MD-ProTector: Multiple Data-Driven Prototype-Based LLM-Generated Text Detection

Anonymous ACL submission

Abstract

As LLM-generated content becomes more sophisticated, detection systems for distinguishing those texts from human-written must operate at scale while handling diverse writing styles and domains. Relying only on standard binary classification formulations can neglect diversity in writing style and domain within each class. We propose MD-ProTector, an encoder-based detector that models internal diversity in both classes using multiple prototypes, which are representative vectors summarizing groups of instances. MD-ProTector represents each class with multiple learnable prototypes that are updated during training. A prototype-to-instance contrastive loss, combined with instance-level contrastive loss, data-driven initialization and dynamic prototype updates, effectively leverages class-internal diversity. Experiments show that MD-ProTector improves detection performance while avoiding bias toward either human-written or LLM-generated texts, and exhibits robustness under distribution shifts from unseen domains and generator models.

1 Introduction

As large language models have become increasingly sophisticated and widely accessible, detecting whether text is human-written or LLM-generated has become essential for ensuring credibility in digital communication (Kwon and Jang, 2025). This is particularly important for blocking automated fake news and phishing attacks, as well as maintaining academic integrity (Najjar et al., 2025). In large-scale deployment scenarios, detection systems must be applied to massive volumes of content that span diverse writing styles and domains (Li et al., 2024). Treating such diversity is important for practical detection systems operating in real-world platform services such as social media and news portals (Wu et al., 2024).

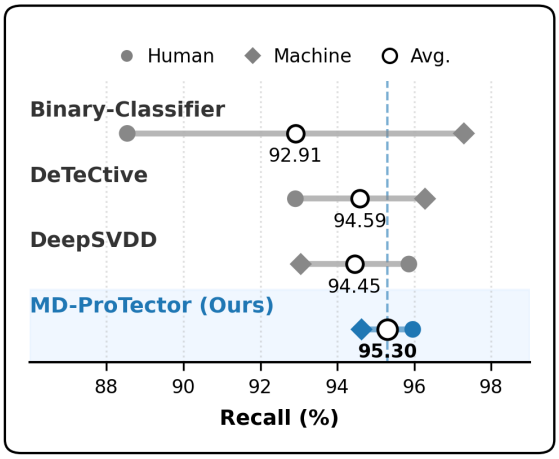

Figure 1: **Unbiased High-Performance Detection.** Standard *Binary Classifiers* exhibit a **biased judgment tendency** (wide gap). Conversely, **MD-ProTector** achieves the highest Average Recall with a minimal gap. This demonstrates consistent, high detection accuracy for both text classes, mitigating the bias of the standard *Binary Classifiers*.

In these practical deployment settings, pre-embedded signals like watermarks or model internals such as log-likelihoods are not guaranteed to utilize (Kandula et al., 2025; Christ et al., 2024; Mitchell et al., 2023). Under these conditions, detectors based on lightweight text encoders provide a practical solution. They operate directly on the input text without requiring access to the generation pipeline or model internals (Kuznetsov et al., 2024). This allows the same detection pipeline to be applied uniformly across diverse writing styles and domains, which is feasible for large-scale deployment (Rodriguez et al., 2022).

One of the simplest designs for encoder-based detectors is to attach a binary classification head and train it using a cross-entropy loss (Ippolito et al., 2020). Despite such intra-class diversity in writing style and domain, all texts are reduced to two categories during training under this design.

The standard binary classification objective is a coarse supervision signal that does not explicitly consider such intra-class diversity (Cui et al., 2016). Several prior works have attempted to move beyond standard binary classification formulations by adding structural constraints, but still overlook intra-class diversity within at least one of the two classes, which can limit detection performance (Guo et al., 2024; Zeng et al., 2025). This observation motivates the need for detectors that model intra-class diversity in both classes, equipped with better reflection on the diversity observed in real-world texts.

For this end, we propose **MD-ProTector**, a novel LLM-generated text detector with multiple data-driven prototypes. Our approach trains the encoder with multiple learnable prototypes, which are representative embedding vectors that summarize groups of instances within each class (Snell et al., 2017). By modeling multiple prototypes for both classes, MD-ProTector is able to capture intra-class diversity on both the human-written and LLM-generated classes in more balanced way. MD-ProTector is mainly trained with prototype-to-instance contrastive loss that aligns each text instance with nearby prototypes. To support stable and effective alignment, the method designs data-driven prototype initialization, dynamic prototype updates, and complementary instance-level consistency during training. Instead of collapsing each class into a single global cluster, MD-ProTector represents both human-written and LLM-generated texts using multiple prototypes.

We assess detection performance of MD-ProTector on a large-scale benchmark containing diverse writing styles and generation patterns. We show that the proposed approach improves detection performance while reducing performance gap between human-written and LLM-generated texts. We further show that MD-ProTector is effective under distribution shifts arising from unseen domains and unseen generator models. In addition, we perform extensive ablation studies to analyze the contribution of each component, validating the effectiveness of multi-prototype modeling, data-driven initialization and training objectives.

Our contributions can be summarized as follows:

- We propose MD-ProTector, a novel detection framework that explicitly models intra-class diversity within human-written and LLM-generated texts by representing each class with multiple data-driven prototypes rather than a single global cluster.

- We develop a prototype-aware training strategy centered on a prototype-to-instance contrastive objective, supported by data-driven prototype initialization and dynamic updates to improve LLM-generated text detection performance.

- We demonstrate the superior detection performance of our method on a large-scale benchmark, validating its generalization capability under distribution shifts and verifying the effectiveness of our design choices through extensive ablation studies.

