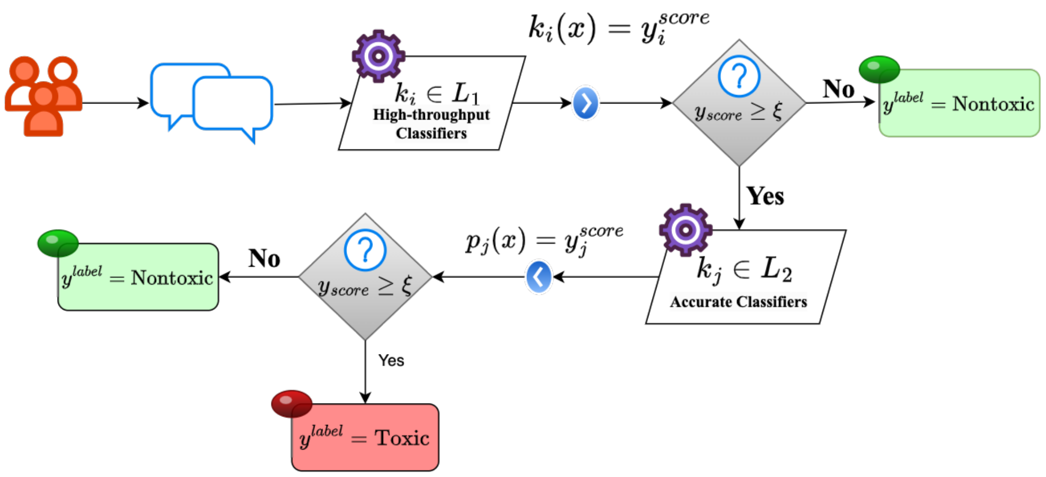


Figure 1: Architecture of Multi-Stage Cascade Inference Systems

can enhance resource allocation and inference efficiency by learning to select classifiers sequentially rather than exhaustively applying the entire ensemble.

While deep learning and ensemble approaches have improved toxicity detection accuracy, they lack efficient mechanisms for dynamic resource allocation. Existing RL systems demonstrate potential for adaptive decision-making but have not been yet integrated into real-time moderation pipelines for toxicity detection. This gap, where high-accuracy models do not scale efficiently and adaptive RL frameworks have not yet been applied to toxicity detection, directly motivates the development of our proposed approach. To address this, we introduce PPO-CIS, a reinforcement-learning-driven cascaded inference system that dynamically selects classifiers based on content complexity and runtime conditions, enabling both high accuracy and high throughput.

3. Methodology

In this section, we present the structure of our proposed approach. We begin by constructing a multi-stage cascade of classifiers tailored for toxicity detection (Section 3.1). To optimize the performance of these cascades, we employ Proximal Policy Optimization (PPO) combined with a specially designed reward function (Section 3.2). This methodology aims to achieve higher throughput and more accurate classification of toxic content.

3.1. Cascaded Inference Systems

The proposed multi-stage cascade inference system is designed to efficiently and accurately detect toxic content in real-time social media applications. This system utilizes a hierarchical approach with multiple stages of classifiers, where each stage is tailored to balance speed (throughput) and accuracy. The architecture of the proposed CIS is illustrated in Figure 1. The initial stage employs high-throughput classifiers to process the bulk of the data quickly, while the subsequent stages use more accurate classifiers to refine the detection of toxic content. This system can effectively handle the massive volume and velocity of UGC.

Let $K = \{k_1, k_2, \ldots, k_n\}$, where $n \in \mathbb{N}$, represent a set of classifiers developed for toxic content detection. Each classifier $k_i \in K$ is evaluated using a tuple of performance and efficiency metrics: accuracy (α), precision (ρ), recall (σ), F1-score (ϕ), throughput (ψ), individual instance latency (λ_q), average latency per classifier (λ), and additional processing cost (ω). The additional cost ω_{k_i} reflects the computational overhead incurred by classifier k_i, including CPU/GPU usage, memory consumption, and monetary cost (e.g., instance time). Together, these metrics form the evaluation tuple:

$$(\alpha_{k_i}, \rho_{k_i}, \sigma_{k_i}, \phi_{k_i}, \psi_{k_i}, \lambda_{q_{k_i}}, \lambda_{k_i}, \omega_{k_i})$$

which enables a comprehensive assessment of the effectiveness and cost-efficiency of the classifier for real-time toxicity detection. Once the classifiers are developed, we select $k_i \in K$ based on their ψ, α, and ϕ. We consider classifiers with high throughput, high accuracy, and a balance between accuracy and throughput. This results in three sets of classifiers including Most Accurate classifiers (V_{acc}), Moderately Accurate classifiers that offer a balanced trade-off between throughput and accuracy (V_{mod}) and High-Throughput classifiers(V_{fast}):

$$V_{\text{acc}} = \{k_i \in K \mid \max(\alpha_{k_i})\}$$

$$V_{\text{mod}} = \{k_j \in K \mid \text{moderate}(\alpha_{k_j}) \text{ and } \text{moderate}(\psi_{k_j})\}$$

$$V_{\text{fast}} = \{k_k \in K \mid \max(\psi_{k_k})\}$$

where $V_{\text{acc}} \cap V_{\text{mod}} \cap V_{\text{fast}} = \emptyset$.

In social media applications, real-time processing is critical, as users expect to see their content quickly (Kirkpatrick, 2016; Urbaniak et al., 2022). Therefore, in the first stage, we use classifiers from $L_1 = V_{\text{fast}} \cup V_{\text{mod}}$ due to their higher throughput. When the CIS receives a set of samples $D = \{x_1, x_2, \ldots, x_z\}$, with $z \subseteq \mathbb{N}$, including toxic and nontoxic samples, at least one classifier $k_i \in L_1$ will be employed to generate a toxicity score (y_i^{score}). The nontoxic samples $D_{\text{nontox}} \subseteq D$ are those with a high probability of being nontoxic and y_i^{score} is below a predefined threshold (ξ), while $D_{\text{tox}} = D \setminus D_{\text{nontox}}$ contains potentially toxic samples when the y_i^{score} meets or exceeds the (ξ). If necessary, another classifier $k_j \in L_1$ ($i \neq j$) will be applied to samples in D_{nontox} to confirm their status, ensuring that nontoxic samples are correctly identified and exit the process. Toxic samples are then forwarded to the second stage.

In the second stage, classifiers from $L_2 = V_{\text{acc}} \cup V_{\text{mod}}$ are used. While the set V_{mod} contains multiple classifiers that offer a balanced trade-off between accuracy and throughput, different members of this pool are allocated to L_1 and L_2 according to the role of each stage. To ensure clear separation of responsibility and avoid redundant execution, the classifier assignments to L_1 and L_2 are mutually exclusive. In other words, even if both stages draw classifiers from V_{mod}, no classifier appears in both L_1 and L_2. Classifiers in L_2 generally provide higher accuracy at lower throughput, which allows the system to refine uncertain or ambiguous samples forwarded from the first stage. Each classifier in L_2 processes the flagged samples and outputs a refined toxicity score (y_i^{score}). Based on this score, the sample is classified as either toxic or nontoxic. Samples confirmed as nontoxic at any stage exit the pipeline, while those confirmed as toxic are labeled and forwarded for moderation or removal as necessary. In simple terms, the first stage quickly filters out clearly nontoxic content, while the second stage performs a more careful evaluation only when needed. To clarify the cascade behavior, consider the following brief example. Suppose the first-stage lightweight classifier receives a user comment and predicts it as nontoxic with a confidence of 0.97. Since this confidence exceeds the threshold ξ, the system outputs the result immediately, and no additional classifiers are invoked. In contrast, if the confidence were 0.62, which is below ξ, the next model in the cascade would be activated. This ensures that computationally expensive models are used only when necessary, improving throughput while preserving decision accuracy for uncertain cases.

However, selecting which classifiers should be assigned to each stage is not trivial, because the choice must balance accuracy, latency, and cost under varying input conditions. If $|L_1| = m_1$ classifiers and $|L_2| = m_2$ classifiers, the total number of possible cases where at least one classifier is selected

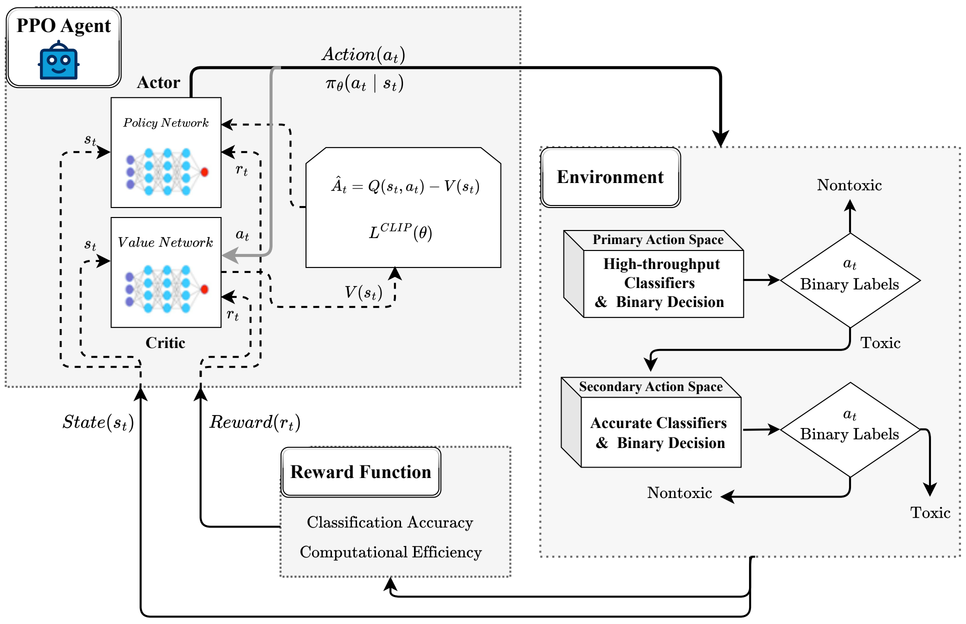


Figure 2: Architecture of the proposed PPO-based cascade inference system for dynamic selection of classifiers

from the union of these two sets $|L_1| + |L_2| = m_1 + m_2$ can be determined using the principle of combinations. The total number of ways to select any subset of classifiers (including the empty set) from these $m_1 + m_2$ classifiers is $2^{m_1+m_2}$. However, since we need at least one classifier to be selected, we must subtract the one case where no classifiers are selected. Thus, the total number of possible cases where at least one classifier is selected from the union of $|L_1| + |L_2| = m_1 + m_2$ is $2^{m_1+m_2} - 1$. This means that manually configuring the cascade for optimal performance is impractical, especially in dynamic, high-volume environments. Moreover, running all classifiers for every sample would waste computational resources. This motivates the use of an RL agent to automatically select the most appropriate classifier at each step, based on the confidence and context of the input.

3.2. DRL-based Cascade Inference Systems

Since each classifier has its own features, in some cases, processing a query with several classifiers can result in a more accurate response. Assume that classifiers in L_1 all have high throughput, which is useful for processing large volumes of data within a second. Among these classifiers, one can have the highest ψ while its α might be less compared to other $k_i \in L_1$. There is another classifier k_j (where $i \neq j$) whose ψ_j is the least throughput in L_1 but still acceptable for analyzing large volumes of data in seconds, and it has the highest α in L_1. There are also other classifiers in L_1 where $\psi < \max(\psi)$ and $\alpha < \max(\alpha)$.

Now, assume that $x \in D$ is processed by $k_1 \in L_1$ and the prediction score k_1^{score} indicates nontoxic content. If another classifier such as k_2 analyzes x, the prediction score might either reinforce the nontoxic label or suggest toxic content. If it is closer to nontoxic, the probability of a nontoxic label for x is higher. If the score is closer to toxic, it is better to call $k_3 \in L_1$ to continue the analysis and get the score with the highest confidence. Finally, those $x \in D$ that are detected as nontoxic with high confidence can be made visible to users and removed from further processing. This reduces the number of samples requiring more processing, allowing us to use classifiers in L_2 with lower throughput but higher accuracy. The same process is repeated in the second stage to get the final decision with a high confidence score. Since manually selecting classifiers is both challenging and resource-intensive, we adopt Proximal Policy Optimization (PPO), a DRL technique, to automate this multi-decision-making process. Further details on PPO are provided in Appendix A.

We present PPO-CIS, depicted in Figure 2, a cascade inference system based on deep reinforcement learning using Proximal Policy Optimization (PPO). PPO-CIS is designed for high-throughput and accurate classification of online toxicity, serving as an early phase in content moderation systems. Instead of deploying a single classifier or all at once, the agent dynamically selects which additional classifiers (if any) to call based on the results of previous ones, and moves to the next stage where more accurate but lower-throughput classifiers analyze the samples using the same rule as in the first stage. Using this approach, the classifiers in the second stage, which have lower throughput, receive fewer samples, allowing them to handle those samples more effectively. In all experiments, latency and throughput are computed end-to-end, where the processing time t for each sample includes both the PPO-CIS decision steps (RL agent inference) and the execution time of all classifiers selected in the cascade. Below, we describe the states, actions, and rewards of our proposed PPO-CIS.

States: The state space S consists of all possible scenarios the algorithm may encounter. For the first stage with $|L_1| = m_1$ classifiers, there are $2^{m_1} - 1$

possible selections of at least one classifier, each associated with a confidence score y_1^{score}. Based on this initial confidence score, the classifiers in the second stage $|L_2| = m_2$ are activated. Consequently, the overall environment includes $|S| = 2^{m_1+m_2} - 1$ states. When at least one classifier in the second stage is activated, a second confidence score y_2^{score} is achieved. It is important to note that the order of classifiers applied in the cascade is not fixed but is dynamically selected to optimize performance.

For a cascade consisting of $H \subseteq K$ classifiers, each possible state $s \in S$ is represented by a vector $\beta = \{\beta_1, \beta_2, \ldots, \beta_H\}$, with the value of β_h set by:

$$\beta_h = \begin{cases} b_h, & \text{if the classifier is employed,} \quad b_h \in [0,1], \\ -1, & \text{if the classifier is not employed,} \end{cases}$$

Here, b_h represents the normalized activation or confidence level of classifier h when it is employed, whereas $\beta_h = -1$ indicates that the classifier is not used in the current cascade stage.

