Multi-scale Feature Fusion-based Dynamic Framework using Continual Learning to Identify Text Generated by Multiple Large Language Models

Muhammad Sohail[a], Zan Hongying[a,*], Muhammad Abdullah[a], Fabio Caraffini[b,*], Arifa Javed[a], Hassan Eshkiki[b]

[a] *School of Computer Science and Artificial Intelligence, Zhengzhou University, Zhengzhou, 450001, Henan, China*
[b] *Department of Computer Science, Swansea University, Swansea SA1 8EN, UK*

Abstract

The rapid advancement of large language models has significantly enhanced the quality of AI-generated text, making it increasingly difficult for detection systems to distinguish from human-written content. Existing detection methods, such as statistical, linguistic, machine learning, and deep learning approaches, often exhibit a decline in performance when applied to new or previously unseen large language models. Additionally, they tend to become outdated due to their static frameworks and inability to adapt to emerging patterns in generative text. To address this limitation, we introduce a novel dynamic fusion framework that integrates multi-scale feature fusion to capture diverse text patterns and employs continual learning with Elastic Weight Consolidation (EWC) to adapt to new models while mitigating catastrophic forgetting. This is the first attempt, to the best of our knowledge, to develop such a dynamic framework for AI-generated text detection. Evaluated on the TuringBench and Deepfake-TextDetect benchmark datasets, our framework achieves an average accuracy of 95.78% and 92.39%, outperforming the standard model by 5.88% and 7.98%, respectively, in distinguishing AI-generated from human-written text across var-

*Corresponding author
Email addresses: sohailm@gs.zzu.edu.cn (Muhammad Sohail), iehyzan@zzu.edu.cn (Zan Hongying), abdullah@gs.zzu.edu.cn (Muhammad Abdullah), fabio.caraffini@swansea.ac.uk (Fabio Caraffini), arifa.javed@gs.zzu.edu.cn (Arifa Javed), h.g.eshkiki@swansea.ac.uk (Hassan Eshkiki)

ious language generative model architectures. The continual learning ensures that the model remains adaptive and accurate over time, which is essential for practical applications in dynamic environments. This dynamic and adaptive approach paves the way for resilient AI-generated text detection systems capable of evolving alongside the rapidly advancing landscape of generative language technologies.

Keywords: Natural Language Processing, Generative Artificial Intelligence, Large Language Models, AI-generated Text Detection, Continual Learning, Elastic Weight Consolidation

1. Introduction

In recent years, the rapid advancement of generative Artificial Intelligence (AI), particularly with the emergence of language models such as Bidirectional Encoder Representations from Transformers (BERT) [18] and subsequently 'powerful' large language models (LLMs) like ChatGPT [2], T5 [59], and LLaMA [70], has revolutionised natural language processing (NLP). These models generate highly coherent and contextually rich text, enabling applications across industries like content creation, education, and customer service. Although these capabilities offer numerous benefits and are reshaping various industries with unprecedented efficiency, they also introduce significant risks, particularly in the form of misinformation, plagiarism, automated content manipulation [66, 37, 52, 36], and the skew of public opinion and discourse [10]. As LLMs become more sophisticated, distinguishing between AI-generated and human-written text has become a pressing concern across academic, media, and digital communication platforms. Thus, the development of models to detect AI-generated text is not only a technical necessity but also a critical safeguard to maintain the integrity of information.

However, identifying AI-generated text is a complex task, particularly due to the diversity and evolving nature of language generation models. Traditional detection methods, ranging from statistical [29] and linguistic approaches [40]

to deep learning models [4, 42], are static (which do not evolve for the new introduced LLMs) and often become outdated as new LLMs emerge. These methods, designed for specific models, fail to adapt to the unique patterns of newer models, leading to a decline in detection performance over time. For example, static detectors trained on earlier models like BERT may fail to identify outputs from newer, more sophisticated models like ChatGPT, highlighting the need the need for dynamic and adaptive detection systems that evolve alongside generative technologies [53].

To address these limitations, continual learning offers a promising solution by allowing models to learn incrementally from new data without forgetting previously acquired knowledge [64]. This is particularly crucial in AI-generated text detection, where models must evolve to keep pace with the continuous advancements in Large Language Models (LLMs). As newer LLMs introduce increasingly sophisticated and diverse patterns of AI-generated text, models need to retain past knowledge while adapting to these novel forms of text.

Effective continual learning for AI-generated text detection requires feature representations that are robust and adaptable for handling new LLMs. These models can change text statistics, leading to feature drift [54, 8]. Hence, if the feature extractor of the previously used model is overly specialised, adaptation to these new text statistical characteristics is challenging [57]. It is then essential to develop a feature representation that is stable yet flexible, capable of capturing both universal features common across AI-generated text and model-specific features unique to each LLM [17, 77]. This makes Multi-scale feature fusion crucial, as it combines features from various linguistic analysis levels. It integrates detailed local features, like token-level anomalies, with broader global patterns, such as semantic coherence. This process allows the model to develop a more robust and adaptable representation [30, 44].

Multi-scale feature fusion in continual learning is motivated by the non-stationary nature of AI-generated text [74]. Unlike human-written content, which often maintains consistent themes and structures, text produced by autoregressive LLMs exhibits significant variation in its statistical properties across

different segments [24, 31]. These variations create a temporal signature, where the likelihood of a token being machine-generated is not uniform but varies depending on its position and context within the document [69]. Multi-scale feature fusion captures these temporal dynamics by analysing text at different granularities, from local token-level anomalies to higher-level discourse structures [80]. This enables the model to track the evolution of text patterns over time, providing a more nuanced and accurate detection of AI-generated content, and by capturing both local and global patterns, multi-scale feature fusion enhances the model's ability to detect temporal changes in AI-generated text [79]. Local patterns help identify model-specific features like token co-occurrences or syntactic errors, while global patterns reflect structures such as semantic coherence and discourse flow [19, 45], and this dual approach allows the model to adapt to new LLMs while maintaining stability across different models [39]. Multi-scale feature fusion with continual learning allows the detection system to recognise new LLM signatures while retaining older model traits [60, 75]. This synergy helps the model evolve with generative technologies, ensuring adaptability and future-proofing for AI-generated text detection.

In this work, we propose a novel framework for detecting AI-generated text (AIGT), termed the dynamic fusion method, which integrates multi-scale (local and global) feature fusion using convolutional neural networks (CNNs) [41], Bidirectional Long Short-Term Memory Networks (BiLSTM) [26], and local & global attention mechanisms, with continual learning through Elastic Weight Consolidation (EWC) [39]. This framework enables the model to adapt to new LLMs without the need for retraining from scratch, allowing it to continuously improve its performance. Our system is designed to learn from diverse AI-generated text sources, improving its ability to generalise across both current and emerging language models.

Extensive experiments conducted on two publicly available benchmark datasets demonstrate that the proposed method outperforms existing approaches. For example, on the `TuringBench` dataset, the proposed dynamic fusion framework achieves an accuracy of 95.78%, surpassing the OpenAI detector by 17.05%.

Similarly, on the `DeepfakeTextDetect` dataset, the proposed method reaches an accuracy of 92.39%, outperforming the recent DetectGPT by 31.91%. Additionally, we investigate the effectiveness of continual learning by comparing the incorporation of Elastic Weight Consolidation (EWC) with a version of the framework without EWC, highlighting its adaptability across emerging LLMs. The experiments further demonstrate the framework's stability as AI models progressively advance, suggesting its resilience and will not become outdated as generative models evolve. Overall, the results highlight the robustness of the proposed method in effectively addressing the challenges posed by the growing number of large language models.

The main contributions of this study are as follows:

- We present a novel dynamic framework using multi-scale feature fusion to effectively detect AI-generated text by capturing diverse text patterns and complex linguistic structures in various large language models.
- We strategically integrate Elastic Weight Consolidation (EWC) into the proposed framework to enable continual learning and mitigate catastrophic forgetting while ensuring adaptability to emerging large language models.
- We show that our approach outperforms several non-continual and state-of-the-art models in distinguishing AI-generated from human-written text across two established datasets.

The article is organised as follows: Section 2 reviews related work. Section 3 describes in details the entire methodology of this investigation. Section 4 discusses the experimental results. Finally, Section 5 concludes this work and identifies future work and research directions.

2. Related Work

This section reviews the evolution of detection methodologies, highlighting their strengths and limitations, and positioning our proposed approach within

this landscape.

2.1. Statistical and Linguistic Feature-Based Methods

Early detection techniques initially depended on statistical analysis of linguistic characteristics. For example, the studies in [16, 35] used metrics such as sentence length, frequency of function words, and word pair distributions. The research cited in [6, 5] identified syntactic irregularities such as the "phrase salad" phenomenon. While these methods were effective for earlier and simpler models, they are not well suited to modern LLMs due to their reliance on static, manually crafted features.

2.2. N-gram and Repetition-Based Approaches

More recent studies have explored statistical irregularities in LLM outputs. Those in [29] proposed methods based on high-order n-gram repetition and BScore metrics. Hamed and Wu [29] demonstrated that only 23% of the bigrams in ChatGPT-generated texts were unique, allowing high-accuracy detection. However, these approaches are often model-specific and struggle to generalise across diverse LLMs.

However, our empirical observations reveal a key limitation in using linguistic feature statistics: these methods rely heavily on extensive corpus statistics and are often tailored to specific types of LLMs. This dependency can limit their generalisability and reduce their effectiveness in detecting text from newer or less studied models.

2.3. Machine Learning (ML) with Stylistic Features

Classifiers trained in stylistic characteristics, such as syllable count, punctuation usage, and sentence structure, have shown promise [63, 65]. In [40] study, a novel algorithm was introduced that uses stylometric signals to detect when AI begins to generate tweets within a timeline. Similarly, in [55], the authors proposed StyloAI, a data-driven model that employs 31 stylometric features and a Random Forest classifier to distinguish AI-generated texts across multi-domain

datasets, achieving high identification accuracy. Although interpretable and effective in some contexts, these methods are vulnerable to stylistic mimicry by advanced LLMs, limiting their robustness in detecting misinformation or deceptive content.