2 Related Works

2.1 LLM-Generated Text Detection

Prior work on LLM-generated text detection includes watermarking, zero-shot statistical methods, and supervised classifiers, among others, depending on model access and the types of detection signals used (Wu et al., 2025). Watermarking injects identifiable patterns during generation, enabling content source verification by model providers (Kirchenbauer et al., 2023). Zero-shot statistical detectors, such as DetectGPT and GLTR, exploit differences in probability distributions using metrics like perplexity or curvature without requiring labeled data (Mitchell et al., 2023; Gehrmann et al., 2019). However, watermarking assumes prior cooperation from generators, and statistical methods often require repeated model inference, incurring high computational cost (Christ et al., 2024; Bao et al., 2024). As a practical alternative in large-scale deployment scenarios, supervised detectors based on lightweight text encoders operate solely on input text and are suitable for in-the-wild deployment (Bakhtin et al., 2019; Uchendu et al., 2020; Wang et al., 2023). These detectors typically rely on encoder architectures such as BERT (Devlin et al., 2019) or RoBERTa (Liu et al., 2021), whose parameter counts are on the order of hundreds of millions—substantially smaller than the multi-billion-parameter scale of modern LLMs.

Despite their efficiency, these classifiers exhibit significant performance degradation on unseen domains or emerging model families (Bakhtin et al., 2019). Recent work attempts to improve robustness by imposing stronger structural constraints

on the representation space. For example, DeTeCtive organizes text instances from multiple generators into a predefined hierarchical contrastive learning while implicitly treating human-written text as a single homogeneous class (Guo et al., 2024). In contrast, DeepSVDD-based one-class detectors enforce compactness by minimizing the variance of machine-generated samples and modeling human-written text as outliers (Zeng et al., 2025). These formulations collapse one of the classes into a single cluster, which is a restrictive assumption considering that both human-written and machine-generated text exhibit substantial heterogeneity across domains, prompts, and model families. To address this, we propose learning multiple data-driven prototypes for both classes, enabling explicit modeling of intra-class diversity on both sides rather than enforcing asymmetric or single-cluster assumptions.

2.2 Prototype-Based Representation Learning

Prototype-based methods represent each class by one or more representative points in an embedding space and classify inputs based on their similarity to these prototypes. A canonical example is Prototypical Networks, which learn an encoder and compute class prototypes as the mean embeddings of support examples, classifying queries via distances to these prototypes in a metric-learning framework (Snell et al., 2017). This view treats prototypes as data-driven summaries of class structure: training encourages instances of the same class to concentrate around their prototype while separating prototypes of different classes. Subsequent work has extended this idea beyond the original few-shot setting, exploring multiple prototypes per class, part- or attribute-level prototypes, and more flexible similarity functions in settings ranging from metric and meta-learning to interpretable models in vision and NLP (Chen et al., 2019; Hong et al., 2023).

Prototype-based representations have also been applied to detection problems, where prototypes are used to characterize an in-distribution manifold for out-of-distribution or anomaly detection, and to self-supervised or contrastive learning (Chen et al., 2024; Li et al., 2021). In many of these settings, prototypes either model a single normal distribution, or are primarily used to improve structure within a single class or task (Xie et al., 2023). Our problem requires modeling heterogeneous distributions on both sides of the detection problem, where neither human-written nor LLM-generated text can be adequately represented by a single global cluster. We therefore adopt a multi-prototype formulation and incorporate data-driven initialization and prototype update strategies to improve LLM-generated text detection.

3 Proposed Method

We propose **MD-ProTector**, a **M**ultiple **D**ata-driven **Pro**totype-based de**Tector** designed to capture the intra-class diversity of human-written and LLM-generated texts. This section describes how prototypes are initialized and updated (3.2), and how they are integrated into a unified training objective for lightweight LLM-generated text detection (3.3).

3.1 Problem Formulation and Encoder

Given an input space $\mathcal{X}$ of text sequences, each instance is associated with a label $y \in \{0, 1\}$ indicating whether the text is LLM-generated ($y = 0$) or human-written ($y = 1$). A lightweight transformer encoder f_θ is employed to map an input text x to a fixed-dimensional vector. To ensure directional alignment, the output is projected onto the unit hypersphere via L_2 normalization:

$$\mathbf{z} = \frac{f_\theta(x)}{\|f_\theta(x)\|_2} \in \mathbb{S}^{d-1}. \tag{1}$$

All subsequent operations are performed in this normalized embedding space.

3.2 Prototype Modeling and Dynamics

This subsection describes how class-specific prototypes are initialized and dynamically updated to track evolving representation within each class. A set of learnable prototypes $\mathcal{P}_c = \{\mathbf{p}_{c,1}, \ldots, \mathbf{p}_{c,R}\}$ is maintained for each class c. Here, $c = 0$ and $c = 1$ correspond to LLM-generated and human-written text respectively, and R denotes the number of prototype vectors assigned to each class.

Data-Driven Initialization. Prototypes are initialized using a data-driven approach to avoid poor local optima. Before training, we extract embeddings for the training set and perform K-Means algorithm per class (McQueen, 1967). The resulting centroids serve as the initial prototypes $\mathcal{P}_c^{(0)} = \{\mathbf{p}_{c,r}^{(0)}\}_{r=1}^{R}$.

EMA-based Prototype Updates. Each prototype $\mathbf{p}_{c,r}$ is updated using an Exponential Moving