Action Space: In the PPO-CIS, the action space is designed to adapt dynamically based on the stage of classification. Initially, the action space A_1 includes selecting one of the m_1 classifiers in L_1 or making a classification decision (toxic or nontoxic), resulting in $|A_1| = m_1 + 2$ possible actions. If the sample is classified as **toxic** in the first stage, the action space changes to A_2, where the actions include selecting one of the classifiers in L_2 or making a final classification decision. This design ensures that the agent can make more granular decisions initially and refine these decisions in subsequent stages.

Thus, the action spaces are defined as follows:

- **Termination Action Space** (A_T):
  - A_T consists of two actions:
    - **Classify as Toxic**
    - **Classify as Nontoxic**
- **First Stage Action Space** (A_1):

  $$A_1 = |L_1| + 2$$

  The available actions for the agent at the first stage include:

  - Applying any classifier $k_i \in L_1$

  - Classifying a sample as nontoxic
  - Classifying a sample as toxic

  If the agent classifies a sample as nontoxic (taking the action of nontoxic $\in A_T$), the process terminates for that sample.

- **Second Stage Action Space** (A_2):

$$A_2 = |L_2| + 2$$

  The available actions for the agent at the second stage include:

  - Applying any classifier $k_j \in L_2$
  - Classifying a sample as nontoxic
  - Classifying a sample as toxic (which must be reviewed by moderators)

The design of these action spaces enables the agent to dynamically select the most appropriate classifiers and make classification decisions that maximize the confidence in correct labeling while minimizing processing time. By structuring the action spaces in stages, the system ensures that nontoxic samples are efficiently filtered out, and toxic samples receive thorough and detailed analysis.

To clarify the dynamic decision-making process, consider a sample comment x entering the first stage. The agent begins in state s_0, where no classifiers have been applied yet. It chooses an action from A_1. Suppose the agent selects classifier $k_1 \in L_1$, and the resulting score is $y_{k_1}^{\text{score}} = 0.58$, which is below the threshold ξ. This suggests uncertainty, so the agent transitions to a new state s_1, where the state vector now indicates that k_1 has been applied. In state s_1, the agent again selects an action from A_1, depending on the confidence context. If the agent chooses another classifier $k_3 \in L_1$, and this yields $y_{k_3}^{\text{score}} = 0.91$, which exceeds ξ, the agent may choose the termination action "Classify as nontoxic," completing the decision without invoking the more computationally expensive classifiers in L_2. Alternatively, if both first-stage classifiers produce low-confidence scores (e.g., 0.58 and 0.63), the agent transitions to the second-stage action space A_2, where it may select a highly accurate classifier $k_j \in L_2$ to refine the decision. This example demonstrates how the agent adaptively chooses classifiers based on confidence, minimizing unnecessary computation while maintaining accuracy.

Rewards: The reward function in our PPO-CIS is designed to balance the need for accurate classification with the efficiency of processing time and resource utilization. The rewards are structured to reflect the importance of minimizing classification errors and optimizing computational resources. The following factors are considered to define the reward function:

1. Classification Accuracy Reward: Correct classifications (True Positive (TP) and True Negative (TN)) are rewarded, while incorrect classifications (False Positive (FP) and False Negative (FN)) incur penalties. This ensures that the system prioritizes accuracy. Different weights can be assigned to each type of error based on the organization's policy. For instance, FN (where toxic content is misclassified as nontoxic) might incur a higher penalty due to the potential harm of exposing toxic content to users. The classification reward is defined as follows:

$$R_{\text{class}}(x, y) = \begin{cases} +r & \text{if } TP, TN \\ -\iota r & \text{if } FP, FN \end{cases} \tag{1}$$

where:

- r is the reward for a correct classification (TP or TN).
- ι is a penalty factor ($\iota > 1$) to differentiate the severity of misclassifications. For example, ι can be higher for FN to reflect the higher penalty due to the potential harm of exposing toxic content.
- x is the predicted label.
- y is the true label.

This reward structure allows the agent to prioritize accuracy by rewarding correct classifications and penalizing incorrect ones, with adjustable weights to reflect the relative importance of different types of errors.

2. Computational Efficiency Reward: The efficiency of processing user-generated content (UGC) is measured in terms of latency. To incentivize quick processing and penalize delays, we propose a reward function that rewards low latency and penalizes high latency. This ensures that the system can handle large volumes of data efficiently.

The reward function $R_{\text{time}}(t)$ is defined as follows:

$$R_{\text{time}}(t) = \begin{cases} \varphi + \vartheta_1 \cdot \log_2(u - t) & \text{if } 0 \leq t < u \\ \varphi + \vartheta_1 \cdot \log_2(t - u) & \text{if } t \geq u \end{cases} \tag{2}$$

where:

- t is the processing time for a sample.
- u is the upper limit of acceptable processing time.
- φ is a constant offset to ensure a baseline reward, set to $\varphi \geq 1$.
- ϑ_1 is a scaling factor to amplify the differences between rewards and penalties, set to $\vartheta_1 > 1$.

The function is designed to achieve the following:

- **Reward for Low Latency**: When the processing time t is less than the acceptable limit u, the agent receives a reward. The reward increases as t decreases, encouraging the agent to minimize latency.
- **Penalty for High Latency**: When the processing time t exceeds the acceptable limit u, the agent incurs a penalty. The penalty increases as t increases, discouraging the agent from allowing high latency.

The constants φ and ϑ_1 are critical for tuning the reward function. By setting ϑ_1 to a value greater than 1, we significantly amplify the reward for low latency and the penalty for high latency, creating a strong incentive for the agent to optimize processing times effectively.

3. Single Observation Reward: The total reward for each observation combines both classification accuracy and computational efficiency. This ensures the agent is incentivized to achieve both high accuracy and low processing times.

$$R_{\text{input}}(x, y, t) = R_{\text{class}}(x, y) + R_{\text{time}}(t) \tag{3}$$

where:

- $R_{\text{class}}(x, y)$ is the reward for classification accuracy, based on the comparison of the predicted label x with the true label y (see Equation 1).
- $R_{\text{time}}(t)$ is the reward for computational efficiency (see Equation 2).

4. Batch Reward: We also introduce a reward function for a batch of M observations. This allows the system to evaluate accuracy and other metrics for a batch of samples, incorporating throughput as a metric to guide the agent in optimizing processing efficiency for batches of data.

The throughput ψ_M for a batch of M observations is calculated as:

$$\psi_M = \frac{M}{\sum_{i=1}^{M} t_i} \tag{4}$$

where $\sum_{i=1}^{M} t_i$ is the total processing time for the batch.

5. Total Batch Reward: The reward for the batch combines the classification and time rewards for each observation and incorporates the throughput reward:

$$R_M = \sum_{i=1}^{M} (R_{\text{time}}(t_i) + R_{\text{class}}(x_i, y_i)) \tag{5}$$

The total reward for the batch is:

$$R_{\text{tot}} = R_M + \psi_M \tag{6}$$

This reward function ensures that the agent is encouraged to make accurate classifications efficiently, balancing the trade-off between processing speed and computational cost.

To ensure that the agent processes samples within acceptable time ranges, we apply a final adjustment to the batch reward based on the evaluated metric Δ:

$$\hat{R}_{\text{tot}}(\Delta) = \begin{cases} -\vartheta_2 \cdot R_M & \text{if } \Delta \leq l \\ R_M & \text{if } l < \Delta \leq U \\ \vartheta_2 \cdot R_M & \text{if } \Delta > U \end{cases} \tag{7}$$

where:

- U and l are the upper and lower bounds of the acceptable range of the evaluated metric Δ.
- Δ is the value obtained by our approach for a batch of M samples.
- ϑ_2 is a manually defined bonus/penalty factor.

PPO is a powerful RL algorithm that can utilize both single observation rewards and batch rewards to guide the agent's learning process. During each step of the training process, the agent receives an immediate reward for each observation $R_{\text{input}}(x, y, t)$ (Equation 3). This reward informs the agent about the quality of its classification decision and the efficiency of its processing time for each individual sample.

PPO can also be configured to use batch rewards, which provide feedback based on the collective performance over a batch of observations. The total reward for a batch of M observations is R_{tot} (Equation 5), where R_M is the sum of rewards for each observation in the batch, and ψ_M is the throughput reward. Additionally, the final adjustment (($\hat{R}_{\text{tot}}$; see Equation 7) ensures that the agent considers the overall efficiency and accuracy within acceptable ranges.

By incorporating both single and batch rewards, PPO can balance the trade-offs between immediate and long-term performance, encouraging the agent to optimize for both individual and collective metrics. This dual reward structure helps the agent to make decisions that are beneficial in both the short-term (single observations) and long-term (batches), leading to more robust and effective learning.

4. Experimental Setup

4.1. Toxic Datasets

To evaluate our proposed technique, we used two distinct toxic datasets, detailed in Section 4.1.1 and Section 4.1.2.

4.1.1. ***AugmenToxic Dataset***

We utilized a toxic content dataset developed by our team, based on publicly available datasets from Google Jigsaw and Kaggle (Jigsaw, 2017), and described in (Bodaghi et al., 2024). This dataset was generated using the instruct-tuning of FLAN-T5 (Chung et al., 2022) and further optimized through Reinforcement Learning from Human Feedback (RLHF) (Filar and Vrieze, 2012).

The dataset consists of 278,352 samples, equally divided between toxic and nontoxic categories (139,176 samples each). All nontoxic samples were generated and rated by humans. The toxic samples include 16,225 instances generated and rated by humans, while the remaining 122,951 samples were generated by the optimized FLAN-T5 and rated using the Google Perspective API[1].

The dataset is divided into training, validation, and test sets, with proportions of 70%, 10%, and 20%, respectively (see Table 1). The training and validation sets were used for developing and fine-tuning the classifiers, while the test set was reserved for evaluating the performance of classifiers and developing the DRL-based and cascaded models. In this paper, we refer to this dataset as “AugmenToxic” and denote it as $D_{\text{AugmenToxic}}$.

Table 1: Number of Samples per Set for $D_{\text{AugmenToxic}}$

Set	**Number of Samples**
Train	194,846
Validation	27,835
Test	55,670

*4.1.2. **ToxiGen Dataset***

The second dataset, ToxiGen, consists of 274,000 machine-generated statements, covering both toxic and nontoxic content related to 13 different minority groups (Hartvigsen et al., 2022). We selected 8,000 samples, equally divided into 4,000 toxic and 4,000 nontoxic. This dataset was exclusively used to test the performance of the classifiers and for developing the DRL-based and cascaded classification models. Throughout this paper, we will refer to this dataset as “ToxiGen” and denote it as D_{ToxiGen}.

4.2. Classifiers

Our selection of classifiers for cascaded inference was guided by strategic objectives aimed at optimizing performance and adaptability:

Architectural Diversity: We selected classifiers with diverse architectures tailored for various facets of text processing, including CNN-based models and transformer-based models. This diversity ensures comprehensive coverage across different types of textual data and tasks.

Performance Variability: Each classifier was deliberately chosen to offer distinct trade-offs in performance metrics, such as accuracy and com-

[1]https://perspectiveapi.com/

Table 2: All Models Selected for Experimentation

Objective	**Model**	**Accuracy**	**F1-score**	**Throughput (s)**
Accuracy	CNN_α	95.04%	95.10%	494
	$\text{CNN-fastText}_\alpha$	94.17%	94.29%	307
	BERT-base_α	96.97%	97.22%	7.4
	$\text{RoBERTa-base}_\alpha$	96.26%	97.00%	7.8
Throughput	CNN_ψ	93.27%	93.2%	1,512
	CNN-fastText_ψ	91.90%	92.24%	418
	BERT-Tiny_ψ	95.10%	95.19%	169
	$\text{DistilRoBERTa-base}_\psi$	96.64%	96.68%	11

putational efficiency (throughput). This approach allows us to explore different levels of computational demand while maintaining high accuracy levels across different application scenarios.

These objectives underpin our approach to building a robust cascaded inference system capable of effectively handling diverse tasks and adapting to varying computational requirements.

To operationalize these goals, we evaluated four different detectors sourced from recent studies. For each classifier, we developed two variants: one optimized for accuracy and another for high throughput, resulting in a total of eight distinct classifiers.

We trained detectors using the dataset described in Section 4.1.1, with the validation set used for hyperparameter tuning. Four classifier types were evaluated: CNN, CNN with fastText embeddings (Mikolov et al., 2017), BERT, and RoBERTa (Liu et al., 2019). To identify the most accurate and high-throughput models, we tested multiple variants with randomly selected hyperparameters on the validation set. Further details on classifier development are provided in Appendix B. Throughout this paper, we denote the accurate variants of classifiers as α and the high-throughput variants as ψ. Since all datasets used in this paper are balanced with an equal number of toxic and nontoxic samples in all sets (train, validation, and test), we used Accuracy (α) and Throughput (ψ) as the evaluation metrics for comparing the performance of classifiers.

CNN-based: The experimental results for the most accurate CNN and CNN-fastText models are shown in Table 2, the results are based on the validation set of $D^{\text{val}}_{\text{AugmenToxic}}$. Detailed information about the parameters of different variants is provided in Appendix B.

As shown in Table 2, the throughput for the accurate variants of CNN and CNN-fastText is not high, which is acceptable since our focus was on improving accuracy. To develop high-throughput variants, we modified the architecture, resulting in some loss of accuracy (see Appendix B). The high-throughput variants of CNN-based classifiers are also depicted in Table 2.