2.4. Deep Learning and Fine-Tuned Language Models

There is a substantial amount of research [7, 1, 71, 4, 42] that investigates how well fine-tuned language models can detect text generated by LLMs. In particular, studies in 2019 recognized that fine-tuning transformer-based models such as RoBERTa [46] serves as a strong baseline for this task. These models can deliver optimal classification results across different encoding setups [20, 23]. The OpenAI detector [58] later adopted a similar fine-tuning approach. Recent studies [27, 49, 50, 14, 76] further validate the superior performance of the fine-tuned BERT family, which outperforms linguistic feature statistics, stylistic features, and machine learning classifier methods, demonstrating resilience to various attack techniques within domain settings.

Nevertheless, encoder-based fine-tuning approaches often lack robustness [7, 71, 4, 42]. These models tend to overfit to their training data or the source model's distribution, leading to a decline in performance when tested on unseen text from new AI models. Furthermore, fine-tuning LM classifiers faces limitations when dealing with text generated by different models [62].

PLM-based detectors often show a decline in performance when identifying AI-generated text from new or previously unseen LLMs, except in studies like [15]. The superior performance reported by the PLM-based detector in this study can be interpreted as a result emerging from the narrow test dataset employed. Using a larger dataset could easily change this trend and reveal a deterioration in performance. On the other hand, detectors trained on a specific LLM often struggle to generalize to others, even in controlled domain settings. For instance, in [28], a RoBERTa detector trained on GPT-3.5 responses in the HC3 corpus achieved an impressive F1 score of 99.82%. However, when tested on variants of ChatGPT using the same prompts and domain settings, the per-

formance dropped by up to 19%. This decrease was attributed to variations in perplexity across models, rather than content or domain shifts. Similarly, in [78], a RoBERTa-Base detector trained on GPT-3.5-turbo summaries from the *DetectRL* benchmark achieved an AUROC of 99%, but saw a significant decline to 55% when tested on Claude-instant outputs from the same documents. The authors mentioned that the drop was due to differences in statistical patterns, rather than domain-related factors. The *EAGLE* framework in [11] also raised similar concerns, with performance decreasing by 5-15% when tested on unseen models like GPT-3.5 and Claude, owing to distributional mismatches between LLMs. Lastly, a study using the *MIRAGE* benchmark [21] observed a 17% drop in AUROC for a RoBERTa-Base detector trained on GPT-3.5 text when tested on Claude-3.5-haiku outputs, which the authors attributed to increased variability in token sequences, rather than domain-related shift. Collectively, these studies indicate that the decline in these detectors performance is primarily due to LLMs differences, emphasizing the need for more robust detection systems.

A schematic comparative summary across multiple models and datasets is shown in Table 1.

Table 1: Comparison across the reviewed methods. For each class, we indicate the level of adaptability to new LLMs and whether the models use multi-scale feature (MSF) fusion.

Methods	**Description**	**Adaptability**	**MSF**
Statistical [16, 35, 6, 5]	Uses features like word frequency, sentence length, and n-grams to detect AI-generated text.	✗	✗
Linguistic [22, 29]	Uses high-order n-grams and BScore to detect LLM-generated text based on similarity features.	✗	✗
ML [65, 63, 55]	Uses classifiers based on stylistic features (e.g., syllable count, word length, sentence structure).	Very Limited	✗
DL [7, 46, 27]	Uses fine-tuning transformer models, which usually show higher performances in AI-text detection.	Limited	✗
Our Method	Uses continual learning, thus adapting the system to evolving LLMs without retraining from scratch.	Very High	✓

2.5. Limitations and Research Gap

Current detection methods struggle with identifying content generated by new, advanced models, such as ChatGPT-4 [53], which they have not encountered before. Many detectors, as highlighted in recent studies [56], often fail to recognise these new LLMs, leading to poor detection performance. For example, the OpenAI detector [67], which was trained on GPT-2 data, achieves only a 74.74% AUROC when identifying texts from newer models like GPT-3.5-Turbo and GPT-4. However, it performs accurately with GPT-2-generated texts. This indicates a significant challenge for detectors to generalise across different LLMs, especially those not present in the training dataset. Mostly existing detectors face similar difficulties in adapting to unseen and new models, despite showing good results with known models. This highlights the need for a detection system that can adapt and generalise effectively across various LLMs is crucial for real-world applications.

To address these gaps, in this research, we propose a novel dynamic fusion framework that leverages multi-scale feature fusion to extract diverse features from various AI models. By incorporating continual learning that enables the detection system to adapt to new and emerging LLMs over time without requiring retraining from scratch. This combination allows for a more robust, scalable, and long-term solution to cross-LLMs detection, even with a small sample size, ensuring that detectors can generalise well to newly introduced LLMs in real-world scenarios.

3. Workflow overview

We present a methodology for building our AI-generated text detection framework and evaluating its performance that goes through a pipeline consisting of the four main phases below.

1. Preparation of the dataset in sub-datasets for continual learning (section 3.1);
2. Designing a new framework for AI-generated text detection (section 3.3);

3. Adopting the continual learning strategy (section 3.4);
4. Evaluating the model performance (sections 4).

Detailed elaborations of these steps are provided in the following sections, and a graphical overview is depicted in Figure 1.

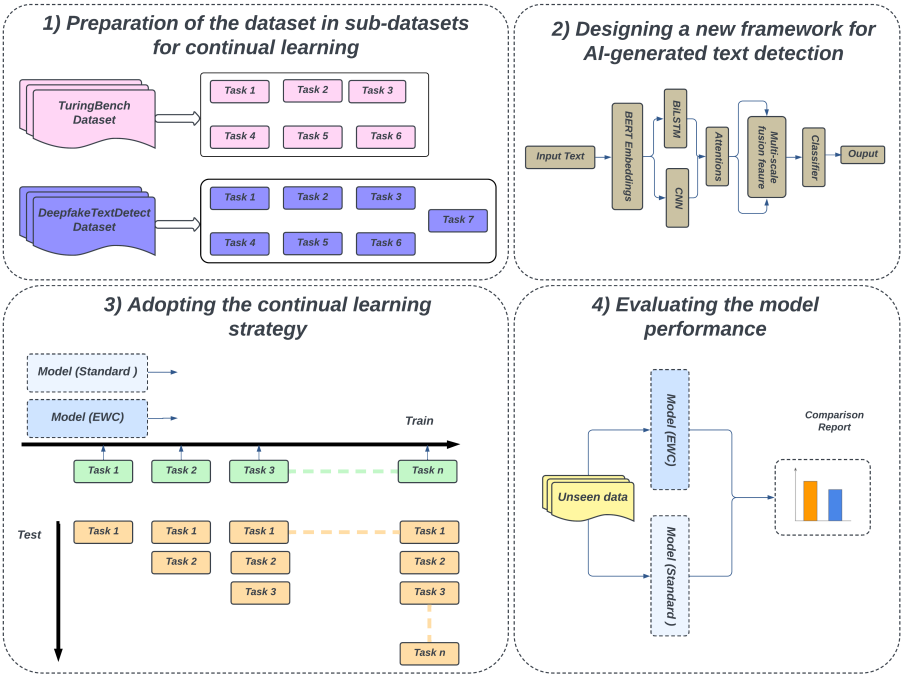


Figure 1: Methodology, a graphical overview.

3.1. Datasets

To evaluate the effectiveness of our framework, we employ two well-known public benchmark datasets, `TuringBench` [72] and `DeepfakeTextDetect` [42]. Both contain a balanced mix of human-written and AI-generated texts from a wide range of LLMs, making them highly relevant for studying detection in dynamic and evolving scenarios.

The `TuringBench` dataset explores "Turing tests" for neural text generation includes $10k$ human-written content primarily from news articles such as CNN.

It features articles ranging from 200 to 400 words alongside text from 19 text generation models, including GPT-1, various GPT-2 versions, GPT-3, multiple GROVER versions, CTRL, XLM, XLNet, FAIR, Transformer-XL, and PLM variants. Each model supplied $8k$ samples, classified by label.

The `DeepfakeTextDetect` dataset combines human-written text from ten diverse datasets such as news articles, stories, scientific writing, and etc. this dataset is pair with 27 major LLMs, including those of OpenAI, LLaMA, and EleutherAI, as well as pieces of text produced by GPT-4 and their paraphrased version to further challenge detection systems.

3.2. Data preparation for continual learning

To simulate a continual learning scenario, each dataset is divided into smaller subsets, referred to as "tasks" (Figure 1). In our framework, a task represents a detection process performed using a dedicated pool of large language models (LLMs). Moreover, a task is a cohesive unit of AI-generated text samples within a specific time window, where models are selected based on their prevalence, architectural advancements, and data availability during that period (details are presented in Table 2 and Table 3). We employed a time-based grouping strategy to ensure that each task corresponds to a specific set of LLMs, reflecting the temporal evolution of AI-generated content, as newer models are continuously introduced. This approach is designed to mimic how newer LLMs are introduced and how the characteristics of generated text change over time. As new LLMs are introduced, the model is updated incrementally, learning from each new task without needing to retrain from scratch. This ensures that the model continuously adapts to the evolving landscape of AI-generated text.

To maintain a balanced and fair evaluation, each task is designed with an equal distribution of human- and AI-generated text. The TuringBench dataset is divided into six subdatasets, each forming a unique task as detailed in Table 2, while the DeepfakeTextDetect dataset is divided into seven subdatasets, as shown in Table 3. This structure ensures that the performance of the model is evaluated in different pools of LLM and varying types of AI-generated content,

which is consistent with the dynamic nature of AI text generation.