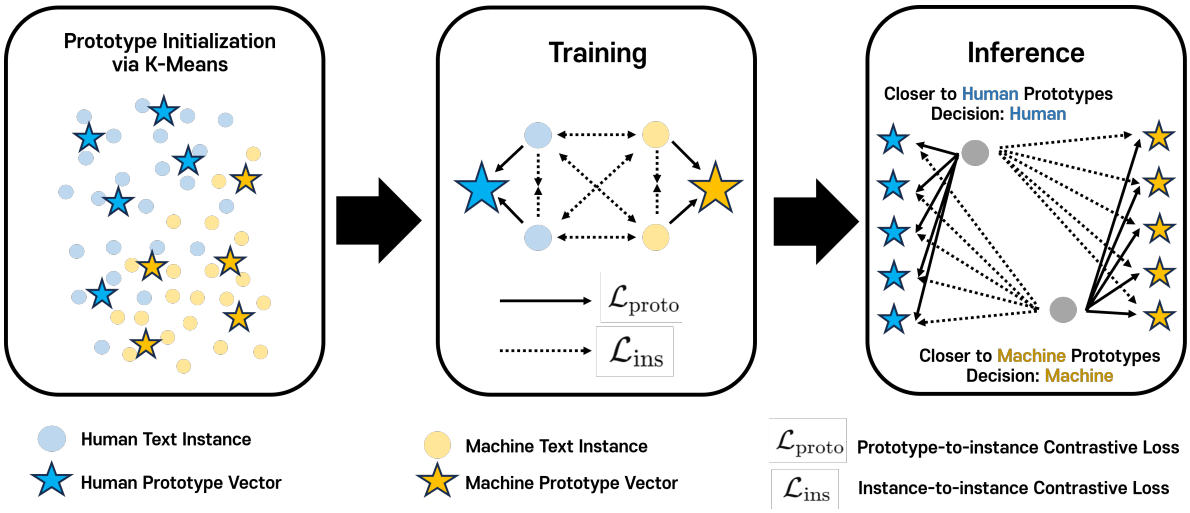

Figure 2: **Overview of MD-ProTector.** Our detector encodes input text into a normalized embedding space. MD-ProTector learns multiple prototypes (colored markers) for each class (Human/Machine) to capture diversity. The model is trained via a dual objective: aligning instances with class prototypes ($\mathcal{L}_{\text{proto}}$) while densifying local clusters ($\mathcal{L}_{\text{ins}}$).

Average (EMA) scheme to stably track the evolving embedding space during training (Zhan et al., 2020).

At each training step t, the update relies on $\bar{\mathbf{z}}_{c,r}^{(t)}$, which is the weighted average of embeddings of the current mini-batch with respect to their assignment to current prototype $\mathbf{p}_{c,r}$. Its exact computation is described in the following paragraph.

The prototype update is performed as

$$\mathbf{v}_{c,r}^{(t+1)} = \gamma \mathbf{p}_{c,r}^{(t)} + (1-\gamma)\bar{\mathbf{z}}_{c,r}^{(t)}, \tag{2}$$

where γ controls the momentum of the update. The resulting vector is projected back onto the unit hypersphere:

$$\mathbf{p}_{c,r}^{(t+1)} = \frac{\mathbf{v}_{c,r}^{(t+1)}}{\|\mathbf{v}_{c,r}^{(t+1)}\|_2}. \tag{3}$$

Soft Assignment. To determine how instances are associated with each prototype, we employ a soft assignment mechanism. Given an embedding $\mathbf{z}$ and the set of prototypes for class c, the assignment weight to prototype $\mathbf{p}_{c,r}$ is defined as

$$a_{c,r}(\mathbf{z}) = \frac{\exp(\mathbf{z}^\top \mathbf{p}_{c,r}/\kappa)}{\sum_{k=1}^{R} \exp(\mathbf{z}^\top \mathbf{p}_{c,k}/\kappa)}, \tag{4}$$

where κ is a parameter that controls the sharpness of the assignment. Using these weights, the prototype-specific embedding used in the EMA update is computed as

$$\bar{\mathbf{z}}_{c,r}^{(t)} = \frac{\sum_{i \in \mathcal{B}_c} a_{c,r}(\mathbf{z}_i)\, \mathbf{z}_i}{\sum_{i \in \mathcal{B}_c} a_{c,r}(\mathbf{z}_i) + \epsilon}, \tag{5}$$

where $\mathcal{B}_c$ denotes the subset of the mini-batch with label c and ϵ is a small constant for numerical stability. The same assignment weights are also used in the prototype-based contrastive objective described in the next subsection.

3.3 Training Objectives

This subsection introduces the training objectives designed to effectively utilize multiple prototypes, combining global alignment to class prototypes ($\mathcal{L}_{\text{proto}}$) with local consistency among peer instances ($\mathcal{L}_{\text{ins}}$).

Prototype-to-instance Contrastive Loss ($\mathcal{L}_{\text{proto}}$). This objective compares how strongly an instance embedding $\mathbf{z}_i$ aligns with prototypes of its ground-truth class c_i, relative to the full set of prototypes across both classes. Positives are defined as a soft mixture of class-conditional prototypes using the assignment weights $a_{c_i,r}(\mathbf{z}_i)$ introduced in the previous subsection, leading to the following contrastive formulation:

$$\mathcal{L}_{\text{proto}} = -\log \frac{\sum_{r=1}^{R} a_{c_i,r}(\mathbf{z}_i)\, \phi(\mathbf{z}_i, \mathbf{p}_{c_i,r})}{\sum_{\mathbf{p} \in \mathcal{P}} \phi(\mathbf{z}_i, \mathbf{p})}, \tag{6}$$

where $\phi(\mathbf{u}, \mathbf{v}) = \exp(\mathbf{u}^\top \mathbf{v}/\tau)$ and $\mathcal{P}$ denotes the set of all prototypes across classes.

Instance-to-instance Contrastive Loss ($\mathcal{L}_{\text{ins}}$). To further encourage local compactness within each class, we incorporate an instance-to-instance InfoNCE objective. Given a mini-batch $\mathcal{B}$, let $\mathcal{P}(i) = \{j \in \mathcal{B} \mid c_j = c_i\}$ denote the set of

indices sharing the same class label as instance i. The loss is defined as

$$\mathcal{L}_{\text{ins}} = -\sum_{i\in\mathcal{B}} \frac{1}{|\mathcal{P}(i)|} \sum_{j\in\mathcal{P}(i)} \log \frac{\phi(\mathbf{z}_i, \mathbf{z}_j)}{\sum_{k\in\mathcal{B}\setminus\{i\}} \phi(\mathbf{z}_i, \mathbf{z}_k)}. \tag{7}$$

This objective complements the global alignment enforced by $\mathcal{L}_{\text{proto}}$ by explicitly densifying local neighborhoods within each class. While $\mathcal{L}_{\text{proto}}$ anchors embeddings to class-level prototypes, $\mathcal{L}_{\text{ins}}$ promotes fine-grained compactness among peer instances.