Transformer-based: We used the Huggingface transformer library, compatible with TensorFlow 2.15.0, for our work. We experimented with different variants of BERT and RoBERTa to select the most accurate and high-throughput variants for each. The experiments resulted in "bert-base-uncased" and "roberta-base" as the most accurate models, while "BERT-Tiny" (Jiao et al., 2020) and "DistilRoBERTa" were identified as the high-throughput variants. We explored a distinct set of hyperparameters randomly to identify the configurations yielding the desired results. The optimal hyperparameters for each model are detailed in Appendix B. The results are shown in Table 2. As shown in this table, $BERT_\alpha$ is the most accurate classifier with an accuracy of 96.97%, whereas $fastText_\psi$ has the lowest accuracy at 91.90%. When considering throughput, $BERT_\alpha$ processes the fewest samples per second, handling only 7.4, while CNN_ψ achieves the highest throughput at 1,512 samples per second. Additionally, CNN_α offers a balance between accuracy and speed, with an accuracy of 95.04% and a throughput of 494 samples per second.

This analysis highlights that no single classifier excels in both accuracy and throughput, making it challenging to select an ideal solution for toxicity detection. Although CNN_α performs moderately well in both aspects, its throughput may be insufficient during peak times when platforms receive higher volumes of UGC. Moreover, in scenarios with a high ratio of toxic content, CNN_α may struggle to process all UGC efficiently, indicating the need for a more robust solution that can handle diverse and high-volume content effectively.

To demonstrate the relative performance and ensure no single classifier dominates, we evaluated all selected classifiers using separate test sets from $D_{\text{AugmenToxic}}$ (55,653 samples) and D_{ToxiGen} (8,000 samples). Notably, these data points were not used during classifier training or validation. As of now, we refer to these datasets as $D^{\text{test}}_{\text{AugmenToxic}}$ and $D^{\text{test}}_{\text{ToxiGen}}$ to distinguish them from the training and validation sets. Table 3 and Table 4 compare the classifiers' performance on the $D^{\text{test}}_{\text{AugmenToxic}}$ and the $D^{\text{test}}_{\text{ToxiGen}}$ dataset, respectively. Each table illustrates the percentage of samples that one classifier

Table 3: Classifier performance on misclassified samples for $D^{\text{test}}_{\text{AugmenToxic}}$

	CNN$_\alpha$	**fastText**$_\alpha$	**BERT**$_\alpha$	**RoBERTa**$_\alpha$	**CNN**$_\psi$	**fastText**$_\psi$	**BERT**$_\psi$	**RoBERTa**$_\psi$
CNN$_\alpha$	-	14%	93%	91%	12%	16%	92%	91%
fastText$_\alpha$	16%	-	93%	90%	13%	9%	92%	90%
BERT$_\alpha$	50%	47%	-	32%	48%	42%	32%	29%
RoBERTa$_\alpha$	52%	48%	49%	-	51%	42%	54%	31%
CNN$_\psi$	11%	10%	92%	91%	-	13%	92%	90%
fastText$_\psi$	21%	12%	92%	89%	18%	-	92%	90%
BERT$_\psi$	50%	48%	37%	43%	47%	47%	-	40%
RoBERTa$_\psi$	48%	43%	43%	26%	46%	40%	49%	-

Table 4: Classifier performance on misclassified samples for $D^{\text{test}}_{\text{ToxiGen}}$

	CNN$_\alpha$	**fastText**$_\alpha$	**BERT**$_\alpha$	**RoBERTa-base**$_\alpha$	**CNN**$_\psi$	**fastText**$_\psi$	**BERT**$_\psi$	**lRoBERTa**$_\psi$
CNN$_\alpha$	-	11%	74%	71%	14%	11%	76%	77%
fastText$_\alpha$	35%	-	77%	75%	28%	5%	79%	80%
BERT$_\alpha$	55%	46%	-	19%	55%	43%	32%	30%
RoBERTa$_\alpha$	55%	47%	30%	-	55%	44%	38%	31%
CNN$_\psi$	18%	6%	76%	72%	-	5%	77%	78%
fastText$_\psi$	40%	13%	78%	76%	33%	-	80%	81%
BERT$_\psi$	59%	50%	34%	30%	59%	47%	-	39%
RoBERTa$_\psi$	55%	46%	21%	10%	55%	43%	30%	-

misclassified (rows) but another correctly classified (columns).

For example, the $BERT_\alpha$ detector accurately identified 93% of comments misclassified by CNN_α. In the $D^{\text{test}}_{\text{AugmenToxic}}$ set, 35.20% of samples were correctly classified by all eight classifiers, with 17.08% being toxic and 18.12% nontoxic. Similarly, in $D^{\text{test}}_{\text{ToxiGen}}$, 50.22% of samples were correctly classified overall, comprising 35.64% toxic and 14.58% nontoxic samples. As of now, we refer to these datasets as $D^{\text{test}}_{\text{AugmenToxic}}$ and $D^{\text{test}}_{\text{ToxiGen}}$ to distinguish them from the training and validation sets. In addition, for $D^{\text{test}}_{\text{ToxiGen}}$, only 50.22% of the samples were correctly classified by all classifiers. Specifically, 35.64% of the toxic samples were correctly identified as toxic, while only 14.58% of the nontoxic samples were correctly identified as nontoxic.

For $D^{\text{test}}_{\text{AugmenToxic}}$, only 45.20% of the samples were correctly classified by all classifiers. Among these, 27.08% of the toxic samples were correctly identified as toxic, while 18.12% of the nontoxic samples were correctly identified as nontoxic. These findings underscore that no single classifier dominates in accuracy across both datasets, affirming the robustness and comparative performance of each classifier.

4.3. Baselines

To evaluate the performance of PPO-CIS, we compare it with several baselines:

Dynamic vs. Static Classifier Selection: PPO-CIS dynamically adjusts classifier selection based on real-time confidence scores to optimize detection accuracy and efficiency. In contrast, static baselines employ predetermined combinations of classifiers within the cascade, with decisions finalized using majority voting, soft voting, or the output of the final classifier in the sequence. This comparison highlights DRL-CIS's adaptability and its performance advantages over fixed configurations in real-world applications.

Comparison with CETRA: We benchmark PPO-CIS against CETRA (Lavie et al., 2023) to evaluate the effectiveness of its reward function, Proximal Policy Optimization (PPO) as the DRL model, and cascaded classifier arrangements with dual variants emphasizing throughput and accuracy. This comparison underscores the superiority of PPO-CIS's cascaded design in toxicity detection scenarios, showcasing its enhanced adaptability and effectiveness.

In real-time scenarios, we assess PPO-CIS and the baselines across critical metrics including throughput, processing time, and detection accuracy. This evaluation provides insights into the operational efficiency of PPO-CIS for dynamic toxicity detection, highlighting its capability for rapid decision-making and robust performance.

4.4. Experimental setting

As detailed in Section 4.2, we first trained and validated classifiers using the $D^{\text{train}}_{\text{AugmenToxic}}$ and $D^{\text{val}}_{\text{AugmenToxic}}$ datasets (see Section 4.1.1). Subsequently, the $D^{\text{test}}_{\text{AugmenToxic}}$ and the entire $D^{\text{test}}_{\text{ToxiGen}}$ datasets were utilized to evaluate these classifiers. Each data point was assessed for binary and probabilistic labels, alongside processing times, providing comprehensive metrics such as accuracy, precision, recall, F1-score, throughput per second, and average processing time. These evaluations formed the basis for developing our deep reinforcement learning (DRL) models, using the $D^{\text{test}}_{\text{AugmenToxic}}$ and $D^{\text{test}}_{\text{ToxiGen}}$ datasets for testing and refinement.

For developing DRL-based models, we split $D^{\text{test}}_{\text{AugmenToxic}}$ and $D^{\text{test}}_{\text{ToxiGen}}$ into training, test, and validation sets. To compare the performance of all potential cascade inference systems (CIS) with single classifiers, we held out two balanced datasets, $\hat{D}^{\text{test}}_{\text{AugmenToxic}}$ and $\hat{D}^{\text{test}}_{\text{ToxiGen}}$, containing 5,000 and 2,000

Table 5: PPO-Specific Hyperparameters

Hyperparameter	**Value**
Clip Range (ϵ)	0.2
Learning Rate (lr)	0.0003
Batch Size	256
Number of Epochs	10
Discount Factor (γ)	0.99
GAE Lambda (λ)	0.95
Entropy Coefficient	0.01
Value Function Coefficient	0.5

samples, respectively. All results reported from this point onwards are based on these datasets.

Our framework was implemented in Python v3.8 using OpenAI Gym (Brockman et al., 2016). For our DRL agents, we employed Proximal Policy Optimization (PPO) (Section 10.2) for PPO-CIS and Actor-Critic with Experience Replay (ACER) (Section 10.1) for CETRA, utilizing the "ChainerRL" library (Nandy and Biswas, 2018). CETRA, originally designed for dynamic classifier selection in ensemble learning for malware detection, provided a comparative baseline with five distinct reward functions evaluated separately.

In our implementation of PPO-CIS, we used an "FCSoftmaxPolicyAdam" architecture for the policy network, featuring fully connected layers followed by softmax activation to facilitate decision-making. The "FCVFunction" served as the value function, crucial for estimating expected returns during policy updates. The Adam optimizer was configured with a conservative learning rate of 1×10^{-5} and gradient clipping with a threshold of 0.5, ensuring stable training dynamics by limiting excessive gradient updates.

The neural network architecture included an input layer with 8 neurons to handle the state representation required for decision-making, a hidden layer with 64 neurons, and an output layer with 12 neurons supporting 12 possible actions across two distinct action spaces. These actions encompassed selecting among individual classifiers and making final toxicity classifications both critical for effective decision-making in cascade inference systems.

To ensure reproducibility and clarity of our proposed method, we specify the hyperparameters used in our experiments in Table 5 and Table 6.

Table 6: Reward Function Hyperparameters

Hyperparameter	**Value**
Classification Reward (r)	10
Classification Penalty Factor (ι)	2
Upper Limit of Acceptable Processing Time (u)	1
Constant Offset (φ)	1
Scaling Factor (ϑ_1)	10
Bounds for Final Adjustment (U and l)	$U = 1.5$, $l = 0.5$
Bonus/Penalty Factor (ϑ_2)	1.2

Our experimental approach prioritized stability and efficiency, with carefully chosen architecture and optimizer configurations tailored to the demands of real-time inference tasks. By detailing these parameters, we aimed to enhance transparency and reproducibility while ensuring robust performance in complex decision environments.

The reward function in CETRA is defined as follows:

$$C_{2_{\text{Total}}}(T) = \begin{cases} r & \text{for TP or TN} \\ -1 \times \sum_{t=1}^{T} C_2(t) & \text{for FN or FP} \end{cases} \tag{8}$$

where r is a constant, and $C_2(t)$ is the reward function for each step, defined as:

$$C_2(t) = \begin{cases} \frac{t}{d_2} & \text{for } 0 \leq t < d_2 \\ 1 + \log_2 \left(\frac{\min(t, t_2(s))}{d_2} \right) & \text{for } d_2 \leq t \end{cases} \tag{9}$$

The reward values were assigned differently in five separate experiments. In Experiment 1, TP and TN both received a reward value of $C_{2,j}$, whereas FP and FN both received a penalty of $-C_{2,j}$. Experiment 2 maintained the same rewards for TP and TN at $C_{2,j}$, but increased the penalties for FP and FN to $-10C_{2,j}$. In Experiment 3, a uniform reward of 1 was assigned to both TP and TN, with FP and FN penalties remaining at $-C_{2,j}$. Experiment 4 increased the rewards for TP and TN to 10, while keeping the FP and FN penalties at $-C_{2,j}$. Finally, Experiment 5 further increased the rewards for TP and TN to 100, with FP and FN penalties still at $-C_{2,j}$. In our experi-

Table 7: Performance Metrics of Classifiers on $\hat{D}^{\text{test}}_{\text{AugmenToxic}}$ and $\hat{D}^{\text{test}}_{\text{ToxiGen}}$

Dataset	**Model**	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1-score (%)**	**Throughput (s)**	**Latency (s)**
$\hat{D}^{\text{test}}_{\text{AugmenToxic}}$	CNN_α	93.18%	94.17%	95.42%	94.79%	148	0.0067
	$fastText_\alpha$	92.23%	90.19%	94.78%	92.43%	64.43	0.01552
	$BERT_\alpha$	**93.22%**	92.38%	94.20%	93.28%	**4.595**	0.2176
	$RoBERTa_\alpha$	92.67%	90.85%	93.43%	92.12%	4.73	0.2113
	CNN_ψ	**90.33%**	90.27%	94.30%	92.09%	**269.58**	0.00371
	$fastText_\psi$	91.07%	92.25%	93.84%	93.08%	121.48	0.00823
	$BERT_\psi$	91.00%	87.27%	96.00%	91.43%	7.843	0.1275
	$RoBERTa_\psi$	91.56%	89.48%	94.20%	91.78%	5.41	0.1848
$\hat{D}^{\text{test}}_{\text{ToxiGen}}$	CNN_α	93.54%	89.46%	91.25%	90.35%	28.01	0.0357
	$fastText_\alpha$	91.52%	90.42%	93.25%	91.82%	19.93	0.050175
	$BERT_\alpha$	**93.91%**	90.52%	95.45%	92.92%	**4.10**	0.2442
	$RoBERTa_\alpha$	92.85%	89.79%	94.98%	92.31%	4.23	0.2366
	CNN_ψ	**90.09%**	88.34%	92.78%	90.83%	**52.56**	0.01903
	$fastText_\psi$	90.19%	89.36%	92.98%	91.07%	34.81	0.0287
	$BERT_\psi$	91.82%	88.76%	93.75%	91.18%	5.75	0.174
	$RoBERTa_\psi$	92.76%	93.14%	92.30%	92.73%	4.56	0.21925

ment, d_2 was set to 0.5 because both datasets include a balanced number of toxic and nontoxic samples.

4.5. Computational Resources

For our experiments, we used the "Paperspace P6000" cloud computing instance, which is equipped with NVIDIA P6000 GPUs. Each P6000 GPU offers 24 GB of memory, 30 GB of RAM, and 8 vCPUs, providing robust computational power for our tasks. This setup supports multi-GPU configurations, including 2x and 4x instances, allowing for scalability and parallel processing. The cost of using the P6000 instance is $1.10 per hour.