Table 2: Tasks (sub-datasets) information on the AI-generated text (`TuringBench` dataset)

Task	**AI-generated text**	
	Models	**Comments**
Task 1	GPT-1 [Early 2019]	First powerful AI system for text generation.
Task 2	GPT-2 variants : small, medium, large, xl & pytorch [Mid and Late 2019]	Marked a significant leap in model scale and complexity.
Task 3	GPT-3, CTRL, GROVER-(base, large & mega) [2020]	GPT-3 ($175 \cdot 10^9$ parameters) with specialised text models.
Task 4	XLM, XLNet-(base & large) [2020-21]	Handle multi-lingual , and long-range dependencies
Task 5	FAIR (WMT19), FAIR (WMT20), Transformer-XL [Mid 2021]	Specialised translation models and memory-efficient sequence handling.
Task 6	PLM-(distill & GPT-2) [Late 2021]	Optimised models for faster text generation

Table 3: Tasks (sub-datasets) information on the AI-generated text (`DeepfakeTextDetect` dataset)

Task	**AI-generated text**	
	Models	**Comments**
Task 1	GPT-J-6B [Early 2021]	Early phase models, smaller and simpler, foundational to LLM development.
Task 2	Early GPT-3 Series, text-davinci-002 [2021]	Initial GPT-3 models forming the foundation for large language models
Task 3	text-davinci-003, gpt-3.5-turbo [2022]	Enhanced versions of GPT-3, . offering better performance and fine-tuning
Task 4	FLAN-T5-(small, base), LLaMA(7B,13B) [Early 2023]	LLMs offering improved scalability and fine-tuning for diverse tasks.
Task 5	GPT-NeoX-20B, FLAN-T5-large, OPT(125M,350M,1.3B, 2.7B, 6.7B, 13B, 30B,iml-max-1.3B,iml-30B), LLaMA(30B, 65B) [Mid 2023]	Larger, more complex models for improved performance and task versatility.
Task 6	FLAN-T5(xl, xxl) [2023]	Cutting-edge models handling complex tasks with significant performance advancements.
Task 7	BLOOM-7B, T0(3B,11B)[Late 2023]	Advanced models designed for complex tasks with specialized capabilities.

3.3. The proposed framework

We proposed a groundbreaking deep learning architecture to detect AI-generated text by extracting and fusing local and global semantic features, while incrementally adapting to new LLMs without retraining from scratch, as shown in Figure 2. The framework consists of six different components.

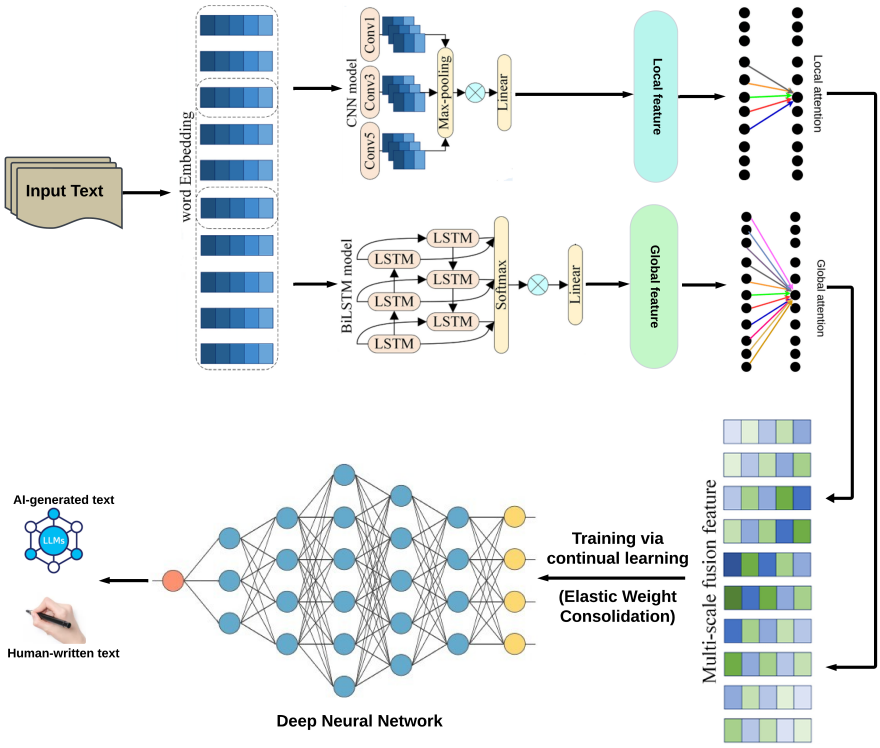


Figure 2: Proposed framework

Input Representation via Pre-trained BERT

We employed DistilBERT-base-uncased [61] to transform raw input text into meaningful embeddings due to its ability to produce contextual, semantically rich representations with low computational overhead, making it suitable for our framework. The input text is tokenized into a sequence of m tokens, where m varies based on the text length. Each token is mapped to a d-dimensional

embedding vector, yielding an embedding matrix $E \in \mathbb{R}^{m \times d}$, with $d = 768$. This transformation process is defined as Eq. 1.

$$E = \text{DistilBERT}_{\text{base-uncased}}(T) = [e_1, e_2, \ldots, e_m], \qquad e_i \in \mathbb{R}^d \tag{1}$$

where T denotes the input text sequence, $E \in \mathbb{R}^{m \times d}$ is the embedding matrix, and each $e_i \in \mathbb{R}^d$ is a column vector representing the contextual embedding of the i-th token, derived from the pre-trained DistilBERT model. This embedding matrix captures both syntactic and semantic features of the input text, enabling effective downstream processing for our AI-generated text detection task.

Text length analysis has been performed to select a suitable backbone encoder, showing that the vast majority of text in both datasets fits within the DistilBERT 512 token window. Tables A.14, A.15 and A.16 summarising input length statistics and ablation results are included in Appendix A, demonstrating the overall impact of encoder choice on model performance.

Local Feature Extraction with CNN

We used CNN layers to the BERT embeddings E from previous step to extract local features [32], effectively modelling n-gram patterns and short-range dependencies. Max-pooling is used to reduce feature dimensionality while preserving high-activation patterns. We used PyTorch's default hyper-parameters for convolutional layers to capture local features like edges. Additional hyper-parameters, such as kernel size and number of filters, are empirically tuned via grid search as listed in Table 4.

Table 4: CNN Architectural Hyper-Parameter Settings

Hyper-Parameter	**Value**	**Range Tested**
`number_of_input_channels`	768	Fixed (BERT embeddings)
`number_of_layers`	3	2, 3, 4, 5
`cnn_filters`	64	32, 64, 128
`kernel_size`	3	3, 5, 7
`padding`	1	0, 1, 2

Global Feature Extraction with BiLSTM

In addition to CNN, Bidirectional LSTM is applied on embeddings E to capture long-range dependencies. It processes the sequence in both forward and backward directions and outputs concatenated hidden states that encode global context. This step ensures the model discovers sentence-level semantics and broader narrative flow as well. BiLSTM hyper-parameters, like hidden size and bidirectional setting, as shown in Table 5, were tuned via grid search to optimise performance.

Table 5: BiLSTM Architectural Hyper-Parameter Settings for Text Classification

Hyper-Parameter	**Value**	**Range Tested**
`input_feature_vector`	768	Fixed (BERT embeddings)
`num_layers`	3	1, 2, 3
`lstm_hidden`	128	64, 128, 256
`bidirectional`	True	True, False
`batch_first`	True	Fixed

Attention Mechanisms

Attention mechanisms are applied in two scopes: Local Attention on CNN outputs (F_{local}) and Global Attention on BiLSTM outputs (F_{global}). Local Attention focusses on contextually relevant local features by using a fixed window approach. This mechanism enables the model to capture finer details within a smaller localised context. Global Attention, on the other hand, allows the model to weight features across the entire sequence, enabling it to prioritise important long-distance dependencies. The attention scores for both local and global attention are computed using the query-key-value scheme and softmax normalisation, as follows:

The Local Attention is calculated as Eq. 2:

$$\text{Attention}_{local}(q, K, V) = \text{softmax}\left(\frac{qK^T}{\sqrt{d_k}}\right)V \tag{2}$$

Where q is the query vector, K is the key matrix, V is the value matrix, and d_k is the dimension of the key. This equation computes attention for each local window, focussing on a limited context around each element.

The Global Attention is defined as Eq. 3:

$$\text{Attention}_{global}(q, K, V) = \text{softmax}\left(\frac{qK^T}{\sqrt{d_k}}\right)V \tag{3}$$

In global attention, the query q is compared with all keys K from the entire sequence, allowing the model to capture long-range dependencies throughout the input sequence. These attention scores are then normalised via softmax, ensuring that the weight across the sequence is distributed proportionally to the relevance of each part of the sequence. These local and global attention calculations used here follow those in previous work [48, 3].

Multi-scale Feature Fusion

The local and global features enhanced with attention mechanisms are concatenated into a single multi-scale feature using Eq. 4 below.

$$F_{\text{fused}} = [F_{\text{local}}, F_{\text{global}}] \tag{4}$$

Final Classification via DNN

We employ a Deep Neural Network (DNN) to classify text as AI-generated or human-written, leveraging a fused feature vector $\mathbf{F}_{\text{fused}}$. The DNN comprises fully connected layers that transform $\mathbf{F}_{\text{fused}}$, followed by a sigmoid activation function. The network configuration, detailed in Table 6, is optimised to effectively distinguish AI-generated text. Hyperparameters were optimised via a grid search, selecting the best combination based on performance.

Table 6: DNN Architectural Hyper-Parameter Settings

Hyper-Parameter	**Value**	**Range Tested**
`hidden_layer`	5	3, 5, 7
`threshold`	0.5	0.3, 0.5, 0.7
`loss_function`	`BCE`	`BCE`
`activation_function`	`ReLU (hidden layer), Sigmoid (output layer)`	`ReLU,Tanh Sigmoid`

3.4. Training with Continual Learning

The goal of continual learning is to let a model to learn new tasks while preserving knowledge from previously learned tasks. To achieve this, we utilised Elastic Weight Consolidation (EWC), which helps prevent Catastrophic interference [39]. This section describes methodology, training setup, and key components of the continual learning processes, as shown in the third block of Figure 1.