Total Training Objective. The final objective function is a weighted sum of the global prototype alignment and local instance consistency:

$$\mathcal{L}_{\text{total}} = \alpha\mathcal{L}_{\text{proto}} + \beta\mathcal{L}_{\text{ins}}, \tag{8}$$

where α and β are hyperparameters controlling the relative contribution of each term. The encoder parameters θ are optimized to minimize $\mathcal{L}_{\text{total}}$, while the prototypes are updated separately via the EMA mechanism described in Section 3.2.

3.4 Inference

At inference time, classification is performed by comparing the proximity of an input embedding to the class-specific prototype sets. For a given embedding $\mathbf{z}$, we compute a soft-assignment-weighted average of squared distances to the prototypes of each class c:

$$D(\mathbf{z}, c) = \sum_{r=1}^{R} a_{c,r}(\mathbf{z}) \, \|\mathbf{z} - \mathbf{p}_{c,r}\|_2^2. \tag{9}$$

This quantity summarizes how closely the instance aligns with the collection of prototypes representing class c, while accounting for multiple class-internal sub-modes via the assignment weights.

The final score is obtained by taking the difference between the two class-specific distances:

$$S(\mathbf{z}) = D(\mathbf{z}, 0) - D(\mathbf{z}, 1), \tag{10}$$

where a larger value indicates that the input is closer, on average, to human-written prototypes than to LLM-generated ones. A binary prediction is produced by thresholding this score using a value δ selected on a validation set:

$$\hat{y} = \mathbb{I}(S(\mathbf{z}) > \delta). \tag{11}$$

Algorithm 1 Training MD-ProTector

Require: Training set $\mathcal{D}$, encoder f_θ, number of prototypes R
Ensure: Trained encoder f_θ and prototype sets $\{\mathcal{P}_c\}_{c\in\{0,1\}}$

```
Initialize class-wise prototype sets P_c^(0) using K-Means on encoder embeddings.
for each training step do
    Encode minibatch: z_i ← f_θ(x_i) and normalize.
    Compute L_proto and L_ins.
    Update encoder parameters θ by minimizing L_total.
    Update prototypes via EMA using soft assignments.
end for
```

Algorithm 2 Inference

Require: Input text x, encoder f_θ, prototype sets $\{\mathcal{P}_c\}_{c\in\{0,1\}}$, threshold δ

```
z ← f_θ(x) and normalize.
Compute class-wise distances D(z, 0) and D(z, 1).
Predict ŷ = 𝕀(D(z, 0) − D(z, 1) > δ).
return ŷ
```

4 Experiments

We first detail the experimental setup and MD-ProTector's main detection performance. We then analyze its robustness against distribution shifts and validate component contributions via ablation studies.

4.1 Experimental Setup

Dataset. We use the MAGE dataset (Li et al., 2024), a large-scale benchmark designed for realistic, open-domain LLM-generated text detection. The dataset consists of approximately 95k human-written texts drawn from diverse domains, including scientific writing and informal texts, alongside 236k LLM-generated samples produced by 27 large language models from seven model families, including GPT, LLaMA, OPT, and FLAN-T5. The LLM-generated texts are created under multiple prompting strategies—continuation, topical, and specified—resulting in substantial diversity in both content and generation patterns. This heterogeneous composition reflects real-world detection scenarios that involve substantial diversity in both domains and generator models.

Table 1: **Main Detection Performance.** Comparison with various baseline categories. **HumanRec** and **MachineRec** denote recall measured separately on human-written and LLM-generated texts, respectively. **AvgRec** denotes the average of Human and Machine Recall. MD-ProTector achieves the best AvgRec while minimizing the gap between HumanRec and MachineRec, suggesting that our method is unbiased and equally effective across both classes.

Method	AvgRec	HumanRec	MachineRec	Acc	F1
FastText†	78.80	86.34	71.26	78.89	80.53
GLTR†	55.42	12.42	**98.42**	54.92	21.79
DetectGPT†	60.48	86.92	34.05	60.79	69.16
Binary-Classifier‡	92.91	88.54	97.29	92.86	92.62
DeTeCtive‡	94.59	92.91	96.28	94.57	94.54
DeepSVDD‡	94.45	95.86	93.05	94.46	94.58
MD-ProTector (Ours)	**95.30**	**95.96**	94.63	**95.30**	**95.39**

Table 2: **Robustness Evaluation under Distribution Shifts.** Aggregated performance on (A) Unseen Domains and (B) Unseen Generator Models. DeepSVDD performs well on unseen models but shows lower Human Recall on unseen domains. **MD-ProTector** shows the highest Overall Average across the combined scenarios.

Method	AvgRec	HumanRec	MachineRec	Acc	F1
(A) Unseen Domains (Topic Shift)					
Binary-Classifier	67.03	35.06	**99.01**	66.49	47.93
DeTeCtive	76.73	55.95	97.50	76.33	67.49
DeepSVDD	86.56	87.59	85.52	86.46	85.52
MD-ProTector (Ours)	**87.59**	**91.32**	83.87	**87.67**	**88.34**
(B) Unseen Models (Generator Shift)					
Binary-Classifier	89.19	86.56	91.83	89.19	89.00
DeTeCtive	87.83	83.06	**92.60**	**91.11**	91.25
DeepSVDD	**91.08**	91.34	90.82	90.92	**91.29**
MD-ProTector (Ours)	90.51	**92.40**	88.62	90.52	90.77
Overall (Average of (A) & (B))					
Binary-Classifier	78.11	60.81	**95.42**	77.84	68.46
DeTeCtive	82.28	69.50	95.05	83.72	79.37
DeepSVDD	88.82	89.47	88.17	88.69	88.41
MD-ProTector (Ours)	**89.05**	**91.86**	86.24	**89.10**	**89.55**

Baselines. We compare MD-ProTector with a range of representative LLM-generated text detectors. These include FastText (Joulin et al., 2017) as a lightweight lexical baseline, statistical methods such as GLTR (Gehrmann et al., 2019) and DetectGPT (Mitchell et al., 2023), and several encoder-based detectors built on pretrained language models. For encoder-based methods, we include a standard Binary Classifier trained with a cross-entropy loss, as well as DeTeCtive (Guo et al., 2024) and DeepSVDD (Zeng et al., 2025).