This computing environment was consistently used for both developing and training the classifiers, as well as for implementing and evaluating the DRL-based approaches. The consistent usage of this environment ensures that our experiments have a reliable and stable computational foundation, allowing for accurate performance comparison and benchmarking.

5. Experimental Results

In this section, we present the results from testing all baseline models. We start by evaluating the performance of individual classifiers on the

$\hat{D}^{\text{test}}_{\text{AugmenToxic}}$ and $\hat{D}^{\text{test}}_{\text{ToxiGen}}$ datasets, as shown in Table 7. Subsequently, Section 5.1 details the performance of various classifier combinations arranged in cascades. This analysis highlights the effectiveness of different cascaded configurations and identifies the optimal setups for toxicity detection. The best-performing cascades are then compared with the DRL-based approaches. Finally, Section 5.2 discusses the experimental results for CETRA with different reward functions and the proposed PPO-CIS.

According to Table 7, for both datasets, $BERT_\alpha$ is the most accurate classifier but has the lowest throughput. Conversely, CNN_ψ demonstrated the highest throughput as expected, but with lower accuracy.

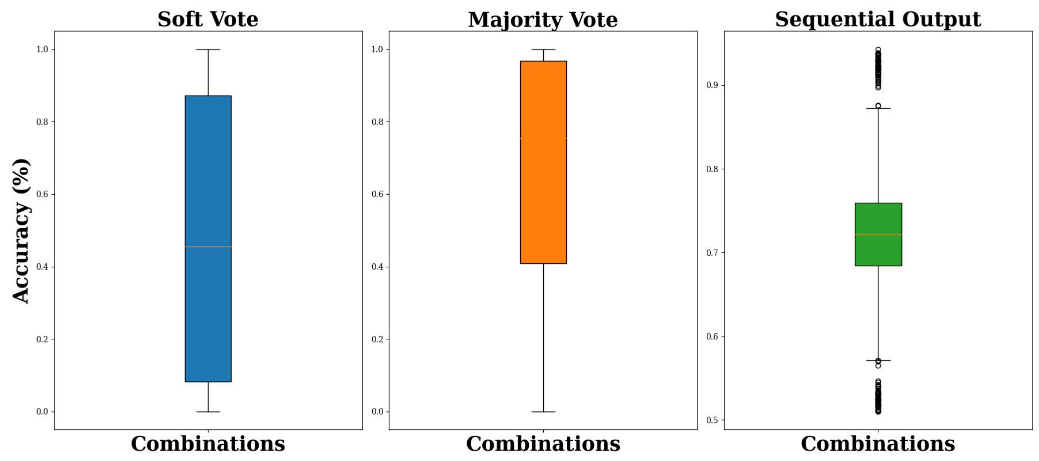
5.1. Cascade of Classifiers

Since we have 8 classifiers, where 4 (CNN-based) have higher throughput and the other 4 (transformer-based) have lower throughput but higher accuracy, the low-throughput classifiers are not capable of processing the large amount of data received per second. However, we tested all possible combinations of classifiers and recorded the total $(\alpha_{P_i}, \rho_{P_i}, \sigma_{P_i}, \phi_{P_i}, \psi_{P_i}, \lambda_{P_i})$ for each set, as explained in Section 3.1. Note that the computational cost ω_{P_i} is considered zero in our experiments. The results for $\hat{D}^{\text{test}}_{\text{AugmenToxic}}$ are shown in Table 8, and for $\hat{D}^{\text{test}}_{\text{ToxiGen}}$ in Table 9.

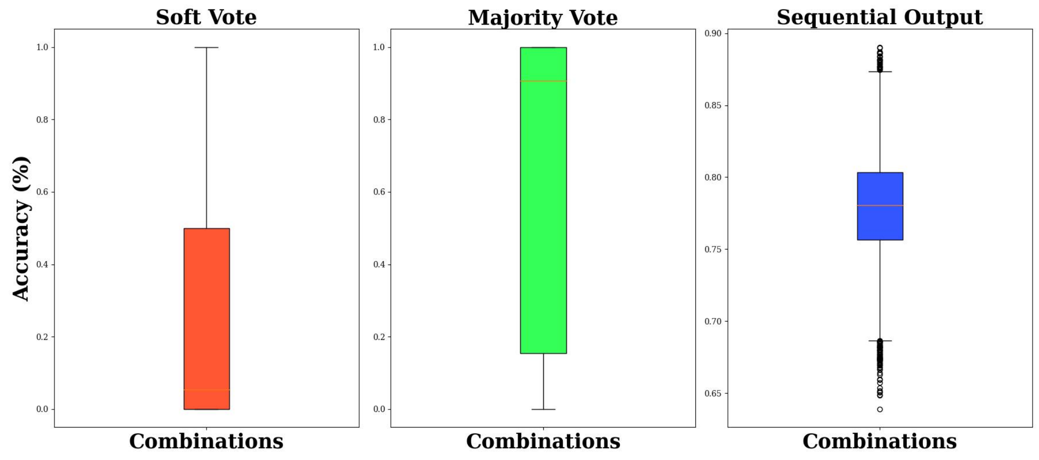
The combination of 8 classifiers, using different voting methods (majority and soft), or the last response (sequential output), results in a large number of possible cases when considering at least 2 classifiers for each dataset. Therefore, we only report the competitive combinations that result in higher accuracy and throughput compared to a single classifier.

Figure 3 illustrates the accuracy achieved by combining at least two classifiers in a cascaded manner using various label determination techniques on different datasets. For the $\hat{D}^{\text{test}}_{\text{AugmenToxic}}$ dataset, the soft vote method shows a wide accuracy range (0.0 to 1.0) with a median of 0.5, indicating high variability. The majority vote method has a similar wide range but a higher median of 0.6, also showing significant variability. The sequential output method, however, has a more concentrated accuracy range (0.5 to 0.9) with a median of 0.72, demonstrating higher consistency and reliability.

For the $\hat{D}^{\text{test}}_{\text{ToxiGen}}$ dataset, the soft vote method also shows a wide accuracy range (0.0 to 1.0) with a median of 0.35, indicating high variability and less consistent performance. The majority vote method again displays a broad range of accuracies (0.0 to 1.0) with a higher median accuracy of 0.65, but still shows significant variability. In contrast, the sequential output method

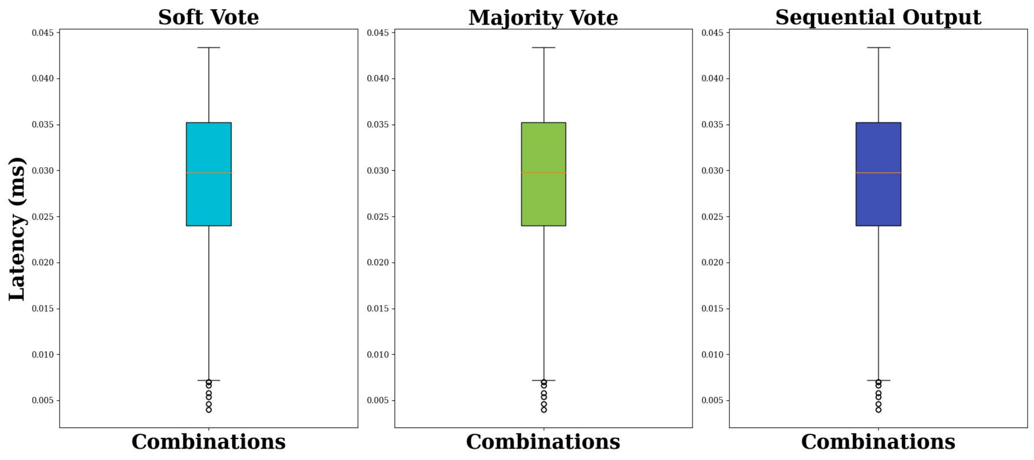


(a) $\hat{D}^{\text{test}}_{\text{AugmenToxic}}$

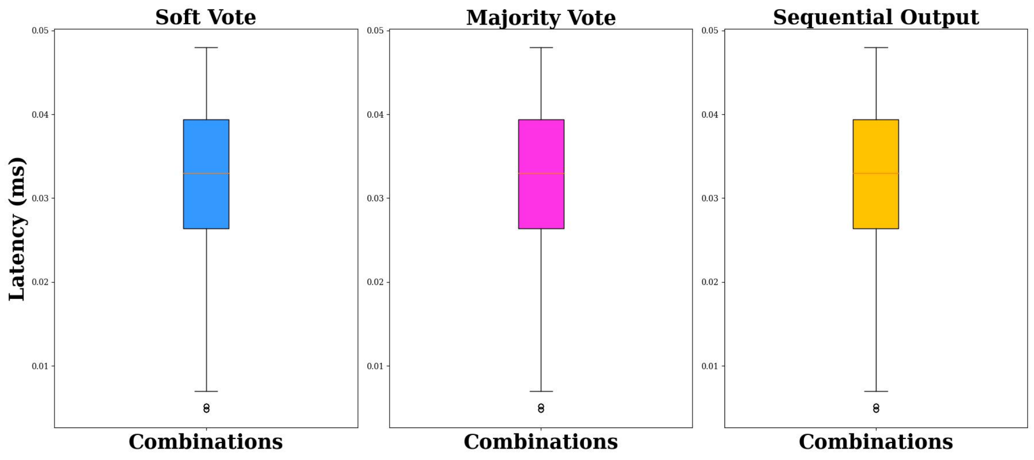


(b) $\hat{D}^{\text{test}}_{\text{ToxiGen}}$

Figure 3: Comparison of accuracy distributions for different combinations of classifiers using different label determination techniques on the (a) $\hat{D}^{\text{test}}_{\text{AugmenToxic}}$ and (b) $\hat{D}^{\text{test}}_{\text{ToxiGen}}$ datasets.



(a) $\hat{D}^{\text{test}}_{\text{AugmenToxic}}$



(b) $\hat{D}^{\text{test}}_{\text{ToxiGen}}$

Figure 4: Comparison of latency distributions for different combinations of classifiers using different label determination techniques on the (a) $\hat{D}^{\text{test}}_{\text{AugmenToxic}}$ and (b) $\hat{D}^{\text{test}}_{\text{ToxiGen}}$ datasets.

demonstrates a more concentrated distribution of accuracies, ranging from 0.65 to 0.98, with a median accuracy near 0.75, indicating less variability and higher consistency. Overall, the sequential output technique outperforms both the soft vote and majority vote methods in terms of median accuracy and consistency for both datasets.

The box plots in Figure 4 illustrate the latency distributions for different combinations of classifiers using soft vote, majority vote, and sequential output label determination techniques on the $\hat{D}^{\text{test}}_{\text{AugmenToxic}}$ and $\hat{D}^{\text{test}}_{\text{ToxiGen}}$ datasets.

For the AugmenToxic dataset (Figure 4a), the soft vote method shows latencies ranging from approximately 0.015 ms to 0.045 ms, with a median latency around 0.03 ms, indicating moderate variability in latency across different classifier combinations. The majority vote method exhibits a similar latency range, from approximately 0.015 ms to 0.045 ms, with a median latency slightly above 0.03 ms, suggesting comparable variability. The sequential output method also demonstrates latencies in the same range, with a median latency around 0.03 ms, showing consistent performance across classifier combinations.

For the ToxiGen dataset (Figure 4b), the soft vote method shows a latency range from approximately 0.01 ms to 0.05 ms, with a median latency around 0.03 ms, indicating some variability in latency. The majority vote method has a latency range similar to the soft vote method, but with a slightly lower median latency around 0.03 ms, demonstrating similar variability. The sequential output method shows a latency range from approximately 0.01 ms to 0.05 ms, with a median latency slightly above 0.03 ms, indicating consistent performance with less variability compared to the other methods.

Overall, all three label determination techniques exhibit similar latency distributions for both the AugmenToxic and ToxiGen datasets. The median latencies are consistent around 0.03 ms, with some variability across different classifier combinations. The choice of label determination technique does not significantly impact the latency, indicating comparable performance in terms of latency across the different methods.

Based on the results presented in Table 8 and Table 9, several key insights can be drawn about the performance of different classifier combinations in terms of throughput, accuracy, and F1-score.

In Table 8, the highest throughput of 723 samples per second is achieved by the No. 1 combination. It is significantly higher than the throughput of single classifiers (CNN_ψ and $fastText_\psi$), and it is notably higher than that of transformer-based classifiers applied alone. The combinations No. 5 and No.