Elastic Weight Consolidation

EWC balances plasticity for quick adaptation to new data with stability to retain past knowledge. Excessive plasticity can lead to forgetting old tasks, so stability helps preserve learned information. The EWC assesses the weights based on their importance in prioritising essential weights while allowing others to adapt. The challenge lies in maintaining key parameters from past tasks while learning new ones.

To train a model with EWC, the dataset is divided into tasks. As shown in Figure 3, at each step t of the EWC process, two key components must be calculated: First, the optimal weight matrix W^*_{t-1}, obtained by training the model on the previous task $(t-1)$; and second, the Fisher information matrix F_{t-1}, which quantifies the importance of the parameters learned from Task $t-1$, computed following the procedure outlined in [39]. The Fisher matrix is recalculated for each task, starting from Task 1 and continuing through each subsequent task, ensuring incremental learning while retaining important knowledge from prior tasks. We have used the diagonal approximation approach to compute the Fisher matrix, which reduces computational complexity and enhances scalability.

Hence, the the matrices for task 1 must be calculated before iterating tasks 2 to n. Information on EWC and parameters (as used in this study) is given in Algorithm 1.

According to Algorithm 1, a Fisher information matrix-based regularisation is added to the loss function to protect important weights from earlier tasks.

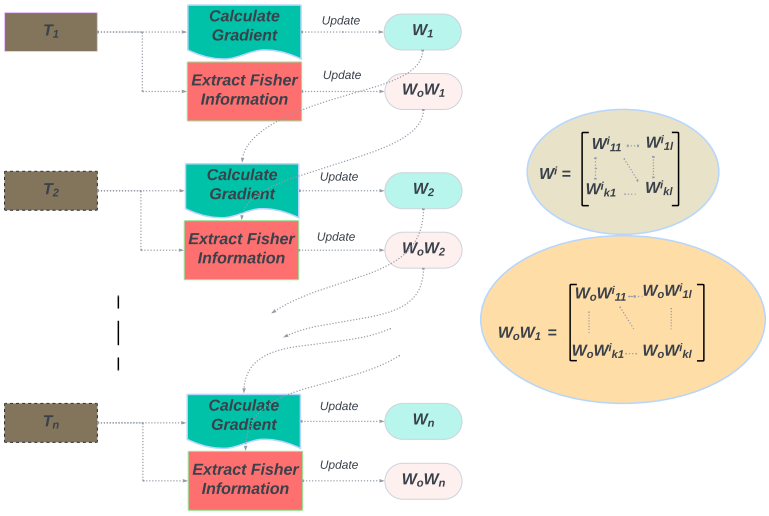


Figure 3: EWC procedure - a graphical representation.

Algorithm 1 EWC Pseudocode [39]

Input:

Tasks 1 to n

λ: Regularisation strength ▷ We empirically set it to $1e3$

$\mathcal{L}_t$: Loss function for task $t = 1, \ldots, n$ ▷ We use BCE (Eq. 6)

Train on task 1 to obtain initial parameters $\mathbf{W}_1^*$.

$\mathbf{W}_1^* = \arg\min_{\mathbf{W}} \mathcal{L}_1(\mathbf{W})$

Calculate the Fisher information matrix for task 1.

$F_1 = \mathbb{E}\left[\nabla_{\mathbf{W}} \log p(\text{Task}_1|\mathbf{W}) \nabla_{\mathbf{W}} \log p(\text{Task}_1|\mathbf{W})^T\right]$ ▷ see [39]

for $(t == 2; t <= n; t++)$ **do**

Calculate the Fisher information matrix for task t:

$F_t = \mathbb{E}\left[\nabla_{\mathbf{W}} \log p(\text{Task}_1|\mathbf{W}) \nabla_{\mathbf{W}} \log p(\text{Task}_1|\mathbf{W})^T\right]$ ▷ to use in Task $t+1$

Train on task t with the loss function incorporating EWC regularisation:

$\mathcal{L}(\mathbf{W}) = \mathcal{L}_t(\mathbf{W}) + \sum_{i=1}^{t-1} \frac{\lambda}{2} F_i \left(\mathbf{W}_i - \mathbf{W}_i^*\right)^2$

$\mathbf{W} \leftarrow \mathbf{W} - \alpha \nabla_{\mathbf{W}} \mathcal{L}(\mathbf{W})$ ▷ Update $\mathbf{W}$

end for

Output: Optimised parameters $\mathbf{W}$.

This helps the model learn new tasks while maintaining past performance. Parameters are updated via gradient descent to minimise both task-specific loss and the regularisation, ensuring stability during adaptation.The loss function in EWC consists of two components: the loss associated with the current task and the distance between the weight vector of the current task and that of the previous task. we report EWC loss function in Eq. 5,

$$L(W) = L_B(W) + \lambda \sum_i \frac{1}{2} F_i (W_i - W^*_{A,i})^2 \tag{5}$$

where $L_B(W)$ represents the loss for the current task, λ denotes the relative importance of the previous task in comparison to the current task, W_i represents the parameters of the current task, $W^*_{A,i}$ represents the parameters of the previous task, and F_i is the Fisher information matrix for task i.

The training loss is often the binary cross-entropy (BCE) loss, commonly used for binary classification tasks, which we report in Eq. 6,

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(\hat{y_i}) + (1 - y_i) \log(1 - \hat{y_i})] \tag{6}$$

where y_i is the binary label (0/1) for the i^{th} sample, $\hat{y_i}$ is the predicted probability (0 to 1) for the i^{nd} sample, and N is the total number of samples.

Note that when applying EWC to our framework, we train it using the learning hyperparameters indicated in Table 7.

3.5. Experimental setup

We implemented a dynamic fusion framework for AI-generated text detection in Python 3.11, using PyTorch 2.6.0 and the Hugging Face `transformers` library, running on an NVIDIA Tesla A100 GPU (40 GB VRAM, CUDA 12.1). The framework employs the pre-trained `distilbert-base-uncased` model to generate text embeddings. To optimise computational efficiency and memory usage, we applied mixed-precision training using `GradScaler` and `autocast`.

The model was trained for 5 epochs with the Adam optimiser, using an 70/15/15 train/validation/test split. This split was achieved using the train_test_split

function from scikit-learn, which applies a random partitioning method to ensure that each experimental run uses a different subset of the data. The process was repeated for five independent runs to evaluate the model's performance consistently. The results reported represent the average performance across these runs, ensuring that the findings are robust and not dependent on any single data partitioning.

For reproducibility, key learning parameters are detailed in Table 7, with additional architecture parameters provided in their respective section tables. Hyperparameter optimisation was performed via grid search, where we explored variations in learning rate, batch size, dropout rate, and other key parameters. The results of this optimisation are detailed in Table 7.

Table 7: Learning Hyper-Parameter Settings

Hyper-Parameter	**Value**	**Range Tested**
Learning Rate	1e-5	1e-5, 2e-5, 5e-5
Batch Size	16	8, 16, 32
Number of Epochs	5	3, 5
Optimiser	Adam	Adam, SGD
Dropout Rate	0.3	0.1, 0.3, 0.5
EWC regularisation coefficient (λ)	10^3	$10^1, 10^2, 10^3$

We evaluated the performance of our model using accuracy, precision, recall and F1-score. These metrics provide a comprehensive understanding of the model's performance.

Accuracy (A) measures the overall correctness of the model as Eq. 7, where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives.

$$A = \frac{TP + TN}{TP + TN + FP + FN} \tag{7}$$

Precision (P) measures the proportion of correctly predicted positive instances (AI-generated text) out of all instances predicted as positive using Eq. 8. It is important for ensuring that when the model classifies text as AI-generated, it is likely to be correct.

$$P = \frac{TP}{TP + FP} \tag{8}$$

Recall (R) measures the proportion of correctly predicted positive instances (AI-generated text) out of all actual positive instances using Eq. 9. It is critical in applications where it is more important to detect all AI-generated text, even at the cost of some false positives.

$$R = \frac{TP}{TP + FN} \tag{9}$$

F1-Score ($F1$) is the harmonic mean of precision and recall, balancing the trade-off between them using Eq. 10. The F1-score provides a balanced view of the model's ability to detect AI-generated text, especially when there is an uneven class distribution.

$$F1 = 2 \times \frac{P \times R}{P + R} \tag{10}$$

These metrics were chosen because they offer a thorough evaluation of the model's performance to detect both true positives (AI-generated text) and minimise false negatives and false positives. Using these metrics, we ensure a more robust and nuanced assessment of the capabilities of our model. Furthermore, these results were compared with the baseline models reported in the original papers [72, 43] (see Table 11).

We also use further metrics from [38] to measure the competence of a model in retaining past knowledge while learning new tasks using Eq. 11, 12, and 13.

$$\Omega_{\text{base}} = \frac{1}{T} \sum_{i=1}^{T} \alpha_{\text{base},i} \tag{11}$$

$$\Omega_{\text{new}} = \frac{1}{T} \sum_{i=1}^{T} \alpha_{\text{new},i} \tag{12}$$

$$\Omega_{\text{all}} = \frac{1}{T} \sum_{i=1}^{T} \alpha_{\text{all},i} \tag{13}$$

Here, T is the total task count, $\alpha_{\text{base},i}$ is the base task's accuracy after learning task i, α_{ideal} is the offline model's base task accuracy, $\alpha_{\text{new},i}$ is the

accuracy on task i post-learning task i, and $\alpha_{\text{all},i}$ is the overall accuracy on all data after learning task i. In particular, Ω_{base} measures base task retention after new learning, Ω_{new} gauges performance on newly learned task i, and Ω_{all} examines retaining old knowledge while acquiring new.

4. Results and Analysis

This section outlines the effectiveness evaluation of our proposed framework using two benchmark datasets, TuringBench and DeepfakeTextDetect, and compares the results with state-of-the-art methods. We analyse the performance of the proposed model, which employs a continual learning approach, by comparing it to the base model that does not incorporate continual learning.