Implementation Details. We adopt 125M Unsupervised SimCSE-RoBERTa (Gao et al., 2021) as the backbone encoder f_θ for MD-ProTector, the Binary Classifier, and all other encoder-based baselines, following the implementation choices used in DeTeCtive and DeepSVDD to ensure a fair comparison. The number of prototypes per class is set to $R = 16$ and initialized via K-Means clustering to facilitate warm-start optimization. Models are trained for 30 epochs with a batch size of 32 using AdamW with a learning rate of 2×10^{-5} and 2000 warmup steps. The temperature parameter is set to $\tau = 0.1$, and the loss weights are $\alpha = 1.0$ and $\beta = 1.0$. The EMA momentum is set to $\gamma = 0.9$. $\kappa = 0.07$ for soft assignments. All experiments are conducted using two NVIDIA RTX 3090 GPUs.

4.2 Main Detection Performance

[0†]Results for FastText, GLTR, and DetectGPT are retrieved from the MAGE benchmark (Li et al., 2024).

[0‡]Binary-Classifier, DeTeCtive and DeepSVDD are trained under the same experimental environment as MD-

Table 1 reports the main detection performance across all methods. Clear differences emerge not only in overall accuracy but also in class-wise behavior, highlighting how different modeling assumptions affect detection outcomes.

FastText achieves only moderate performance, reflecting its reliance on surface-level lexical cues. GLTR attains very high Machine Recall (98.42%) but fails to detect human-written text (Human Recall 12.42%), while DetectGPT shows the opposite pattern with high Human Recall (86.92%) and low Machine Recall (34.05%). These results indicate that methods relying on generation statistics tend to over-specialize in a single class, leading to unstable detection behavior.

The standard binary classifier also exhibits a clear bias toward machine class. Although it attains a high Machine Recall of 97.29%, its Human Recall drops to 88.54%, resulting in an AvgRec of 92.91%. This pattern indicates a strong bias toward predicting LLM-generated text and suggests that binary supervision collapses the diverse human-written texts into an insufficiently expressive representation, ultimately limiting overall detection performance.

Introducing additional structure into encoder-based detectors partially alleviates this issue. DeTeCtive and DeepSVDD both improve AvgRec compared to the Binary Classifier, reaching 94.59% and 94.45%, respectively. However, they exhibit different bias patterns. DeTeCtive favors LLM-generated text, achieving a Machine Recall of 96.28% at the expense of Human Recall (92.91%), while DeepSVDD shows the opposite tendency, with strong Human Recall (95.86%) but reduced sensitivity to LLM-generated text (93.05%). While these approaches mitigate the collapse observed in the standard binary formulation, they still fall short of achieving the best overall performance.

MD-ProTector achieves the strongest results among all evaluated methods. It records the highest AvgRec of **95.30%** and F1-score of **95.39%**, while maintaining consistently high recall for both human-written (95.96%) and LLM-generated text (94.63%), resulting in the least class-wise gap ($\Delta = 1.36\%$). These results demonstrate that explicitly modeling internal diversity for both classes enables MD-ProTector to overcome the limitations of standard binary supervision and prior structured baselines, leading to superior and more stable detection performance.

ProTector.

4.3 Robustness Evaluation

We evaluate the robustness of MD-ProTector under distribution shifts induced by unseen domains and unseen generator models. In the *Unseen Domains* setting, the detector is trained by excluding all samples from one domain and evaluated on that held-out domain, repeating this process across 10 distinct domains. In the *Unseen Models* setting, texts generated by one generator model are entirely excluded from training and used only for evaluation, with this procedure repeated over 7 distinct generator models. For each held-out domain or model, a separate detector is trained to ensure strictly unseen evaluation. Table 2 reports the performance aggregated over all runs, providing a comprehensive assessment of robustness to domain and generator shifts.

Under out-of-distribution conditions, the standard Binary Classifier exhibits severe performance degradation, particularly in the Unseen Domain setting. While Machine Recall remains high, Human Recall drops sharply, indicating that the detector has a strong bias to predict an instance as LLM-generated. This collapse mirrors the limitations observed in the main detection results and shows that the weaknesses of binary classification are further amplified under domain shifts. Succeeding two encoder-based baselines alleviate this behavior to some extent but remain sensitive to semantic diversity across domains.

In contrast, MD-ProTector demonstrates clear advantages under domain shifts, achieving the highest Average Recall and F1-score while maintaining substantially higher Human Recall than competing methods. Across the combined out-of-distribution scenarios, MD-ProTector also attains the best overall performance. These results indicate that explicitly modeling internal diversity within both classes enables more stable detection under distribution shifts, supporting the effectiveness of the proposed multi-prototype framework in realistic settings.

4.4 Ablation Studies

To validate the design choices of MD-ProTector, we analyze the impact of training objectives, prototype counts, initialization strategies, and hyperparameter sensitivity.

Synergy of Dual Objectives. Table 3 presents the performance across different combinations of

loss functions. Using $\mathcal{L}_{\text{ins}}$ alone yields 94.12%, while $\mathcal{L}_{\text{proto}}$ alone also achieves 94.09%. The best performance (95.30%) is obtained by combining both objectives. This suggests a complementary effect where $\mathcal{L}_{\text{proto}}$ establishes global separation between human and machine manifolds, while $\mathcal{L}_{\text{ins}}$ improves local density within sub-clusters.