Table 8: Best-Performing Classifier Combinations on $\hat{D}^{\text{test}}_{\text{AugmenToxic}}$

No.	**Combination**	**Label Determination**	**Accuracy$_\alpha$ (%)**	**Precision$_\rho$ (%)**	**Recall$_\sigma$ (%)**	**F1-score$_\phi$ (%)**	**Throughput$_\psi$ (s)**	**Latency$_\lambda$ (s)**
1	$fastText_\psi(0), CNN_\psi \vert\vert fastText_\psi(1), CNN_\psi$	Sequential	**90.32%**	89.13%	96.08%	92.47%	**723**	0.0013
2	$fastText_\alpha(0), BERT_\psi \vert\vert fastText_\alpha(1), BERT_\psi$	Sequential	93.16%	92.95%	93.40%	93.17%	192.22	0.0052
3	$BERT_\alpha(1), BERT_\psi$	Sequential	92.39%	92.39%	92.46%	92.42%	86.21	0.0116
4	$CNN_\alpha(1), fastText_\alpha(1), BERT_\alpha(1), RoBERTa_\alpha(1), CNN_\psi \vert\vert CNN_\alpha(0), fastText_\alpha(0), BERT_\alpha(0), RoBERTa_\alpha(0), fastText_\psi$	Sequential	**95.03%**	95.12%	98.22%	**96.60%**	**340**	0.0224
5	$CNN_\alpha(1), fastText_\alpha(0), BERT_\psi$	Majority	92.01%	93.59%	97.79%	95.64%	311.64	0.0032
6	$CNN_\alpha(0), BERT_\alpha(0), fastText_\psi(1), BERT_\psi(0), RoBERTa_\psi(1)$	Majority	91.35%	91.28%	95.42%	93.31%	51.02	0.0196
7	$CNN_\alpha(0), CNN_\psi(1), fastText_\psi(1), BERT_\psi(1), RoBERTa_\psi \vert\vert CNN_\alpha(1), CNN_\psi(0), fastText_\psi(0), BERT_\psi(0), RoBERTa_\psi$	Majority	94.94	93.79	96.50%	95.13%	82	0.012
8	$BERT_\alpha(0), RoBERTa_\alpha(1), fastText_\psi(1), BERT_\psi \vert\vert BERT_\alpha(1), RoBERTa_\alpha(0), fastText_\psi(0), BERT_\psi$	Majority	**96.14%**	96.09%	96.38%	**96.19%**	**34.97**	0.0286
9	$CNN_\psi(1), fastText_\psi(1), BERT_\psi(1), RoBERTa_\psi(1) \vert\vert CNN_\alpha(0), fastText_\alpha(0), BERT_\alpha(0), RoBERTa_\alpha(0), CNN_\psi$	Majority	**96.15%**	96.10%	97.25	**96.64%**	**42.74**	0.0234
10	$BERT_\alpha(1), CNN_\psi(1), fastText_\psi \vert\vert BERT_\alpha(0), CNN_\psi(0), fastText_\psi$	Soft	92.63%	92.63%	91.30%	96.17%	53.19	0.0188
11	$fastText_\alpha(0), BERT_\alpha(1), CNN_\psi(1), fastText_\psi(0), BERT_\psi \vert\vert fastText_\alpha(1), BERT_\alpha(0), CNN_\psi(1), fastText_\psi(1), BERT_\psi$	Soft	93.10%	92.38%	94.20%	93.28%	40.65	0.0246
12	$CNN_\alpha(0), fastText_\alpha(0), BERT_\alpha(1), RoBERTa_\alpha(1), CNN_\psi(0), fastText_\psi \vert\vert CNN_\alpha(0), fastText_\alpha(1), BERT_\alpha(0), RoBERTa_\alpha(0), CNN_\psi(0), fastText_\psi(0), RoBERTa_\psi(0), BERT_\psi(0)$	Soft	**84.08%**	92.67%	83.20%	87.68%	**28.09**	0.0356
13	$CNN_\alpha(1), fastText_\alpha(1), BERT_\alpha(1), RoBERTa_\alpha(1), CNN_\psi(1), fastText_\psi(1), RoBERTa_\psi \vert\vert CNN_\alpha(1), fastText_\alpha(0), BERT_\alpha(1), RoBERTa_\alpha(1), CNN_\psi(1), fastText_\psi(0), RoBERTa_\psi(1)$	Soft	91.74%	85.32%	92.89%	88.94%	43.04	0.0234

2 follow with throughputs of 311 and 192 samples per second, respectively. These combinations are among the best in terms of throughput, but have comparatively lower accuracy.

The highest accuracy and F1-score are achieved by combinations using majority vote for label determination, specifically No. 8 and No. 9. Despite their high accuracy, these combinations handle fewer than 50 samples per second, suggesting that single classifiers might be preferable in scenarios that require higher throughput. The combination No. 4 stands out with an accuracy of 95.03% and an F1-score of 96.60%, coupled with a more reasonable throughput of 340 samples per second, making it a balanced choice compared to No. 8 and No. 9.

Transformer-based models ($BERT_\alpha$, $BERT_\psi$, $RoBERTa_\alpha$, $RoBERTa_\psi$) in the cascade significantly reduce throughput, especially at primary stages. Applying CNN-based classifiers at primary stages boosts throughput. For instance, in combination No. 5, all samples are initially processed by CNN_α.

Table 9: Best-Performing Classifier Combinations on $\hat{D}^{\text{test}}_{\text{ToxiGen}}$ dataset

No.	**Combination**	**Label Determination**	**Accuracy$_\alpha$ (%)**	**Precision$_\rho$ (%)**	**Recall$_\sigma$ (%)**	**F1-score$_\phi$ (%)**	**Throughput$_\psi$ (s)**	**Latency$_\lambda$ (s)**
1	$CNN_\alpha(1), BERT_\psi$	Sequential	92.74%	92.18%	96.96%	94.51%	138.88	0.0072
2	$fasText_\alpha(1), BERT_\alpha(0), RoBERTa_\alpha(0), RoBERTa_\psi$	Sequential	90.48%	87.27%	94.30%	90.65%	64.93	0.0154
3	$CNN_\alpha(1), BERT_\alpha(0), RoBERTa_\alpha(0),$ $BERT_\psi(0), RoBERTa_\psi$	Sequential	90.90%	89.48%	92.46%	90.94%	29.59	0.0338
4	$CNN_\alpha(1), fastText_\alpha(1), BERT_\alpha(1), RoBERTa_\alpha(1),$ $CNN_\psi\|\|$ $CNN_\alpha(0), fastText_\alpha(0), RoBERTa_\alpha(0), CNN_\psi$	Sequential	91.43%	92.14%	95.91%	93.99%	36.50	0.0274
5	$fastText_\psi(0), CNN_\psi(1), CNN_\alpha(1),$ $fastText_\psi(1), CNN_\alpha(0), CNN_\psi(0)$	Sequential	**92.74%**	93.85%	96.07%	94.95%	**556**	0.00179
6	$CNN_\alpha(0), fastText_\alpha(0), fastText_\psi(1)$	Sequential	93.50%	93.14%	94.09%	93.61%	**301.8**	0.003315
7	$CNN_\alpha(1), CNN_\psi(1), RoBERTa_\psi(1)$	Majority	93.73%	92.80%	95.05%	93.91%	217	0.0046
8	$CNN_\alpha(1), fastText_\alpha(0), BERT_\alpha(0), RoBERTa_\alpha$	Majority	93.87%	92.39%	95.93%	94.12%	208	0.0048
9	$BERT_\alpha(0), RoBERTa_\alpha(0), CNN_\psi(1), RoBERTa_\psi(1)$	Majority	**96.06%**	93.12%	95.85%	94.43%	38.17	0.0262
10	$CNN_\alpha(0), fastText_\alpha(1), fastText_\psi(1), BERT_\psi(1),$ $RoBERTa_\psi(0)\|\|CNN_\alpha(1), fastText_\alpha(0),$ $fastText_\psi(0), BERT_\psi(0), RoBERTa_\psi$	Majority	**96.84%**	94.98%	97.45%	96.20%	30.49	0.0328
11	$BERT_\alpha(0), RoBERTa_\alpha(1), CNN_\psi(1), fastText_\psi(1),$ $RoBERTa_\psi\|\|BERT_\alpha(1), RoBERTa_\alpha(0), CNN_\psi(0),$ $fastText_\psi(0), RoBERTa_\psi$	Majority	**97.10%**	96.43%	97.66%	**97.07%**	28.86	0.0324
12	$CNN_\alpha(1), fastText_\alpha(0), RoBERTa_\alpha(1)$	Soft	90.94%	90.92%	95.25%	92.98%	149.5	0.0066
13	$CNN_\alpha(0), BERT_\alpha(1), RoBERTa_\alpha(1), BERT_\psi\|\|$ $CNN_\alpha(1), BERT_\alpha(0), RoBERTa_\alpha(0), BERT_\psi$	Soft	88.70%	86.73%	95.20%	90.77%	43.86	0.0228
14	$CNN_\alpha(0), RoBERTa_\alpha(1), fastText_\psi(0),$ $RoBERTa_\psi\|\|$ $fastText_\alpha(0), BERT_\alpha(1), CNN_\psi(0), BERT_\psi$	Soft	89.72%	87.27%	92.59%	89.85%	43.10	0.0232
15	$BERT_\alpha(1), RoBERTa_\alpha(1), CNN_\psi(1), RoBERTa_\psi\|\|$ $BERT_\alpha(0), RoBERTa_\alpha(0), fastText_\psi(0), BERT_\psi$	Soft	93.68%	93.56%	97.02%	95.26%	37.02	0.027

Toxic samples are then processed by $fastText_\alpha$, and only those classified as nontoxic undergo further analysis by $BERT_\psi$. This staged filtering improves throughput by reducing the number of samples needing in-depth processing by $BERT_\psi$. One of the most surprising results was achieved by No. 3, where $BERT_\alpha$ was applied first, followed by $BERT_\psi$. The throughput significantly improved to 86, even though the throughput for both individual models was less than 10.

soft vote methods do not significantly enhance accuracy or throughput. Furthermore, long cascades, such as those in combinations No. 11, No. 12, and No. 13 fail to improve accuracy and instead reduce throughput, indicating that more classifiers do not necessarily lead to better performance.

In Table 9, combination No. 5 achieves the highest throughput (ψ) of 556 samples per second by avoiding transformer-based classifiers. The lowest throughput (ψ) of 28.86 samples per second is observed in combination No. 11, which uses highly accurate but slower classifiers, such as $BERT_\alpha$, at the initial stage. Notably, this combination also achieves the highest accuracy with majority vote.

Combinations using majority voting, such as No. 10 and No. 11, yield higher accuracy and F1-scores. For instance, No. 10 achieves an accuracy of 96.84% and an F1-score of 96.20%, with a throughput of 30.49 samples per second. No. 11 achieves the highest accuracy of 97. 10% and an F1 score of 97. 07%, although it has a lower throughput of 28.86 samples per second.

On the other hand, soft voting combinations, such as No. 12 and No. 13, tend to exhibit lower accuracy and F1-scores, indicating that soft voting may reduce performance compared to majority voting. However, No. 15 achieves the highest accuracy and lower throughput among soft vote models.

Overall, balancing accuracy and throughput is crucial. Using fast classifiers like CNNs in the initial stages can improve throughput without significantly compromising accuracy. For example, in combination No. 5 in Table 9, all samples are processed by CNN_{α}, then toxic samples by $fastText_{\alpha}$, and only nontoxic samples undergo further analysis by $BERT_{\psi}$. This staged filtering reduces the number of samples needing in-depth processing, thereby improving throughput.

5.2. DRL-based Inference Systems

After developing top-performing classifiers in a cascaded manner, we demonstrated that this approach could reduce processing time for low-throughput classifiers. In addition, high-throughput combinations still exhibited lower accuracy compared to those with higher accuracy but very low throughput.

A key insight from our results is that the order of classifiers is crucial. By applying fast classifiers at the initial stages and slow classifiers in the final stages, we can achieve higher throughput. Nevertheless, caution is required because adding many classifiers does not necessarily lead to better performance and may even negatively impact it. This underscores the need for dynamic classifier selection.

Moreover, the results presented in Section 5.1 indicated that good results could be achieved by using cascades of classifiers. The key to success lies in dynamically selecting classifiers to enhance both accuracy and throughput. This objective can be accomplished by designing a reward function that considers both accuracy and throughput.

Therefore, the next step is to explore dynamic classifier selection within the cascade. This approach will guide the selection process towards our goal of simultaneously increasing accuracy and throughput.

To evaluate PPO-CIS for toxicity detection, we compared it with CETRA across three key aspects: (a) the reward function used, (b) the dynamic selection of classifiers in a cascaded manner, where high-throughput classifiers are prioritized in the initial stages and more accurate classifiers are employed in subsequent stages, and (c) the DRL model applied, where CETRA employs Actor-Critic with Experience Replay (ACER) (Wang et al., 2017) (see Appendix A for details), and our framework is implemented using PPO.

Note that all five reward functions were tested for both DRL approaches (ACER and PPO). The final results for each dataset are presented in Table 10 and Table 11. All results are reported based on the test set for each dataset, which includes 5,000 samples for $\hat{D}^{\text{test}}_{\text{AugmenToxic}}$ and 2,000 samples for $\hat{D}^{\text{test}}_{\text{ToxiGen}}$.

Please note that we use $CETRA - CIS_i$ for $i = 1$ to 5 to denote models that use the CETRA structure, employing ACER as the DRL model, with all 8 classifiers at the same stage. The index $i = 1$ to 5 refers to the specific reward functions proposed by (Lavie et al., 2023). Additionally, we use $ACER - CIS$ to refer to our proposed CIS, which includes two action spaces, where ACER is applied as the DRL model instead of PPO, along with our proposed reward function. We also use $ACER - CIS_i$ for $i = 1$ to 5 to denote models tested with the five different reward functions. Similarly, $PPO - CIS$ refers to our proposed CIS with PPO as the DRL model, incorporating its own reward function and two space actions. Furthermore, $PPO - CIS_i$ for $i = 1$ to 5 denotes the proposed PPO-CIS models tested with the reward functions proposed by (Lavie et al., 2023).

We compared the performance of various inference systems for high-throughput toxicity detection on the $\hat{D}^{\text{test}}_{\text{AugmenToxic}}$ dataset. According to the results in Table 10, the $CETRA - CIS$ models demonstrate a good balance between accuracy and throughput. The $CETRA - CIS_1$ model achieves an accuracy of 94.88% with a throughput of 63.72 samples per second and a latency of 0.0314 seconds. Among the $CETRA - CIS$ models, $CETRA - CIS_5$ has the highest throughput at 172.41 samples per second, but it compromises slightly on accuracy, achieving 94.17%. The $CETRA - CIS_4$ model provides a slightly higher accuracy of 95.48% with a throughput of 111.52 samples per second and a lower latency of 0.0179 seconds, making it a well-rounded option.