4.1. Performance of the detection model across multiple language models

Prior to implementing continual learning, we first evaluated our proposed detection model on each large language model from both datasets, without dividing the datasets into tasks. Table 8 presents experimental results, highlighting the growing difficulty in detecting AI-generated text as language models evolve. As shown in the `TuringBench` dataset, earlier models such as GPT-1 and GPT-2-small were easily detected, with accuracy rates of 99.35% and 96.80%, respectively, due to identifiable patterns in their output, such as repetitive structures, unnatural phrasing, and limited contextual understanding. These early models lacked the sophistication to generate convincingly human-like text, making detection straightforward.

In contrast, in the `DeepfakeTextDetect` dataset, it is illustrated that more advanced models such as GPT-3.5-turbo and LLaMA-65B exhibited comparatively lower detection accuracy at 93.73% and 88.57%, respectively. This decline in detection accuracy reflects their improved ability to mimic human writing, with more coherent, contextually appropriate responses, and better grammar, making the output more natural and reducing detectable markers.

The trend observed across model families shows that the detection accuracy decreases as models evolve and the sophistication of the model increases. If

Table 8: Performance of the detection model across different language models on `TuringBench` and `DeepfakeTextDetect` datasets. For the full detailed result tables, see Tables B.17 and B.18 of Appendix B.

TuringBench Dataset								
Text Generator	Overall Performance		AI-Generated Text			Human-Written Text		
	Accuracy	**AUC**	**Precision**	**Recall**	**F1-Score**	**Precision**	**Recall**	**F1-Score**
GPT-1	99.35%	99.36%	99%	99%	99%	99%	99%	99%
GPT-2 (small)	96.80%	96.81%	96%	98%	97%	98%	96%	97%
GPT-2 (large)	96.22%	96.18%	96%	97%	96%	97%	95%	96%
GPT-3	95.76%	95.71%	95%	97%	96%	97%	94%	96%
GROVER (base)	97.99%	97.99%	98%	98%	98%	98%	98%	98%
...	...	...	...	...	...	...	...	...
DeepfakeTextDetect Dataset								
text-davinci-002	84.54%	85.25%	64%	87%	73%	95%	84%	89%
gpt-3.5-turbo	93.73%	94.23%	83%	95%	89%	98%	93%	96%
LLaMA-65B	88.57%	81.62%	72%	70%	71%	93%	93%	93%
GLM130B	83.17%	82.98%	82%	87%	84%	85%	78%	82%
FLAN-T5-base	91.47%	85.42%	74%	76%	75%	95%	95%	95%
...	...	...	...	...	...	...	...	...

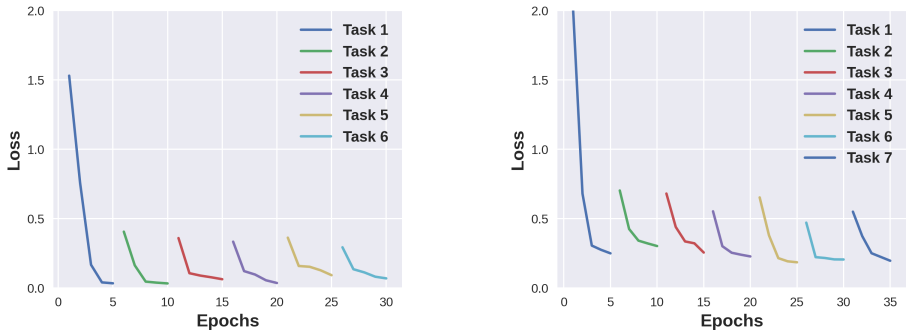
detection methods remain static, their effectiveness diminishes over time. This shift from easily detectable to more evasive models underscores the need for dynamic detection systems. The 10-15% decline in detection accuracy across generations highlights that static detectors quickly become obsolete. Furthermore, performance variations across different model architectures, such as 98.38% for OPT-13B versus 88.57% for LLaMA-65B, emphasise the need for adaptive detection systems capable of accommodating new large language models and evolving training techniques.

4.2. Continual Learning: Classic VS Continual learning

To demonstrate the benefits of adopting continual learning, we trained the "standard model" which, in our study, refers to a non-continual model trained on both datasets without any mechanisms for continual learning (i.e., only $\mathcal{L}_{\text{BCE}}$ is used — see Equation 6). This model serves as a baseline to compare the performance of our proposed framework. The standard model processes the data statically, without any adaptation to new tasks, allowing us to evaluate

how our continual learning framework addresses issues such as forgetting and adaptation to new tasks. We then analysed the results and compared them with those obtained by the proposed framework, which incorporates the EWC procedure in training, as shown in Section 3.4.

Figure 4 shows that training on tasks without EWC causes a sharp initial loss spike due to the difficulty of adapting to a new task. This results in higher error rates and indicates a reset of acquired knowledge in our application case, a phenomenon known as 'catastrophic forgetting' [51], where new information overwrites existing knowledge.



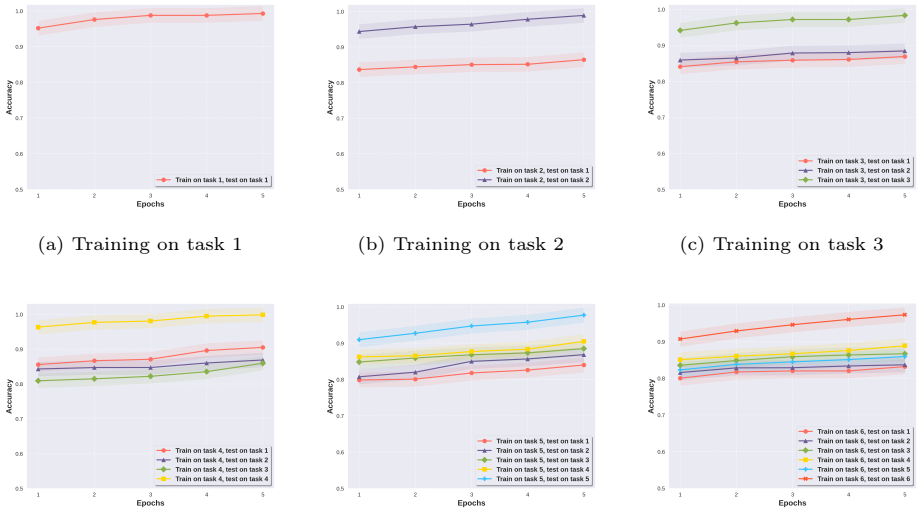
(a) `TuringBench` dataset loss

(b) `DeepfakeTextDetect`

Figure 4: Loss function trend for training without EWC.

Evaluating a multitask learning system requires assessing both current task performance and knowledge retention from past tasks. Removing EWC undermines this, as illustrated in Figures 5 and 6.

A clear trend of accuracy degradation on earlier tasks is observed as the model is trained on subsequent tasks. For example, Figures 5-(b) and 6-(b) show that when the model is trained in task 2, there is a significant drop in precision in task 1. This trend persists and becomes more evident as we progress on the next tasks. To prevent this problem, EWC introduces a regularisation term that limits changes in critical parameters, penalising deviations from their optimal values noted in previous tasks (see $\mathcal{L}_{\text{EWC}}$ in Algorithm 1). Our results confirm this beneficial effect.



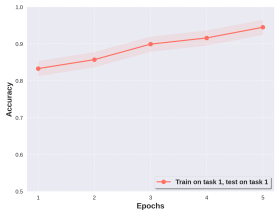
(a) Training on task 1 (b) Training on task 2 (c) Training on task 3

(d) Training on task 4 (e) Training on task 5 (f) Training on task 6

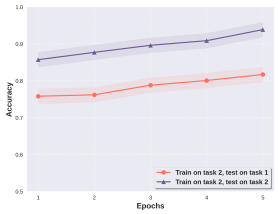
Figure 5: Accuracy of the standard model on previously learned tasks on `TuringBench`.

Figure 7 presents the BCE loss ($\mathcal{L}_{\text{BCE}}$) and EWC loss ($\mathcal{L}_{\text{EWC}}$) of the model during continual learning. Figure 7-(a) and 7-(c) demonstrate a continual drop in $\mathcal{L}_{\text{BCE}}$ loss for each task, showing the model's effectiveness in reducing prediction error across both datasets. The persistently low $\mathcal{L}_{\text{BCE}}$ loss signifies successful learning and stabilisation on individual tasks. Fluctuations in $\mathcal{L}_{\text{EWC}}$ loss during transitions to new tasks highlight the model's effort to balance preserving prior knowledge, enforced by EWC regularisation, with learning new tasks (Figure 7-(b) and 7-(d)). The rise in EWC loss indicates difficulty in retaining accuracy on prior tasks. The spike size highlights parameter importance, as EWC punishes significant changes crucial for previous task performance. This interplay reflects the challenges in continual learning frameworks.

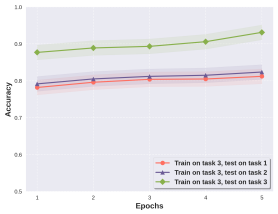
Figures 8 and 9 provide a detailed analysis of the accuracy of the model with continual learning on consecutive tasks during training. Generally, this configuration retains knowledge without significant performance degradation, with only one visible ascent in accuracy while transferring knowledge from task 1 to task 2.