Table 3: **Impact of Training Objectives.**

$\mathcal{L}_{\text{proto}}$	$\mathcal{L}_{\text{ins}}$	**AvgRec (%)**
✓	–	94.12
–	✓	94.09
✓	✓	**95.30**

Impact of Multi-Prototype Modeling. Table 4 examines how the number of prototypes per class influences detection performance. Introducing multiple prototypes consistently improves performance up to a moderate scale, with the best result obtained at $R = 16$. This trend suggests that explicitly modeling intra-class heterogeneity enables the detector to better capture diverse writing styles and generator artifacts. When the number of prototypes is further increased to $R = 32$, performance degrades, indicating that overly fine-grained partitioning can fragment the representation space and weaken class-level discrimination.

Table 4: **Effect of Prototype Count (R).**

# Prototypes (R)	**AvgRec (%)**
1	94.49
2	94.31
4	94.10
8	94.41
16	**95.30**
32	94.06

Effect of K-Means Initialization Strategy. Prototype initialization via K-Means outperforms random initialization by 1.10% (Table 5). K-Means provides a data-driven starting point by anchoring prototypes to the centroids of initial distributions, ensuring more stable convergence compared to random placement.

Table 5: **Effect of Initialization.**

Initialization Method	**AvgRec (%)**
Random Initialization	94.20
K-Means (Ours)	**95.30**

Table 6: **Sensitivity to Temperature (τ).**

Temperature (τ)	**AvgRec (%)**
0.05	94.75
0.1	**95.30**
0.2	93.90
0.5	94.00

Table 7: **Sensitivity to Momentum (γ).**

Momentum (γ)	**AvgRec (%)**
0.5	94.88
0.9	**95.30**
0.95	94.75
0.99	94.89

Hyperparameter Sensitivity. We further investigate the sensitivity of temperature τ and momentum γ for EMA updates. As shown in Table 6, the optimal temperature is found at $\tau = 0.1$. Higher values like $\tau = 0.5$ lead to performance degradation (94.00%), likely due to the over-smoothing of the contrastive distribution. For prototype momentum (Table 7), $\gamma = 0.9$ yields the highest AvgRec. Lower momentum ($\gamma = 0.5$) results in unstable prototype updates, while excessively high momentum ($\gamma = 0.99$) slows down the adaptation of prototypes to the evolving embedding space.

5 Conclusion

In this paper, we presented MD-ProTector, a prototype-based detection framework that explicitly models intra-class diversity in both human-written and LLM-generated text. By representing each class with multiple data-driven prototypes, our method avoids collapsing heterogeneous texts into a single global cluster. MD-ProTector jointly designs data-driven prototype initialization, dynamic prototype updates, and prototype-aware training objectives. A prototype-to-instance contrastive loss aligns text representations with the same class prototypes, with complementary instance-to-instance contrastive loss. Together, these components enable more balanced and discriminative representation learning for LLM-generated text detection. Extensive experiments show that MD-ProTector achieves strong detection performance and maintains robustness under distribution shifts from unseen domains and generator models, while preserving the efficiency of lightweight encoder-based detectors.

Limitations

This work assumes a fixed-label binary detection setting in which each input is classified as either human-written or LLM-generated, and does not address more complex scenarios such as partial generation, human–machine co-editing, or estimating degrees of machine involvement. In addition, the number of prototypes per class is treated as an empirically fixed design choice. Adaptive mechanisms for adjusting prototype cardinality based on data characteristics are not explored. Finally, our analysis focuses on prototype-based modeling and training strategies within a fixed encoder configuration, rather than on optimizing or comparing backbone architectures, as the proposed framework is intended to be generally applicable across lightweight text encoders.

Ethics Statement

The datasets utilized do not include private data or non-public personally identifiable information. The development of reliable text detection systems is crucial for maintaining trust in digital information. However, we acknowledge that detection technologies can potentially be used in a dual-use manner—adversaries might use our detector as a discriminator to train more sophisticated generators that evade detection. Additionally, while we strove to use diverse datasets, the "Human" class in our training data is sourced from web texts (e.g., Reddit, Wikipedia), which may contain inherent biases. Users should be cautious when deploying this model in sensitive contexts, as false positives could unfairly penalize human writers.

References

Anton Bakhtin, Sam Gross, Myle Ott, Yuntian Deng, Marc'Aurelio Ranzato, and Arthur Szlam. 2019. Real or fake? learning to discriminate machine from human generated text. *arXiv preprint arXiv:1906.03351*.

Guangsheng Bao, Yanbin Zhao, Zhiyang Teng, Linyi Yang, and Yue Zhang. 2024. Fast-detectGPT: Efficient zero-shot detection of machine-generated text via conditional probability curvature. In *The Twelfth International Conference on Learning Representations*, pages 24814–24836, Vienna, Austria.

Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. 2019. This looks like that: Deep learning for interpretable image recognition. In *Advances in Neural Information Processing Systems*, volume 32, pages 8928–8939, Vancouver, BC, Canada. Curran Associates, Inc.

Jun-Kun Chen, Jilin Mei, Liang Chen, Fangzhou Zhao, and Yu Hu. 2024. Proto-OOD: Enhancing OOD object detection with prototype feature similarity. *arXiv preprint arXiv:2409.05466*.

Miranda Christ, Sam Gunn, and Or Zamir. 2024. Undetectable watermarks for language models. In *The Thirty Seventh Annual Conference on Learning Theory*, pages 1125–1139, Edmonton, Canada. Proceedings of Machine Learning Research.

Yin Cui, Feng Zhou, Yuanqing Lin, and Serge J. Belongie. 2016. Fine-grained categorization and dataset bootstrapping using deep metric learning with humans in the loop. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016)*, pages 1153–1162, Las Vegas, NV, USA. IEEE Computer Society.

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

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander Rush. 2019. GLTR: Statistical detection and visualization of generated text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 111–116, Florence, Italy. Association for Computational Linguistics.