The $ACER - CIS$ models generally exhibit higher accuracy and throughput compared to $CETRA - CIS$ models. The $ACER - CIS$ model stands out with an accuracy of 97.76%, a high throughput of 213 samples per second, and a very low latency of 0.0094 seconds. This model is particularly effective

Table 10: Performance Comparison of Inference Systems for High-Throughput Toxicity Detection on the $\hat{D}^{\text{test}}_{\text{AugmenToxic}}$

Model	**Action Space**	**Accuracy$_\alpha$ (%)**	**Precision$_\rho$ (%)**	**Recall$_\sigma$ (%)**	**F1-score$_\phi$ (%)**	**Throughput$_\psi$ (samples/s)**	**Latency$_\lambda$ (s)**
$BERT_\alpha$	-	93.22%	92.38%	94.20%	93.28%	4.595	0.2176
CNN_ψ	-	90.33%	90.27%	94.30%	92.09%	269.58	0.0037
$CETRA-CIS_1$	1	94.88%	93.59%	92.20%	92.89%	63.72	0.0314
$CETRA-CIS_2$	1	95.65%	93.24%	96.79%	94.99%	97.12	0.0206
$CETRA-CIS_3$	1	94.38%	94.31%	95.91%	95.11%	83.33	0.0240
$CETRA-CIS_4$	1	95.48%	94.65%	95.79%	95.22%	111.52	0.0179
$CETRA-CIS_5$	1	94.17%	93.78%	95.29%	94.53%	172.41	0.0116
$ACER-CIS$	2	97.76%	97.60%	97.82%	97.71%	213	0.0094
$ACER-CIS_1$	2	95.80%	95.13%	96.70%	95.90%	107.70	0.0186
$ACER-CIS_2$	2	97.70%	96.83%	98.38%	97.60%	105	0.0190
$ACER-CIS_3$	2	96.55%	95.88%	96.93%	96.40%	110	0.0182
$ACER-CIS_4$	2	95.05%	95.38%	95.94%	95.66%	119	0.0168
$ACER-CIS_5$	2	96.12%	95.45%	96.00%	95.72%	160	0.0125
$PPO-CIS$	2	98.17%	97.96%	98.10%	98.03%	384	0.0052
$PPO-CIS_1$	2	96.03%	92.23%	96.95%	94.54%	146	0.0137
$PPO-CIS_2$	2	97.81%	96.40%	98.43%	97.41%	185	0.0108
$PPO-CIS_3$	2	97.05%	96.88%	97.34%	97.11%	251.89	0.0079
$PPO-CIS_4$	2	97.80%	95.13%	98.00%	96.54%	168	0.0119
$PPO-CIS_5$	2	96.50%	95.83%	97.38%	96.60%	203	0.0099

Table 11: Inference Systems Comparison for Toxicity Detection on $\hat{D}^{\text{test}}_{\text{ToxiGen}}$ Dataset

Model	**Action Space**	**Accuracy$_\alpha$ (%)**	**Precision$_\rho$ (%)**	**Recall$_\sigma$ (%)**	**F1-score$_\phi$ (%)**	**Throughput$_\psi$ (samples/s)**	**Latency$_\lambda$ (s)**
$BERT_\alpha$	-	93.91%	90.52%	95.45%	92.92%	4.10	0.2442
CNN_ψ	-	90.09%	88.34%	92.78%	90.83%	52.56	0.0190
$CETRA-CIS_1$	1	92.89%	92.89%	92.89%	92.89%	47.11	0.0425
$CETRA-CIS_2$	1	94.35%	94.14%	94.58%	94.36%	86.90	0.0230
$CETRA-CIS_3$	1	93.04%	92.98%	93.08%	93.03%	79.56	0.0251
$CETRA-CIS_4$	1	93.68%	93.60%	93.65%	93.62%	90.57	0.0221
$CETRA-CIS_5$	1	92.96%	91.83%	93.04%	92.43%	100.3	0.0199
$ACER-CIS$	2	96.87%	96.20%	96.92%	96.56%	206	0.0097
$ACER-CIS_1$	2	95.34%	95.27%	95.39%	95.33%	144	0.0139
$ACER-CIS_2$	2	94.89%	93.92%	95.17%	94.54%	120	0.0167
$ACER-CIS_3$	2	94.01%	93.35%	94.90%	94.12%	127	0.0157
$ACER-CIS_4$	2	95.75%	95.08%	96.65%	95.86%	135	0.0148
$ACER-CIS_5$	2	95.45%	94.78%	96.33%	95.55%	152	0.0132
$PPO-CIS$	2	97.03%	96.81%	97.95%	97.38%	296	0.0068
$PPO-CIS_1$	2	95.10%	93.93%	96.05%	94.98%	113	0.0177
$PPO-CIS_2$	2	96.25%	95.81%	96.37%	96.09%	125	0.0160
$PPO-CIS_3$	2	95.64%	93.99%	96.01%	94.99%	143	0.0140
$PPO-CIS_4$	2	95.92%	93.63%	96.57%	95.08%	182	0.0110
$PPO-CIS_5$	2	95.21%	94.95%	95.84%	95.39%	195	0.0103

for scenarios requiring high accuracy and fast processing times. Similarly, the $ACER-CIS_2$ model achieves an accuracy of 97.70% and a throughput of 105 samples per second with a latency of 0.0190 seconds.

Our proposed approach, the $PPO-CIS$ model, offers the highest accuracy among all models tested. The $PPO-CIS$ model achieves an impressive accuracy of 98.17%, with a throughput of 384 samples per second and a latency of 0.0052 seconds. This model is highly efficient, providing both high accuracy and fast processing times. The $PPO-CIS_3$ model also performs well, with an accuracy of 97.05%, a throughput of 251.89 samples per second, and a latency of 0.0079 seconds.

According to the results in Table 11, the $CETRA-CIS$ models demonstrate varied performance in terms of accuracy and throughput. The $CETRA-CIS_1$ model achieves an accuracy of 92.89% with a throughput of 47.11 samples per second and a latency of 0.0425 seconds. Among the $CETRA-CIS$ models, $CETRA-CIS_5$ has the highest throughput at 100.3 samples per second, but it compromises slightly on accuracy, achieving 92.96%. The $CETRA-CIS_4$ model offers a balance with an accuracy of 93.68% and a throughput of 90.57 samples per second, with a latency of 0.0221 seconds.

The $ACER-CIS$ models generally exhibit higher accuracy and throughput compared to $CETRA-CIS$ models. The $ACER-CIS$ model stands out with an accuracy of 96.87%, a high throughput of 206 samples per second, and a very low latency of 0.0097 seconds. Similarly, the $ACER-CIS_3$ model achieves an accuracy of 94.01% and a throughput of 127 samples per second with a latency of 0.0157 seconds.

The $PPO-CIS$ model maintains high performance, achieving an accuracy of 97.03%, a throughput of 296 samples per second, and a latency of 0.0068 seconds. The *PPO-CIS*$_5$ model also performs well, with an accuracy of 95.21%, a throughput of 195 samples per second, and a latency of 0.0103 seconds. Overall, the proposed Cascade Inference System (CIS) with two action spaces and a reward function that jointly considers accuracy and throughput delivers the best performance when combined with PPO. The CIS using ACER ($ACER-CIS$) also demonstrates a strong balance between accuracy and latency, confirming the effectiveness of the cascade design and reward formulation. While our experiments on two benchmark toxicity datasets demonstrate consistent improvements in both accuracy and efficiency, we recognize that real-time environments involve additional dynamics such as fluctuating content volume and variable latency. Evaluating these conditions requires a controlled social media stream simulation to enable true real-time performance analysis, which we are currently developing as future work. Nonetheless, the present results provide strong evidence that PPO-CIS outperforms existing approaches under standard evaluation settings.

6. Discussion

Existing toxicity detection approaches often face a trade-off between speed and accuracy. Fast models process content quickly but may miss subtle toxicity, while more accurate models are too slow for real-time use. Ensemble methods improve performance but typically apply several models to every input, increasing computational cost. RL has been explored in related domains, but it has not been used to manage classifier selection in cascaded toxicity detection. PPO-CIS addresses these limitations by activating classifiers only when needed. The first stage filters easy, clearly nontoxic samples, and the second stage focuses on harder cases using more accurate models. The PPO agent learns this selection strategy through a reward function that balances accuracy and throughput. Experiments show that PPO-CIS outperforms static cascades and individual classifiers on two benchmark datasets. The main contributions of this work are: (1) a cascaded inference design tailored for real-time moderation, (2) a RL strategy for dynamic classifier selection, and (3) a reward function supporting both accuracy and efficiency.

7. Future Work

Future work will extend PPO-CIS in several directions. First, we plan to evaluate the system in a real-time social media stream environment, where input volume and latency fluctuate dynamically. This will allow us to measure system behavior under realistic traffic conditions. Additionally, integrating multimodal signals (e.g., images, audio, and short videos) can improve detection when harmful content is expressed beyond text. Finally, incorporating fairness-aware objectives will help reduce unintended biases across demographic or linguistic groups.

8. Conclusion

This work introduced PPO-CIS, a cascade inference system guided by Proximal Policy Optimization to balance accuracy and computational efficiency in toxicity detection. By selecting classifiers dynamically based on confidence and throughput constraints, PPO-CIS reduces unnecessary computation while maintaining high accuracy. Experiments on two benchmark datasets show that PPO-CIS improves both performance and efficiency over static and standalone baselines. These results highlight the potential of adaptive, reinforcement learning–driven moderation pipelines for real-time content analysis and platform-scale deployment.

Acknowledgments

This research is supported by NSERC Discovery Grants (RGPIN-2024-04087) and Canada Research Chairs Program (CRC-2019-00041).

References

Agarwal, S., Sonawane, A., Chowdary, C.R., 2023. Accelerating automatic hate speech detection using parallelized ensemble learning models. Expert Systems with Applications 230, 120564. URL: https://www.sciencedirect.com/science/article/pii/S0957417423010667, doi:10.1016/j.eswa.2023.120564.

Aind, A.T., Ramnaney, A., Sethia, D., 2020. Q-Bully: A Reinforcement Learning based Cyberbullying Detection Framework, in: 2020 International Conference for Emerging Technology (INCET), pp. 1–6. URL: https://ieeexplore.ieee.org/document/9154092, doi:10.1109/INCET49848.2020.9154092.

Aljero, M.K.A., Dimililer, N., 2021. A Novel Stacked Ensemble for Hate Speech Recognition. Applied Sciences 11, 11684. URL: https://www.mdpi.com/2076-3417/11/24/11684, doi:10.3390/app112411684. number: 24 Publisher: Multidisciplinary Digital Publishing Institute.

Alkomah, F., Ma, X., 2022. A Literature Review of Textual Hate Speech Detection Methods and Datasets. Information 13, 273. URL: https://www.mdpi.com/2078-2489/13/6/273, doi:10.3390/info13060273. number: 6 Publisher: Multidisciplinary Digital Publishing Institute.

Amarjyoti, S., 2017. Deep Reinforcement Learning for Robotic Manipulation-The state of the art. URL: http://arxiv.org/abs/1701.08878. arXiv:1701.08878 [cs].

Andriotis, C.P., Papakonstantinou, K.G., 2019. Managing engineering systems with large state and action spaces through deep reinforcement learning. Reliability Engineering & System Safety 191, 106483. URL: https://www.sciencedirect.com/science/article/pii/S0951832018313309, doi:10.1016/j.ress.2019.04.036.

Androcec, D., 2020. Machine learning methods for toxic comment classification: a systematic review. Acta Universitatis Sapientiae, Informatica 12, 205–216. doi:10.2478/ausi-2020-0012.

Anjum, Katarya, R., 2024. Hate speech, toxicity detection in online social media: a recent survey of state of the art and opportunities. International Journal of Information Security 23, 577–608. URL: https://doi.org/10.1007/s10207-023-00755-2, doi:10.1007/s10207-023-00755-2.

Arnold, L., Rebecchi, S., Chevallier, S., Paugam-Moisy, H., 2011. An Introduction to Deep Learning. URL: https://www.semanticscholar.org/paper/An-Introduction-to-Deep-Learning-Arnold-Rebecchi/7fc7bb4eec2f27b8f0a0c7fb2a4112c7a7a7abed.

Arulkumaran, K., Deisenroth, M.P., Brundage, M., Bharath, A.A., 2017. Deep Reinforcement Learning: A Brief Survey. IEEE Signal Processing Magazine 34, 26–38. URL: https://ieeexplore.ieee.org/abstract/document/8103164, doi:10.1109/MSP.2017.2743240. conference Name: IEEE Signal Processing Magazine.

Bahdanau, D., Brakel, P., Xu, K., Goyal, A., Lowe, R., Pineau, J., Courville, A., Bengio, Y., 2017. An Actor-Critic Algorithm for Sequence Prediction. URL: http://arxiv.org/abs/1607.07086, doi:10.48550/arXiv.1607.07086. arXiv:1607.07086 [cs].

Bansal, G., Nushi, B., Kamar, E., Horvitz, E., Weld, D.S., 2021. Is the Most Accurate AI the Best Teammate? Optimizing AI for Teamwork. URL: http://arxiv.org/abs/2004.13102, doi:10.48550/arXiv.2004.13102. arXiv:2004.13102 [cs].

Beniwal, R., Maurya, A., 2021. Toxic Comment Classification Using Hybrid Deep Learning Model. doi:10.1007/978-981-15-8677-4_38.

Binxiang, L., Gang, Z., Ruoying, S., 2019. A Deep Reinforcement Learning Malware Detection Method Based on PE Feature Distribution, in:

2019 6th International Conference on Information Science and Control Engineering (ICISCE), pp. 23–27. URL: https://ieeexplore.ieee.org/abstract/document/9107644, doi:10.1109/ICISCE48695.2019.00014.

Birman, Y., Hindi, S., Katz, G., Shabtai, A., 2022. Cost-effective ensemble models selection using deep reinforcement learning. Information Fusion 77, 133–148. URL: https://www.sciencedirect.com/science/article/pii/S1566253521001524, doi:10.1016/j.inffus.2021.07.011.

Bodaghi, A., Fung, B.C.M., A. Schmitt, K., 2024. AugmenToxic: Leveraging Reinforcement Learning to Optimize LLM Instruction Fine-Tuning for Data Augmentation to Enhance Toxicity Detection. ACM Trans. Web URL: https://dl.acm.org/doi/10.1145/3700791, doi:10.1145/3700791.

Bodaghi, A., Fung, B.C.M., Schmitt, K.A., 2023. Technological Solutions to Online Toxicity: Potential and Pitfalls. IEEE Technology and Society Magazine 42, 57–65. doi:10.1109/MTS.2023.3340235.