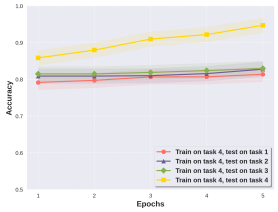
(a) Training on task 1



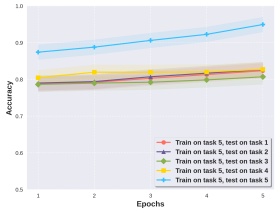
(b) Training on task 2



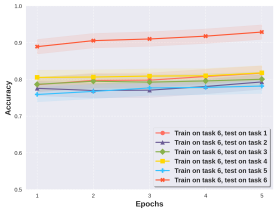
(c) Training on task 3



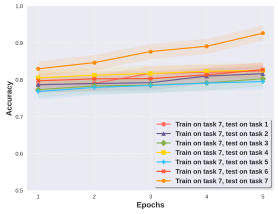
(d) Training on task 4



(e) Training on task 5



(f) Training on task 6

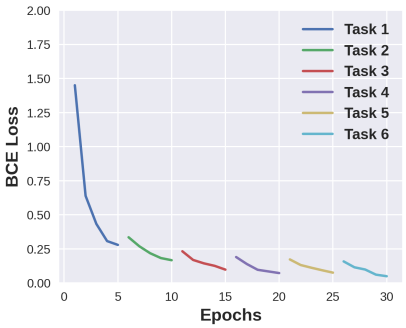


(g) Training on task 7

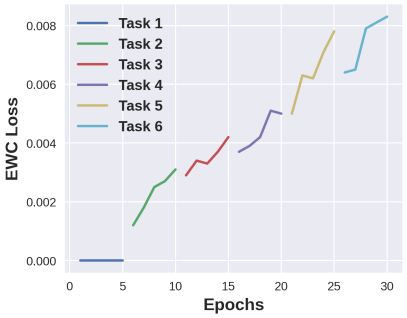
Figure 6: Accuracy of the standard model on previously learned tasks on `DeepfakeTextdetect` dataset

Figure 10 and 11 compares the accuracy of two model variants, with and without EWC, across tasks. Each row shows a task's performance as the model advances from tasks 1 to n. Again, results support using the continuous learning strategy.

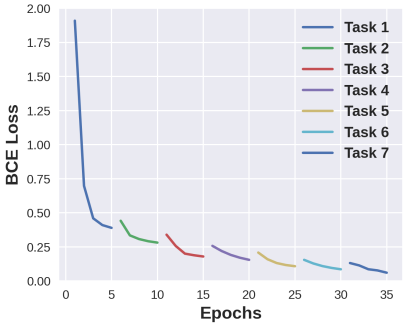
Table 9 shows that continuous learning with EWC is preferable, demonstrating superior *overall performance* (Ω_{all}) and *retention of knowledge* (Ω_{base}), although slightly inferior in terms of *new task knowledge retantion* Ω_{new}.



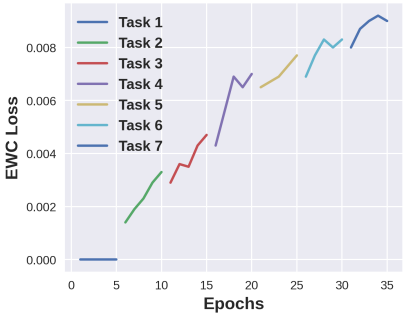
(a) BCE loss on `TuringBench` dataset



(b) EWC loss on `TuringBench` dataset



(c) BCE loss on `DeepfakeTextdetect` dataset

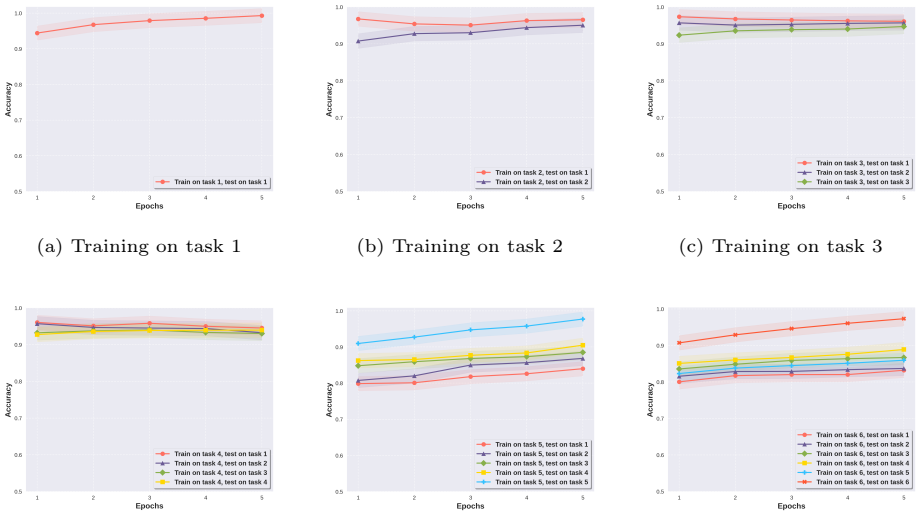


(d) EWC loss on `DeepfakeTextdetect` dataset

Figure 7: Losses of the continual model across sequential tasks on `TuringBench` and `DeepfakeTextdetect` datasets

Table 9: Performance comparison of the standard model (Without EWC) versus the continual learning proposed model (With EWC) on `TuringBench` and `DeepfakeTextDetect` datasets using Kemker's continual learning metrics. The best value for each metric per dataset is highlighted in bold.

Metrics	`TuringBench`		`DeepfakeTextDetect`	
	Without EWC	With EWC	Without EWC	With EWC
Ω_{base}	0.843	**0.997**	0.865	**0.943**
Ω_{new}	**1.023**	1.002	**0.974**	0.949
Ω_{all}	0.955	**0.984**	0.869	**0.930**



(a) Training on task 1 (b) Training on task 2 (c) Training on task 3

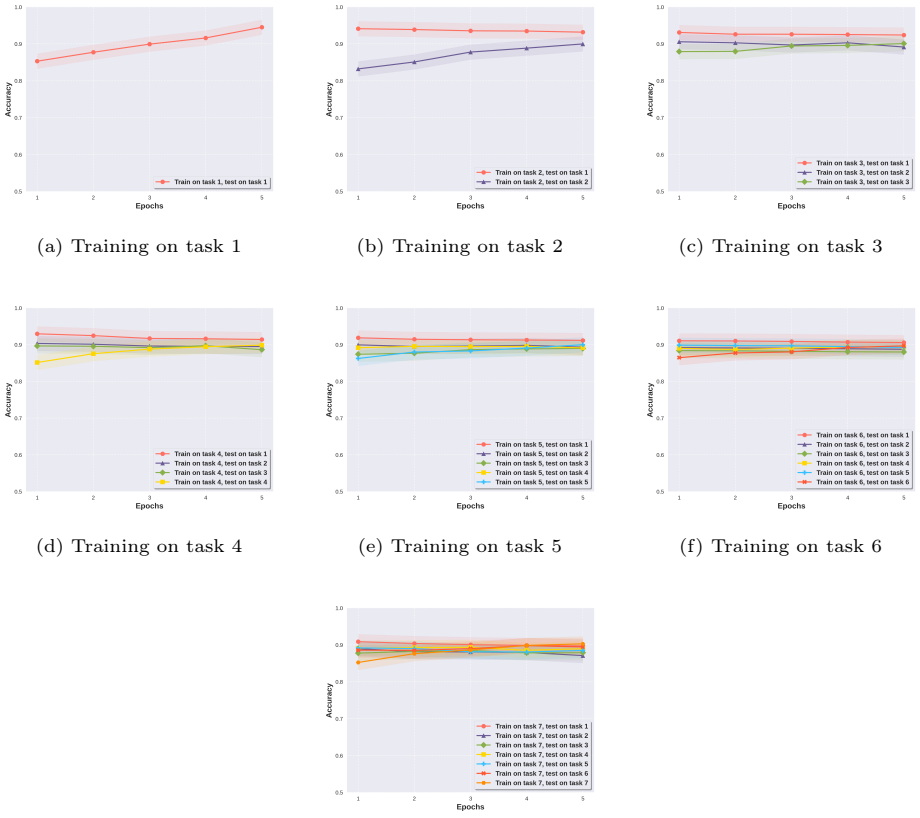
(d) Training on task 4 (e) Training on task 5 (f) Training on task 6

Figure 8: Accuracy of the continual model on previously learned tasks on `TuringBench` dataset

4.2.1. Results on unseen data

Figure 12 presents the performance on unseen task data. The model with EWC consistently outperforms the standard model across all tasks on both datasets. This superior performance indicates that the proposed model effectively integrates and generalises knowledge from previous tasks, while the standard model shows a decline in accuracy, reflecting a poorer transfer of knowledge. These results highlight that the proposed model's knowledge preservation mechanisms enhance its generalisation ability, supporting the hypothesis that continual learning strategies can improve resistance to catastrophic forgetting and enable more effective application of prior knowledge to new tasks.

Statistical analysis:. To conclude this analysis with statistical rigour, a two-sample t-test was performed on the test results (accuracy) using a significance level as indicated in 0.05. The results presented in Table 10 demonstrate a statistically significant difference in accuracy between the two distributions, confirming that the model that employs continual learning exceeds the baseline model without it.



(a) Training on task 1 (b) Training on task 2 (c) Training on task 3

(d) Training on task 4 (e) Training on task 5 (f) Training on task 6

(g) Training on task 7

Figure 9: Accuracy of the continual proposed model on previously learned tasks on `DeepfakeTextdetect` dataset

Table 10: Statistical analysis on testing datasets for the two models with (reference) and without EWC.

Dataset	**T-statistic**	**P-value**	**Significance**
`DeepfakeTextdetect`	-10.41	2.31×10^{-7}	✓
`TuringBench`	-7.43	2.31×10^{-5}	✓

4.3. Comparisons with baselines and state-of-the-art models

We performed a thorough comparison of our proposed framework against the best existing results for the `TuringBench` benchmark dataset reported in [72] and several state-of-the-art models evaluated on `DeepfakeTextDetect` dataset

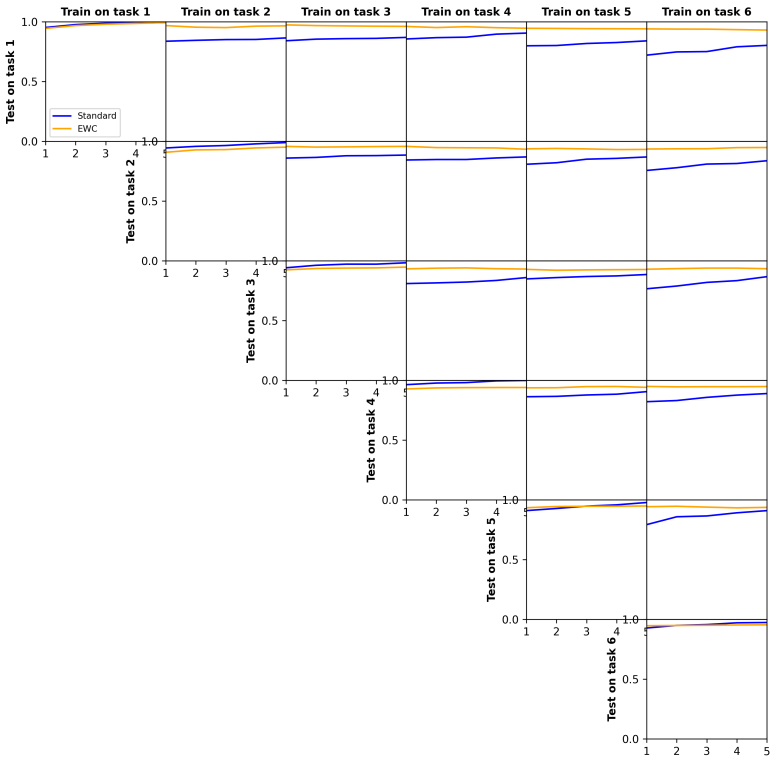


Figure 10: Comparison between the model variants with and without EWC across all tasks on the `TuringBench` dataset.

in [43], as outlined in Section 3.5. The analysis demonstrates the superiority of our model, with consistent outperformance in key accuracy metrics on both datasets, as shown in Table 11. This underscores the critical role of integrating diverse feature sources from different components of our framework for effective detection of AI-generated text.