Xun Guo, Shan Zhang, Yongxin He, Ting Zhang, Wanquan Feng, Haibin Huang, and Chongyang Ma. 2024. DeTeCtive: Detecting ai-generated text via multi-level contrastive learning. In *Advances in Neural Information Processing Systems*, volume 37, pages 88320–88347. Curran Associates, Inc.

Dat Hong, Tong Wang, and Stephen Baek. 2023. ProtoryNet-interpretable text classification via prototype trajectories. *Journal of Machine Learning Research*, 24(264):1–39.

Daphne Ippolito, Daniel Duckworth, Chris Callison-Burch, and Douglas Eck. 2020. Automatic detection of generated text is easiest when humans are fooled. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1808–1822, Online. Association for Computational Linguistics.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain. Association for Computational Linguistics.

Hemanth Kandula, Chak Fai Li, Haoling Qiu, Damianos Karakos, Hieu Man, Thien Huu Nguyen, and Brian Ulicny. 2025. BBN-U.Oregon's ALERT system at GenAI content detection task 3: Robust authorship style representations for cross-domain machine-generated text detection. In *Proceedings of the 1st Workshop on GenAI Content Detection (GenAIDetect)*, pages 358–364, Abu Dhabi, UAE. International Conference on Computational Linguistics.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A watermark for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pages 17061–17084, Vienna, Austria. Proceedings of Machine Learning Research.

Kristian Kuznetsov, Eduard Tulchinskii, Laida Kushnareva, German Magai, Serguei Barannikov, Sergey Nikolenko, and Irina Piontkovskaya. 2024. Robust AI-generated text detection by restricted embeddings. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 17036–17055, Miami, Florida, USA. Association for Computational Linguistics.

Soonchan Kwon and Beakcheol Jang. 2025. A comprehensive survey of fake text detection on misinformation and lm-generated texts. *IEEE Access*, 13:25301–25324.

Junnan Li, Pan Zhou, Caiming Xiong, and Steven Hoi. 2021. Prototypical contrastive learning of unsupervised representations. In *9th International Conference on Learning Representations*, pages 5576–5591, Virtual Event, Austria.

Yafu Li, Qintong Li, Leyang Cui, Wei Bi, Zhilin Wang, Longyue Wang, Linyi Yang, Shuming Shi, and Yue Zhang. 2024. MAGE: Machine-generated text detection in the wild. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 36–53, Bangkok, Thailand. Association for Computational Linguistics.

Zhuang Liu, Wayne Lin, Ya Shi, and Jun Zhao. 2021. A robustly optimized BERT pre-training approach with post-training. In *Chinese Computational Linguistics - 20th China National Conference, CCL 2021*, volume 12869, pages 471–484, Hohhot, China. Springer.

James B McQueen. 1967. Some methods of classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. 2023. DetectGPT: Zero-shot machine-generated text detection using probability curvature. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pages 24950–24962, Honolulu, Hawaii. Proceedings of Machine Learning Research.

Ayat A. Najjar, Huthaifa I. Ashqar, Omar A. Darwish, and Eman M. Hammad. 2025. Detecting ai-generated text in educational content: Leveraging machine learning and explainable AI for academic integrity. *arXiv preprint arXiv:2501.03203*.

Juan Diego Rodriguez, Todd Hay, David Gros, Zain Shamsi, and Ravi Srinivasan. 2022. Cross-domain detection of GPT-2-generated technical text. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1213–1233, Seattle, United States. Association for Computational Linguistics.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, volume 30, pages 4077–4087, Long Beach, CA, USA. Curran Associates, Inc.

Adaku Uchendu, Thai Le, Kai Shu, and Dongwon Lee. 2020. Authorship attribution for neural text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8384–8395, Online. Association for Computational Linguistics.

Zecong Wang, Jiaxi Cheng, Chen Cui, and Chenhao Yu. 2023. Implementing BERT and fine-tuned roberta to detect AI generated news by chatgpt. *arXiv preprint arXiv:2306.07401*.

Junchao Wu, Shu Yang, Runzhe Zhan, Yulin Yuan, Lidia Sam Chao, and Derek Fai Wong. 2025. A survey on LLM-generated text detection: Necessity, methods, and future directions. *Computational Linguistics*, 51(1):275–338.

Junchao Wu, Runzhe Zhan, Derek F. Wong, Shu Yang, Xinyi Yang, Yulin Yuan, and Lidia S. Chao. 2024. DetectRL: Benchmarking llm-generated text detection in real-world scenarios. In *Advances in Neural Information Processing Systems*, volume 37, pages 100369–100401, Vancouver, BC, Canada. Curran Associates, Inc.

Sean Xie, Soroush Vosoughi, and Saeed Hassanpour. 2023. Proto-lm: A prototypical network-based framework for built-in interpretability in large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3964–3979, Singapore. Association for Computational Linguistics.

Cong Zeng, Shengkun Tang, Yuanzhou Chen, Zhiqiang Shen, Wenchao Yu, Xujiang Zhao, Haifeng Chen, Wei Cheng, and zhiqiang xu. 2025. Human texts are

outliers: Detecting LLM-generated texts via out-of-distribution detection. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, San Diego, California, USA.

Xiaohang Zhan, Jiahao Xie, Ziwei Liu, Yew-Soon Ong, and Chen Change Loy. 2020. Online deep clustering for unsupervised representation learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2020)*, pages 6687–6696, Seattle, WA, USA. IEEE Computer Society.

A Detailed OOD Performance

In Section 4.3, we reported the aggregated metrics for Out-of-Distribution (OOD) scenarios. To provide a granular view of the robustness characteristics of each method, we present the detailed breakdown of Average Recall (AvgRec), Human Recall, Machine Recall, Accuracy, and F1-Score for all 10 unseen domains and 7 unseen generator models.

Table 8: **Detailed Metrics: Binary-Classifier.** Human Recall degrades significantly on unseen domains.