Bodaghi, A., Fung, B.C.M., Shahen, J., 2025. A Profit-driven Simulation (PDS) Framework for Comparison of Deep Learning Models for Real-time Toxicity Detection in Social Media. Information Processing & Management , 1–31Under Review.

Breiman, L., 1996. Bagging predictors. Machine Learning 24, 123–140. URL: https://doi.org/10.1007/BF00058655, doi:10.1007/BF00058655.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W., 2016. OpenAI Gym. URL: http://arxiv.org/abs/1606.01540, doi:10.48550/arXiv.1606.01540. arXiv:1606.01540 [cs].

Cerruto, F., Cirillo, S., Desiato, D., Gambardella, S.M., Polese, G., 2022. Social network data analysis to highlight privacy threats in sharing data. Journal of Big Data 9, 19. URL: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-022-00566-7, doi:10.1186/s40537-022-00566-7.

Chen, M., Xu, Z., Weinberger, K., Chapelle, O., Kedem, D., 2012. Classifier Cascade for Minimizing Feature Evaluation Cost, in: Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics, PMLR. pp. 218–226. URL: https://proceedings.mlr.press/v22/chen12c.html. iSSN: 1938-7228.

Cho, K., van Merrienboer, B., Bahdanau, D., Bengio, Y., 2014. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. URL: http://arxiv.org/abs/1409.1259, doi:10.48550/arXiv.1409.1259. arXiv:1409.1259 [cs, stat].

Chung, H.W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., Webson, A., Gu, S.S., Dai, Z., Suzgun, M., Chen, X., Chowdhery, A., Castro-Ros, A., Pellat, M., Robinson, K., Valter, D., Narang, S., Mishra, G., Yu, A., Zhao, V., Huang, Y., Dai, A., Yu, H., Petrov, S., Chi, E.H., Dean, J., Devlin, J., Roberts, A., Zhou, D., Le, Q.V., Wei, J., 2022. Scaling Instruction-Finetuned Language Models. URL: http://arxiv.org/abs/2210.11416. arXiv:2210.11416 [cs].

Cirillo, S., Desiato, D., Polese, G., Solimando, G., Sugumaran, V., Sundaramurthy, S., 2025. Exploring the ability of emerging large language models to detect cyberbullying in social posts through new prompt-based classification approaches. Information Processing & Management 62, 104043. URL: https://www.sciencedirect.com/science/article/pii/S0306457324004023, doi:10.1016/j.ipm.2024.104043.

Cirillo, S., Desiato, D., Scalera, M., Solimando, G., 2023. A Visual Privacy Tool to Help Users in Preserving Social Network Data., in: IS-EUD Workshops. URL: https://ceur-ws.org/Vol-3408/short-s3-08.pdf.

Devlin, J., Chang, M.W., Lee, K., Toutanova, K., 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs] URL: http://arxiv.org/abs/1810.04805. arXiv: 1810.04805.

Dhingra, B., Li, L., Li, X., Gao, J., Chen, Y.N., Ahmed, F., Deng, L., 2017. Towards End-to-End Reinforcement Learning of Dialogue Agents for Information Access. URL: http://arxiv.org/abs/1609.00777, doi:10.48550/arXiv.1609.00777. arXiv:1609.00777 [cs].

Dong, H., Ding, Z., Zhang, S. (Eds.), 2020. Deep Reinforcement Learning: Fundamentals, Research and Applications. Springer Singapore, Singapore. URL: http://link.springer.com/10.1007/978-981-15-4095-0, doi:10.1007/978-981-15-4095-0.

Faal, F., Schmitt, K., Yu, J.Y., 2023. Reward Modeling for Mitigating Toxicity in Transformer-based Language Models. Applied Intelligence 53, 8421–8435. URL: http://arxiv.org/abs/2202.09662, doi:10.1007/s10489-022-03944-z. arXiv:2202.09662 [cs].

Filar, J., Vrieze, K., 2012. Competitive Markov Decision Processes. Springer Science & Business Media. Google-Books-ID: uXDjBwAAQBAJ.

Francois-Lavet, V., Henderson, P., Islam, R., Bellemare, M.G., Pineau, J., 2018. An Introduction to Deep Reinforcement Learning. Foundations and Trends® in Machine Learning 11, 219–354. URL: http://arxiv.org/abs/1811.12560, doi:10.1561/2200000071. arXiv:1811.12560 [cs, stat].

Ganesh, B., Bright, J., 2020. Countering Extremists on Social Media: Challenges for Strategic Communication and Content Moderation. Policy & Internet 12, 6–19. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/poi3.236, doi:10.1002/poi3.236. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/poi3.236.

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT Press. Google-Books-ID: omivDQAAQBAJ.

Gorwa, R., Binns, R., Katzenbach, C., 2020. Algorithmic content moderation: Technical and political challenges in the automation of platform governance. Big Data & Society 7, 2053951719897945. URL: https://doi.org/10.1177/2053951719897945, doi:10.1177/2053951719897945. publisher: SAGE Publications Ltd.

Graves, A., Schmidhuber, J., 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Networks 18, 602–610. URL: https://www.sciencedirect.com/science/article/pii/S0893608005001206, doi:10.1016/j.neunet.2005.06.042.

Hartvigsen, T., Gabriel, S., Palangi, H., Sap, M., Ray, D., Kamar, E., 2022. ToxiGen: A Large-Scale Machine-Generated Dataset for Adversarial and Implicit Hate Speech Detection. URL: http://arxiv.org/abs/2203.09509, doi:10.48550/arXiv.2203.09509. arXiv:2203.09509 [cs].

Hochreiter, S., Schmidhuber, J., 1997. Long Short-Term Memory. Neural Comput. 9, 1735–1780. URL: https://doi.org/10.1162/neco.1997.9.8.1735, doi:10.1162/neco.1997.9.8.1735.

Jahan, M.S., Oussalah, M., 2023. A systematic review of hate speech automatic detection using natural language processing. Neurocomputing 546, 126232. URL: https://www.sciencedirect.com/science/article/pii/S0925231223003557, doi:10.1016/j.neucom.2023.126232.

Jiao, X., Yin, Y., Shang, L., Jiang, X., Chen, X., Li, L., Wang, F., Liu, Q., 2020. TinyBERT: Distilling BERT for Natural Language Understanding. URL: http://arxiv.org/abs/1909.10351, doi:10.48550/arXiv.1909.10351. arXiv:1909.10351 [cs].

Jigsaw, G., 2017. Jigsaw Toxic Comment Classification Challenge. URL: https://kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge.

Kaelbling, L.P., Littman, M.L., Moore, A.W., 1996. Reinforcement Learning: A Survey. Journal of Artificial Intelligence Research 4, 237–285. URL: https://www.jair.org/index.php/jair/article/view/10166, doi:10.1613/jair.301.

Kirkpatrick, D., 2016. Google: 53% of mobile users abandon sites that take over 3 seconds to load.

Kivlichan, I.D., Lin, Z., Liu, J., Vasserman, L., 2021. Measuring and Improving Model-Moderator Collaboration using Uncertainty Estimation. URL: http://arxiv.org/abs/2107.04212, doi:10.48550/arXiv.2107.04212. arXiv:2107.04212 [cs].

Kokatnoor, S.A., Krishnan, B., 2020. Twitter Hate Speech Detection using Stacked Weighted Ensemble (SWE) Model, in: 2020 Fifth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN), pp. 87–92. URL: https://ieeexplore.ieee.org/document/9296199, doi:10.1109/ICRCICN50933.2020.9296199.

Lavie, O., Katz, G., Shabtai, A., 2023. Cost Effective Transfer of Reinforcement Learning Policies. URL: https://papers.ssrn.com/abstract=4341615, doi:10.2139/ssrn.4341615.

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521, 436–444. URL: https://www.nature.com/articles/nature14539, doi:10.1038/nature14539. publisher: Nature Publishing Group.

Li, X., Chen, Y.N., Li, L., Gao, J., Celikyilmaz, A., 2018. End-to-End Task-Completion Neural Dialogue Systems. URL: http://arxiv.org/abs/1703.01008, doi:10.48550/arXiv.1703.01008. arXiv:1703.01008 [cs].

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V., 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. URL: http://arxiv.org/abs/1907.11692, doi:10.48550/arXiv.1907.11692. arXiv:1907.11692 [cs].

Malik, P., Aggrawal, A., Vishwakarma, D.K., 2021. Toxic Speech Detection using Traditional Machine Learning Models and BERT and fastText Embedding with Deep Neural Networks, in: 2021 5th International Conference on Computing Methodologies and Communication (ICCMC), pp. 1254–1259. URL: https://ieeexplore.ieee.org/abstract/document/9418395, doi:10.1109/ICCMC51019.2021.9418395.

Maslej-Krešňáková, V., Sarnovský, M., Butka, P., Machová, K., 2020. Comparison of Deep Learning Models and Various Text Pre-Processing Techniques for the Toxic Comments Classification. Applied Sciences 10, 8631. URL: https://www.mdpi.com/2076-3417/10/23/8631, doi:10.3390/app10238631. number: 23 Publisher: Multidisciplinary Digital Publishing Institute.

Melton, J., Bagavathi, A., Krishnan, S., 2020. DeL-haTE: A Deep Learning Tunable Ensemble for Hate Speech Detection, in: 2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA), IEEE, Miami, FL, USA. pp. 1015–1022. URL: https://ieeexplore.ieee.org/document/9356174/, doi:10.1109/ICMLA51294.2020.00165.

Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., Joulin, A., 2017. Advances in Pre-Training Distributed Word Representations. URL: http://arxiv.org/abs/1712.09405, doi:10.48550/arXiv.1712.09405. arXiv:1712.09405 [cs].

Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T.P., Harley, T., Silver, D., Kavukcuoglu, K., 2016. Asynchronous Methods for Deep Reinforcement Learning. URL: http://arxiv.org/abs/1602.01783, doi:10.48550/arXiv.1602.01783. arXiv:1602.01783 [cs].

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D., 2015. Human-level control through deep reinforcement learning. Nature 518, 529–533. URL: https://www.nature.com/articles/nature14236, doi:10.1038/nature14236. publisher: Nature Publishing Group.

Museng, F., Jessica, A., Wijaya, N., Anderies, A., Iswanto, I.A., 2022. Systematic Literature Review: Toxic Comment Classification, in: 2022 IEEE 7th International Conference on Information Technology and Digital Applications (ICITDA), IEEE, Yogyakarta, Indonesia. pp. 1–7. URL: https://ieeexplore.ieee.org/document/9971338/, doi:10.1109/ICITDA55840.2022.9971338.

Nandy, A., Biswas, M., 2018. Reinforcement Learning with Keras, TensorFlow, and ChainerRL, in: Nandy, A., Biswas, M. (Eds.), Reinforcement Learning : With Open AI, TensorFlow and Keras Using Python. Apress, Berkeley, CA, pp. 129–153. URL: https://doi.org/10.1007/978-1-4842-3285-9_5, doi:10.1007/978-1-4842-3285-9_5.

Nascimento, F.R.S., Cavalcanti, G.D.C., Da Costa-Abreu, M., 2022. Unintended bias evaluation: An analysis of hate speech detection and gender bias mitigation on social media using ensemble learning. Expert Systems with Applications 201, 117032. URL: https://www.sciencedirect.com/science/article/pii/S095741742200447X, doi:10.1016/j.eswa.2022.117032.

O'Shea, K., Nash, R., 2015. An Introduction to Convolutional Neural Networks. URL: http://arxiv.org/abs/1511.08458, doi:10.48550/arXiv.1511.08458. arXiv:1511.08458 [cs].

Paisitkriangkrai, S., 2011. Robust object detection with efficient features and effective classifiers. PhD Thesis. UNSW Sydney. URL: https://unsworks.unsw.edu.au/entities/publication/cd8df6ea-322a-46f0-b485-d094c6c649fd/full.

Pitsilis, G.K., Ramampiaro, H., Langseth, H., 2018. Detecting Offensive Language in Tweets Using Deep Learning. Applied Intelligence 48, 4730–4742. URL: http://arxiv.org/abs/1801.04433, doi:10.1007/s10489-018-1242-y. arXiv:1801.04433 [cs].

Polikar, R., 2012. Ensemble Learning, in: Zhang, C., Ma, Y. (Eds.), Ensemble Machine Learning: Methods and Applications. Springer, New York, NY, pp. 1–34. URL: https://doi.org/10.1007/978-1-4419-9326-7_1, doi:10.1007/978-1-4419-9326-7_1.

Poojitha, K., Charish, A.S., Reddy, M.A.K., Ayyasamy, S., 2023. Classification of social media Toxic comments using Machine learning models. URL: http://arxiv.org/abs/2304.06934. arXiv:2304.06934 [cs].

Raj, V.S., Subalalitha, C.N., Sambath, L., Glavin, F., Chakravarthi, B.R., 2024. ConBERT-RL: A policy-driven deep reinforcement learning based approach for detecting homophobia and transphobia in low-resource languages. Natural Language Processing Journal 6, 100040. URL: https://www.sciencedirect.com/science/article/pii/S2949719123000377, doi:10.1016/j.nlp.2023.100040.

Schapire, R.E., 1990. The strength of weak learnability. Machine Learning 5, 197–227. URL: https://doi.org/10.1007/BF00116037, doi:10.1007/BF00116037.

Schulman, J., Levine, S., Moritz, P., Jordan, M.I., Abbeel, P., 2017a. Trust Region Policy Optimization. URL: http://arxiv.org/abs/1502.05477, doi:10.48550/arXiv.1502.05477. arXiv:1502.05477 [cs].

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O., 2017b. Proximal Policy Optimization Algorithms. URL: http://arxiv.org/abs/1707.06347, doi:10.48550/arXiv.1707.06347. arXiv:1707.06347 [cs].

Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis, D., 2016. Mastering the game of Go with deep neural networks and tree search. Nature 529, 484–489. URL: https://www.nature.com/articles/

nature16961, doi:10.1038/nature16961. publisher: Nature Publishing Group.