Following are the comparison models used with the `TuringBench` dataset:

- Random Forest [12]. This ensemble of decision trees, utilising stylometric and lexical characteristics like word frequency and punctuation density, serves as a classical baseline for testing surface-level differences between human and machine text. It was trained with 100 trees, a maximum depth of 30, on TF-IDF features (1–3 grams) from the `TuringBench` training set.

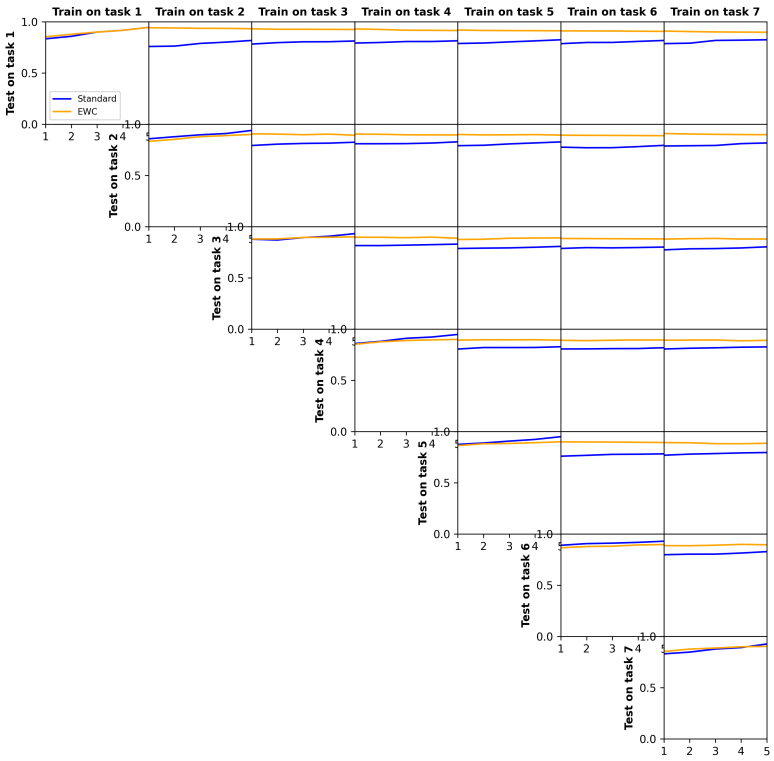
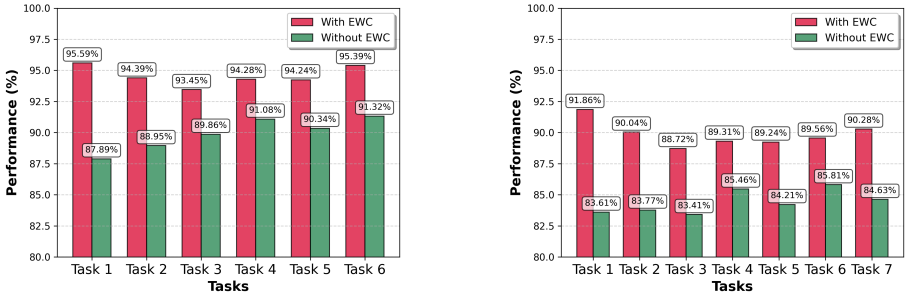


Figure 11: Comparison between the model variants with and without EWC across all tasks on the `DeepfakeTextDetect` dataset.



(a) `TuringBench` dataset

(b) `DeepfakeTextDetect` dataset

Figure 12: Comparison between the model variants with and without EWC on the `TuringBench` and `DeepfakeTextDetect` test set.

- Support Vector Machine (SVM) 3-grams [73]. This model uses character-level 3-gram TF-IDF features to detect short-range repetitions in synthetic text. It was implemented using scikit-learn's LinearSVC with a hinge loss of $C = 1.0$ and trained on normalised and tokenised texts.

- WriteprintsRFC [68]. This is a stylometric pipeline that uses Random Forest classification to detect AI text by analysing lexical, syntactic, and content features to find subtle style inconsistencies. Initially designed for authorship attribution, it applies a 200-tree Random Forest trained on balanced human/synthetic `TuringBench` data 1500.

- Syntax-CNN [33]. This convolutional neural network uses syntactic dependency paths to learn grammatical patterns distinguishing AI from human language. It features three convolutional layers (100, 150, 200 filters), kernel sizes 3–5, dropout 0.5, and was trained for 5 epochs using Adam optimiser (1×10^{-3}).

- N-gram CNN [13]. This is a character-level CNN that learns filters over n-gram windows to capture local lexical patterns. It was implemented with an embedding dimension of 256, kernel sizes of 3–5, max-pooling, and a dense softmax output. We used a learning rate of 10^{-3}.

- BertAA [18]. This is a BERT-base transformer adapted for binary AI vs human authorship attribution. We fine-tuned it for 5 epochs with a batch size of 16, a 512-token context, and a learning rate $2 \cdot 10^{-5}$.

- RoBERTa-Multinomial [47]. This is a multi-class RoBERTa-base classifier that predicts whether a text is generated by a model or written by a human author using deep contextual embeddings and multi-generator signals. It was trained for 5 epochs on the 20-class `TuringBench` label set with learning rate 10^{-5}.

- OpenAI Detector [67]. Developed by OpenAI for AI-text identification, this is a commonly used industry reference baseline. It was evaluated

via its public API on `TuringBench` test subsets without possible further fine-tuning.

For `DeepfakeTextDetect` dataset, the following are the comparison models used:

- DetectGPT [53]. This is a zero-shot method that flags text likely to be synthetic by using log-probability curvature under a language model, employing GPT-NeoX 20B for probability evaluation with 10 perturbations per document.
- GLTR [25]. This model analyses token probabilities to spot high-probability tokens typical of machine output. Features (token ranks top-10/100/1000) are input to a logistic regression classifier with 10-fold cross-validation.
- FastText [34]. This is a fast and interpretable baseline neural classifier using bag-of-words and sub-word n-gram embeddings. We implemented it with embeddings of dimensions 100, $lr = 0.1$, ngrams $= 2$, and we used 5 epochs.
- Longformer [9]. This model is based on a transformer architecture with sparse (global and local) attention, which efficiently handles long sequences and captures document-level coherence for detection. We fine-tuned it for 5 epochs on 512- to 4096-token inputs and a learning rate of 2×10^{-5}.

The `Longformer` model stands out for its ability to efficiently handle long input sequences, using a self-attention mechanism that scales linearly with the sequence length, making it well suited for AI text detection, as demonstrated by its high accuracy shown in the table 11. In contrast, `FastText`, a word-level classification model using bi-grams as features, is known for its efficiency in text classification tasks. It leverages sub-word information for more nuanced word representation, delivering competitive performance. However, its performance in AI-generated text detection lags, highlighting limitations in addressing the complexity of this task.

Furthermore, while `RoBERTa-Multinomial` and `OpenAI Detector` did not surpass our proposed model, both demonstrated notable accuracy improvements. Although not yet competitive with our approach, their detection methodologies offer valuable insights for future model enhancements.

Table 11: Comparison with state-of-the-art methods on `TuringBench` [72] and `DeepfakeTextDetect` [43] datasets. The best number is highlighted in bold.

Datasets	Methods	Accuracy
`TuringBench`	Random Forest [12]	61.47%
	SVM (3-grams) [73]	72.99%
	WriteprintsRFC [68]	49.43%
	OpenAI detector [67]	78.73%
	Syntax-CNN [33]	66.13%
	N-gram CNN [13]	69.14%
	BertAA [18]	78.12%
	RoBERTa-Multinomial [47]	81.73%
	Proposed Framework	**95.78%**
`DeepfakeTextDetect`	GLTR [25]	59.25%
	DetectGPT [53]	60.48%
	FastText [34]	80.23%
	Longformer [9]	92.02%
	Proposed Framework	**92.39%**

4.4. Ablation study

To evaluate the effectiveness of the proposed fusion strategy, we conducted ablation studies to assess the impact of each component on our framework's performance across the `TuringBench` and `DeepfakeTextDetect` datasets. Specific components (e.g., BiLSTM, Attention mechanism, and EWC) were removed, and the model retrained. The full model showed robust performance, with removal of any component leading to significant performance drops as shown in tables 12 and 13. Specifically, removing CNN highlighted its importance in feature extraction, while removing BiLSTM and Attention reduced the model's ability to capture long-term dependencies and prioritise relevant features. The removal of EWC resulted in the large drop, emphasising its role in preventing catastrophic forgetting as shown in Section 4.2. These findings demonstrate the

critical contribution of CNN, BiLSTM, Attentions mechanism, and EWC for feature extraction, sequence learning, and mitigating forgetting.