Scenario	AvgRec	HumanRec	MachineRec	Acc	F1
(A) Unseen Domains					
CMV	83.78	68.91	98.65	84.12	80.92
ELI5	82.69	66.86	98.53	82.79	79.43
HellaSwag	62.29	27.10	97.48	60.97	41.87
ROC	53.06	6.47	99.65	52.43	12.12
Scigen	57.65	16.31	98.98	55.17	27.83
SQuAD	70.73	41.87	99.60	70.66	58.86
TLDR	61.68	24.14	99.22	60.98	38.66
WP	75.11	51.05	99.17	75.26	67.22
XSum	54.25	9.29	99.20	54.05	16.88
Yelp	69.07	38.57	99.57	68.46	55.50
Avg.	67.03	35.06	99.01	66.49	47.93
(B) Unseen Models					
Bloom-7B	89.36	83.92	94.80	89.36	88.75
FLAN-T5-Small	84.54	86.57	82.51	84.54	84.85
GLM-130B	91.51	85.85	97.17	91.51	91.00
GPT-J	92.55	85.30	99.79	92.55	91.96
GPT-3.5-Turbo	85.63	93.48	77.78	85.63	86.68
LLaMA-2-7B	90.63	85.77	95.50	90.63	90.15
OPT-125M	90.13	85.01	95.24	90.13	89.59
Avg.	89.19	86.56	91.83	89.19	89.00

Table 9: **Detailed Metrics: DeTeCtive.** Full breakdown of performance across 17 OOD scenarios.

Scenario	AvgRec	HumanRec	MachineRec	Acc	F1
(A) Unseen Domains					
CMV	92.66	89.01	96.30	92.74	92.30
ELI5	87.86	78.33	97.40	87.92	86.57
HellaSwag	69.87	45.17	94.57	68.95	60.14
ROC	58.24	17.01	99.47	57.68	28.94
Scigen	73.14	48.58	97.69	71.66	64.50
SQuAD	86.04	74.44	97.64	86.01	84.21
TLDR	69.18	40.24	98.12	68.64	56.65
WP	92.80	87.38	98.21	92.83	92.38
XSum	61.46	25.83	97.08	61.30	40.13
Yelp	76.01	53.47	98.55	75.55	69.05
Avg.	76.73	55.95	97.50	76.33	67.49
(B) Unseen Models					
Bloom-7B	93.35	90.79	95.92	93.35	93.18
FLAN-T5-Small	83.80	91.52	76.07	83.80	84.96
GLM-130B	93.04	89.55	96.52	93.04	92.78
GPT-J	95.32	90.92	99.72	95.32	95.10
GPT-3.5-Turbo	87.28	92.83	81.72	87.28	87.95
LLaMA-2-7B	92.55	91.56	93.53	92.55	92.47
OPT-125M	92.40	90.94	93.86	92.40	92.29
Avg.	87.83	83.06	92.60	91.11	91.25

Table 10: **Detailed Metrics: DeepSVDD.** High Machine Recall is observed in Part (B), but Human Recall drops significantly in Part (A).

Scenario	AvgRec	HumanRec	MachineRec	Acc	F1
(A) Unseen Domains					
CMV	90.91	93.05	88.78	90.43	88.25
ELI5	88.44	86.33	90.55	88.53	87.84
HellaSwag	84.54	89.51	79.57	84.57	85.37
ROC	79.61	81.03	78.19	79.56	79.23
Scigen	84.48	86.78	82.19	84.25	83.23
SQuAD	88.67	84.63	92.71	89.55	86.35
TLDR	84.62	89.18	80.06	83.74	81.58
WP	93.21	93.65	92.76	93.20	93.09
XSum	84.06	88.30	79.84	83.81	83.68
Yelp	87.01	83.48	90.54	86.99	86.58
Avg.	86.56	87.59	85.52	86.46	85.52
(B) Unseen Models					
Bloom-7B	93.33	94.95	91.72	93.42	93.82
FLAN-T5-Small	85.47	85.97	84.96	85.47	85.68
GLM-130B	97.88	95.76	100.00	96.19	97.83
GPT-J	98.89	98.27	99.50	98.63	99.02
GPT-3.5-Turbo	85.35	91.40	79.30	85.84	87.47
LLaMA-2-7B	93.35	93.15	93.55	93.35	93.54
OPT-125M	91.92	91.51	92.32	91.90	92.18
Avg.	91.08	91.34	90.82	90.92	91.29

Table 11: **Detailed Metrics: MD-ProTector (Ours).** Consistent and balanced performance across all OOD scenarios.

Scenario	AvgRec	HumanRec	MachineRec	Acc	F1
(A) Unseen Domains					
CMV	94.67	94.76	94.59	94.67	94.56
ELI5	89.90	90.75	89.04	89.89	89.92
HellaSwag	83.06	94.78	71.34	83.49	85.62
ROC	82.79	89.56	76.02	82.88	84.14
Scigen	85.95	87.71	84.19	86.05	86.95
SQuAD	88.61	86.68	90.54	88.61	88.41
TLDR	88.11	89.98	86.24	88.15	88.55
WP	92.29	95.80	88.78	92.27	92.49
XSum	83.31	93.30	73.33	83.36	84.92
Yelp	87.25	89.90	84.62	87.31	87.84
Avg.	87.59	91.32	83.87	87.67	88.34
(B) Unseen Models					
Bloom-7B	92.09	95.03	89.15	92.09	92.32
FLAN-T5-Small	82.79	84.16	81.42	82.79	83.02
GLM-130B	94.77	96.96	92.59	94.78	94.89
GPT-J	98.13	98.20	98.06	98.13	98.13
GPT-3.5-Turbo	84.90	86.72	83.08	84.90	85.17
LLaMA-2-7B	93.26	94.39	92.13	93.26	93.34
OPT-125M	93.78	94.93	92.63	93.78	93.85
Avg.	90.51	92.40	88.62	90.52	90.77