Singh, A., 2020. Detecting Toxicity in a Diverse Online Conversation using Reinforcement Learning. Master's thesis. University of Maryland, Baltimore County. URL: https://search.proquest.com/openview/b7af8344c480abe7ba0a9b705dce511c/1?pq-origsite=gscholar&cbl=18750&diss=y.

Singh, V.K., Shrivastava, U., Bouayad, L., Padmanabhan, B., Ialynytchev, A., Schultz, S.K., 2018. Machine learning for psychiatric patient triaging: an investigation of cascading classifiers. Journal of the American Medical Informatics Association 25, 1481–1487. URL: https://doi.org/10.1093/jamia/ocy109, doi:10.1093/jamia/ocy109.

Sutton, C.D., 2005. 11 - Classification and Regression Trees, Bagging, and Boosting, in: Rao, C.R., Wegman, E.J., Solka, J.L. (Eds.), Handbook of Statistics. Elsevier. volume 24 of *Data Mining and Data Visualization*, pp. 303–329. URL: https://www.sciencedirect.com/science/article/pii/S0169716104240111, doi:10.1016/S0169-7161(04)24011-1.

Sutton, R.S., Barto, A.G., 2018. Reinforcement Learning, second edition: An Introduction. MIT Press. Google-Books-ID: uWV0DwAAQBAJ.

Sutton, R.S., McAllester, D., Singh, S., Mansour, Y., 1999. Policy Gradient Methods for Reinforcement Learning with Function Approximation, in: Advances in Neural Information Processing Systems, MIT Press. URL: https://proceedings.neurips.cc/paper_files/paper/1999/hash/464d828b85b0bed98e80ade0a5c43b0f-Abstract.html.

Urbaniak, R., Ptaszyński, M., Tempska, P., Leliwa, G., Brochocki, M., Wroczyński, M., 2022. Personal attacks decrease user activity in social networking platforms. Computers in Human Behavior 126, 106972. URL: https://www.sciencedirect.com/science/article/pii/S0747563221002958, doi:10.1016/j.chb.2021.106972.

Viola, P., Jones, M., 2001. Rapid object detection using a boosted cascade of simple features, in: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, IEEE Comput. Soc, Kauai, HI, USA. pp. I–511–I–518.

URL: http://ieeexplore.ieee.org/document/990517/, doi:10.1109/CVPR.2001.990517.

Wang, Y., Geng, S., Gao, H., 2018. A proactive decision support method based on deep reinforcement learning and state partition. Knowledge-Based Systems 143, 248–258. URL: https://www.sciencedirect.com/science/article/pii/S095070511730504X, doi:10.1016/j.knosys.2017.11.005.

Wang, Z., Bapst, V., Heess, N., Mnih, V., Munos, R., Kavukcuoglu, K., de Freitas, N., 2017. Sample Efficient Actor-Critic with Experience Replay. URL: http://arxiv.org/abs/1611.01224, doi:10.48550/arXiv.1611.01224. arXiv:1611.01224 [cs].

Waseem, Z., Davidson, T., Warmsley, D., Weber, I., 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks, in: Waseem, Z., Chung, W.H.K., Hovy, D., Tetreault, J. (Eds.), Proceedings of the First Workshop on Abusive Language Online, Association for Computational Linguistics, Vancouver, BC, Canada. pp. 78–84. URL: https://aclanthology.org/W17-3012, doi:10.18653/v1/W17-3012.

Xia, Y., He, D., Qin, T., Wang, L., Yu, N., Liu, T.Y., Ma, W.Y., 2016. Dual Learning for Machine Translation. URL: http://arxiv.org/abs/1611.00179. arXiv:1611.00179 [cs].

Zhang, X., Zou, J., He, K., Sun, J., 2016. Accelerating Very Deep Convolutional Networks for Classification and Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 38, 1943–1955. URL: https://ieeexplore.ieee.org/document/7332968, doi:10.1109/TPAMI.2015.2502579.

Zhu, J., Wu, F., Zhao, J., 2021. An Overview of the Action Space for Deep Reinforcement Learning, in: 2021 4th International Conference on Algorithms, Computing and Artificial Intelligence, ACM, Sanya China. pp. 1–10. URL: https://dl.acm.org/doi/10.1145/3508546.3508598, doi:10.1145/3508546.3508598.

9. Appendices

10. Appendix A

10.0.1. Reinforcement Learning (RL)

Reinforcement Learning (RL) is a ML approach where an agent learns to make decisions by interacting with an environment. The agent selects actions a from a set A, receives rewards r, and adjusts its strategy to maximize cumulative rewards over time. RL involves sequences of states s, actions a, and rewards r (denoted as trajectories τ).

The main objective in RL is for the agent to learn a policy (π), which maps states to actions to maximize expected cumulative rewards. An agent's policy can be either deterministic or stochastic (Andriotis and Papakonstantinou, 2019). A deterministic policy, denoted as $\pi : S \to A$, maps each state directly to a specific action. In contrast, a stochastic policy, denoted as $\pi(a|s) : S \to P(A)$, maps each state to a probability distribution over possible actions, indicating the likelihood of selecting each action given a particular state. While deterministic policies provide clear action choices, they may be limited in adversarial environments and finding the optimal action with respect to the Q-function can be computationally expensive in large action spaces (Wang et al., 2017).

RL algorithms are evaluated based on their ability to develop effective strategies across various environments. During interactions, the agent selects actions a_t at time step t, transitioning from state s_t to s_{t+1} and receiving reward r_t. The objective is to maximize the total future rewards R_T:

$$R_T = \sum_{t=1}^{T} r_t$$

where T is the final time step.

The policy $\pi(a, s)$ dictates the probability of selecting action a in state s. Concurrently, the Q-function $Q(s, a)$ estimates the expected future reward for taking action a in state s, under the current policy π.

RL algorithms are evaluated based on their capacity to develop effective strategies across diverse environments. Through RL, agents learn to explore actions, exploit learned knowledge, and maximize long-term rewards in dynamic and complex environments.

10.0.2. Deep Reinforcement Learning

Deep Reinforcement Learning (DRL) represents an advanced approach that combines the principles of RL with the power of Deep Learning (DL) (Arnold et al., 2011; Francois-Lavet et al., 2018). Unlike traditional RL, which often faces challenges in handling high-dimensional and complex data spaces, DRL leverages Deep Neural Networks (DNNs) to effectively manage and process such data (Dong et al., 2020; O'Shea and Nash, 2015). In contrast, DRL utilizes DNNs to approximate the Q-function $Q(s, a; \theta)$, where θ represents the parameters of the neural network (Mnih et al., 2015). This approach is highly effective for handling large or continuous state-action spaces (Amarjyoti, 2017). Consequently, while the Q-function in traditional RL is simpler and less scalable, the Q-function in DRL provides superior representational power and generalization capabilities, enabling it to perform well in complex environments (Andriotis and Papakonstantinou, 2019).

10.1. Actor-Critic with Experience Replay (ACER)

Actor-critic with experience replay (ACER) (Wang et al., 2017) is a type of actor-critic (Mnih et al., 2016) method, which means it comprises two main components: the actor and the critic. The actor is responsible for selecting actions based on a policy $\pi(s; \theta_\pi)$, where θ_π are the parameters of the policy network. The critic, on the other hand, evaluates the actions taken by the actor by estimating the value function $Q(s, a; \theta_Q)$, where θ_Q are the parameters of the value network. This dual structure allows ACER to simultaneously learn a policy for selecting actions and a value function for assessing the quality of those actions. One of the key innovations in ACER is the use of experience replay. In traditional actor-critic methods, updates are made using the most recent experiences, which can lead to inefficient learning and instability. ACER addresses this issue by storing past experiences in a replay buffer and using these experiences to perform off-policy updates. This approach not only improves sample efficiency by reusing past experiences but also stabilizes training by breaking the correlation between consecutive updates. Another critical aspect of ACER is the use of a trust region policy optimization (TRPO) (Schulman et al., 2017a) approach to ensure that the updates to the policy do not deviate too much from the current policy, preventing large and potentially destabilizing changes. Additionally, ACER employs importance sampling techniques to correct for any bias introduced by using off-policy data from the replay buffer. In summary, ACER enhances traditional actor-critic methods by integrating experience replay for better sample efficiency, trust region optimization for stable policy updates, and importance sampling for unbiased learning. These features enable ACER to effectively learn in complex and high-dimensional environments, making it a powerful algorithm in the field of deep reinforcement learning.

10.2. Proximal Policy Optimization (PPO)

PPO (Schulman et al., 2017b) is designed to improve the stability and reliability of policy gradient (Sutton et al., 1999) methods. Traditional policy gradient methods can suffer from high variance and instability, as updates to the policy can sometimes lead to large, detrimental changes. PPO addresses these issues by ensuring that policy updates are more controlled and constrained.

PPO achieves this through the use of a surrogate objective function that limits the magnitude of policy changes. The key idea is to optimize a clipped objective function, which penalizes changes to the policy that move too far from the current policy (Zhu et al., 2021). The objective function in PPO can be expressed as:

$$\begin{aligned} L^{CLIP}(\theta) = \mathbb{E}_t\Bigg[\min\Bigg(&\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\hat{A}_t, \\ &\text{clip}\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon\right)\hat{A}_t\Bigg)\Bigg] \end{aligned} \tag{10}$$

where: π_θ is the new policy with parameters θ and $\pi_{\theta_{old}}$ is the old policy with parameters θ_{old}. $\hat{A}_t$ is also the advantage estimate at time step t. ϵ is a hyperparameter that controls the clipping range.

The clipping mechanism in PPO ensures that the ratio $\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ stays within the range $[1-\epsilon, 1+\epsilon]$, thus preventing large deviations from the old policy. This constraint helps to maintain the stability of updates and improves overall training performance.

PPO is widely used in various RL tasks due to its simplicity, robustness, and effectiveness. It strikes a good balance between performance and computational efficiency, making it a preferred choice for many complex and high-dimensional environments.

In summary, PPO enhances traditional policy gradient methods by incorporating a clipped surrogate objective function to ensure controlled and

Table 12: Experimental Configurations for CNN-based Classifiers

Model	Hidden Layers	Epochs	Batch Size	Learning Rate	Dropout Rate	Filters	Max Features	Max Length
CNN_α	4	15	512	5.00E-05	0.4	300	100000	400
CNN_ψ	2	23	512	5.00E-05	0.5	300	100000	400
$fastText_\alpha$	4	5	256	0.0002	0.4	300	100000	400
$fastText_\psi$	2	14	256	5.00E-7	0.5	128	100000	400

Table 13: Experimental Configurations for transformer-based Classifiers

Model	Model Name	Epochs	Batch Size	Learning Rate	Weight Decay	Max Length
$BERT_\alpha$	$bert - base - uncased$	1	8	2E-5	0.01	300
$BERT_\psi$	$google/bert_uncased_L - 2_H - 128_A - 2$	3	32	3E-5	0.01	300
$RoBERTa_\alpha$	$roberta - base$	1	4	2E-5	0.01	300
$DistilRoBERTa_\psi$	$distilroberta - base$	2	16	3E-5	0.01	300

stable policy updates. This approach mitigates the risk of large, destabilizing changes to the policy, resulting in more reliable and efficient learning in reinforcement learning applications.

11. Appendix B

In this section, we describe the experimental setup for the classifiers developed in this research. We utilized the training and validation sets from $D_{\text{AugmenToxic}}$ for all classifier development. To ensure consistency across the classifiers, we applied a random-state approach for the train-test split, which resulted in identical training, validation, and testing sets for each classifier. This uniformity ensured that all eight classifiers were evaluated under the same conditions.

Further details on the development process and parameter settings for the various classifiers are provided in subsection 11.1 and subsection 11.2.

11.1. CNN-based: CNN, CNN-fastText

We began with data cleaning and preprocessing, including the removal of URLs, punctuation, digits, and stop-words, and normalization. The text was tokenized, truncated or padded to a fixed length, and converted into word vectors.

The model was tested with different configurations. For accuracy, the optimal variant used four 1D convolutional layers, while the high-throughput variant achieved better performance with two 1D layers. We applied pooling and global max pooling to reduce dimensionality, used ReLU activations in hidden layers, and a Sigmoid activation in the output layer. Binary cross-entropy loss and the Adam optimizer were employed for training.

To accelerate training, we incorporated pre-trained fastText embeddings ("crawl-300d-2M") into the CNN model, resulting in the CNN-fastText variant. These embeddings, trained on Common Crawl with CBOW, character n-grams, and negative sampling, improved training efficiency. Hyperparameter settings are detailed in Table 12.

To develop high-throughput variants, we applied various techniques, including parameter tuning and adopting a shallower network topology. While deeper networks can improve accuracy, they are computationally expensive and can slow down the process. Thus, a shallower network with fewer convolutional 1D layers can be a good compromise between accuracy and throughput. We reduced the number of convolutional layers to three and two, while still maintaining good accuracy. The highest throughput was achieved with two convolutional 1D layers and a specific set of hyperparameters.

11.2. Transformer-based: BERT, RoBERTa, TinyBERT, DistilRoBERTa

We utilized the Hugging Face Transformers library with TensorFlow 2.15.0 to evaluate various models for toxic content detection. Our assessment included several BERT-based models: bert-base-uncased, bert-large-uncased, ALBERT (albert-base-v2), RoBERTa (roberta-base), GroNLP's hateBERT, and vinai's BERTweet (vinai/bertweet-base), along with their distilled variants and 24 BERT miniature models.

For preprocessing, we employed model-specific tokenizers to convert raw text into tokens compatible with each model. This ensured that the data was appropriately formatted for effective processing.

Our testing revealed that bert-base-uncased delivered the best balance of accuracy and acceptable throughput among the BERT variants. Among the BERT miniatures, BERT-Tiny achieved the highest throughput. For RoBERTa models, roberta-base provided the most accurate results, while DistilRoBERTa demonstrated the highest throughput.

We thoroughly explored various hyperparameters to optimize both accuracy and throughput for each model. The detailed hyperparameter settings and results are provided in Table 13.

This comprehensive evaluation enabled us to select the models that offer the best performance for both accuracy and efficiency in toxic content detection.