Table 12: Ablation Study for the Proposed Model on the `TuringBench` Dataset.

Model Variant	CNN	BiLSTM	Attention	EWC	Accuracy
W/o CNN	×	✓	✓	✓	92.04%
W/o BiLSTM	✓	×	✓	✓	91.45%
W/o Attention	✓	✓	×	✓	93.68%
W/o EWC	✓	✓	✓	×	89.90%
Full Model	✓	✓	✓	✓	**95.78%**

Table 13: Ablation Study for the Proposed Model on the `DeepfakeTextDetect` Dataset.

Model Variant	CNN	BiLSTM	Attention	EWC	Accuracy
W/o CNN	×	✓	✓	✓	89.85%
W/o BiLSTM	✓	×	✓	✓	88.28%
W/o Attention	✓	✓	×	✓	90.76%
W/o EWC	✓	✓	✓	×	84.41%
Full Model	✓	✓	✓	✓	**92.39%**

5. Conclusion

This study presents a novel dynamic fusion framework that integrates multi-scale feature fusion with continual learning via EWC to address the growing challenge of detecting AI-generated text.

Experimental results on the two benchmark datasets, `TuringBench` and `DeepfakeTextDetect` show that the proposed model consistently outperforms state-of-the-art baselines. Notably, the continual learning strategy effectively mitigates catastrophic forgetting, enabling the model to retain knowledge across sequential tasks while adapting to new data distributions. The integration of CNN, BiLSTM, and attention mechanisms further enhances the model's ability to capture both local and global textual features, contributing to its superior classification accuracy.

Beyond the scope of AI-generated text detection, the proposed framework offers a scalable and generalisable solution for dynamic environments where data evolves over time. Its potential applications extend to domains such as machine translation, content verification, and adaptive information filtering, aligning with the broader goals of the research community in AI to develop intelligent systems capable of robust decision-making in complex, non-stationary settings.

The main limitation of our dynamic fusion framework lies in the English-centric dataset used to train it, which may not adequately represent the linguistic diversity of AI-generated text in multilingual contexts, thus limiting its applicability to non-English languages. Additionally, the integration of multiple components, such as convolutional neural networks, bidirectional LSTM, local and global attention mechanisms, and elastic weight consolidation, introduces significant computational complexity, potentially restricting its deployment in resource-constrained environments.

Future research will focus on enhancing our framework's performance with multilingual datasets to improve detection across diverse linguistic contexts where misuse of AI-generated text causes problems (e.g., in education, etc.) This will involve tailoring the model to language-specific patterns and evaluating it on multilingual benchmarks. Additionally, the framework will be expanded to accommodate multimodal GenAI content, including text and images, to address emerging challenges in dynamic, multi-format information fusion environments.

Acknowledgements

This research was supported by multiple higher education institutions. For the purpose of Open Access, the authors have applied a CC BY license to any Author Accepted Manuscript (AAM) version arising from this submission.

Availability of data and materials

We used publicly available datasets, which are appropriately cited in the article.

Code Availability

This is ongoing research with future expansion plans. The code will be made available on a reputable public platform. In the meantime, the code developed for this article is available upon reasonable request from the corresponding author.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this article.

Appendix A. Text length analysis

Table A.14: Text Length Statistics for `DeepfakeTextDetect` and `TuringBench` Datasets

Dataset	**Sample Size**	**Mean Length (Tokens)**	**% Samples $\leq$ 512 tokens**	**95% Percentile Length (Tokens)**
DeepfakeTextDetect	141, 000	264	93.8%	421
TuringBench	200, 000	231	95.4%	378

Table A.15: Ablation Study on Maximum Input Length

Model	**Max Tokens**	**DeepfakeTextDetect**	**TuringBench**
DistilBERT (Ours)	512	94.1%	92.7%
Longformer	1,024	94.2%	92.8%

Table A.16: Efficiency Comparison between DistilBERT and Longformer

Model	**Max Tokens**	**GPU Memory Usage**	**Latency (per sample)**	**Accuracy Change**
DistilBERT(our)	512	1.7 GB	1.2 ms	-
Longformer	1,024	6.4 GB	3.9 ms	+0.1 pp

Appendix B. Extended performance tables

Table B.17: Performance of the detection model across different language models in `TuringBench` dataset.

Text Generator	Overall performance		AI-generated text			Human-written text		
	Accuracy	**AUC**	**Precision**	**Recall**	**F1-Score**	**Precision**	**Recall**	**F1-Score**
GPT-1	99.35%	99.36%	99%	99%	99%	99%	99%	99%
GPT-2 (small)	96.80%	96.81%	96%	98%	97%	98%	96%	97%
GPT-2 (medium)	96.28%	96.29%	97%	96%	96%	96%	97%	96%
GPT-2 (large)	96.22%	96.18%	96%	97%	96%	97%	95%	96%
GPT-2 (xl)	95.72%	95.66%	95%	97%	96%	96%	95%	95%
GPT-2 (pytorch)	95.81%	95.76%	98%	93%	96%	94%	99%	96%
GPT-3	95.76%	95.71%	95%	97%	96%	97%	94%	96%
GROVER (base)	97.99%	97.99%	98%	98%	98%	98%	98%	98%
GROVER (large)	98.08%	97.99%	97%	100%	98%	99%	96%	98%
GROVER (mega)	97.97%	97.93%	97%	99%	98%	99%	97%	98%
CTRL	98.56%	98.53%	98%	99%	99%	99%	98%	98%
XLM	99.83%	99.83%	100%	100%	100%	100%	100%	100%
XLNet (base)	99.27%	99.28%	99%	99%	99%	99%	99%	99%
XLNet (large)	99.70%	99.70%	100%	100%	100%	100%	100%	100%
FAIR (WMT19)	91.28%	91.26%	92%	92%	92%	91%	91%	91%
FAIR (WMT20)	91.38%	91.45%	94%	89%	91%	89%	94%	91%
Transformer-XL	98.42%	98.41%	98%	99%	98%	99%	98%	98%
PLM (distill)	97.60%	97.60%	97%	98%	98%	98%	97%	98%
PLM (GPT-2)	96.42%	96.45%	95%	98%	96%	98%	95%	96%

Table B.18: Performance of the detection model across different language models in `DeepfakeTextdetect` dataset.

Text Generator	Overall performance		AI-generated text			Human-written text		
	Accuracy	**AUC**	**Precision**	**Recall**	**F1-Score**	**Precision**	**Recall**	**F1-Score**
text-davinci-002	84.54%	85.25%	64%	87%	73%	95%	84%	89%
text-davinci-003	89.44%	90.31%	73%	92%	81%	97%	89%	93%
gpt-3.5-trubo	93.73%	94.23%	83%	95%	89%	98%	93%	96%
LLaMA-7B	88.24%	85.59%	66%	81%	73%	95%	90%	92%
LLaMA-13B	88.43%	86.29%	67%	83%	74%	95%	90%	93%
LLaMA-30B	86.71%	84.06%	62%	80%	70%	95%	88%	91%
LLaMA-65B	88.57%	81.62%	72%	70%	71%	93%	93%	93%
GLM130B	83.17%	82.98%	82%	87%	84%	85%	78%	82%
FLAN-T5-small	92.95%	86.48%	82%	76%	79%	95%	96%	96%
FLAN-T5-base	91.47%	85.42%	74%	76%	75%	95%	95%	95%
FLAN-T5-large	91.20%	84.60%	72%	75%	74%	95%	94%	95%
FLAN-T5-xl	92.47%	80.21%	89%	62%	73%	93%	99%	96%
FLAN-T5-xxl	94.91%	87.44%	93%	76%	84%	95%	99%	97%
GPT-J-6B	89.85%	90.63%	81%	93%	86%	96%	88%	92%
GPT-NeoX-20B	96.25%	95.33%	96%	93%	94%	97%	98%	97%
OPT-125M	94.25%	84.84%	70%	73%	71%	97%	97%	97%
OPT-350M	96.01%	92.26%	76%	88%	81%	99%	97%	98%
OPT-1.3B	92.06%	86.57%	58%	80%	67%	98%	93%	95%
OPT-2.7B	96.11%	89.68%	81%	82%	81%	98%	98%	98%
OPT-6.7B	92.53%	85.40%	61%	76%	68%	97%	94%	96%
OPT-13B	98.38%	93.54%	94%	88%	91%	99%	99%	99%
OPT-30B	96.73%	85.37%	95%	71%	81%	97%	100%	98%
OPT-iml-max-1.3B	92.38%	82.51%	62%	70%	66%	96%	95%	96%
OPT-iml-30B	95.56%	82.38%	88%	66%	75%	96%	99%	98%
BLOOM-7B	92.01%	91.47%	80%	90%	85%	97%	93%	95%
T0-3B	91.85%	91.25%	80%	90%	85%	96%	92%	94%
T0-11B	92.29%	90.74%	82%	88%	85%	96%	94%	95%

References

[1] Abdullah, M., Hongying, Z., Javed, A., Mamyrbayev, O., Caraffini, F., Eshkiki, H., 2025. A joint learning framework for fake news detection. Displays, 103154.

[2] Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al., 2023. Gpt-4 technical report. arXiv preprint arXiv:2303.08774.

[3] Aguilera-Martos, I., Herrera-Poyatos, A., Luengo, J., Herrera, F., 2024. Local attention mechanism: Boosting the transformer architecture for long-sequence time series forecasting. arXiv preprint arXiv:2410.03805.

[4] Antoun, W., Mouilleron, V., Sagot, B., Seddah, D., 2023. Towards a robust detection of language model generated text: is chatgpt that easy to detect? arXiv preprint arXiv:2306.05871.

[5] Arase, Y., Zhou, M., 2013. Machine translation detection from monolingual web-text. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 1597–1607.

[6] Baayen, R. H., 2001. Word frequency distributions. Vol. 18. Springer Science & Business Media.

[7] Bakhtin, A., Gross, S., Ott, M., Deng, Y., Ranzato, M., Szlam, A., 2019. Real or fake? learning to discriminate machine from human generated text. arXiv preprint arXiv:1906.03351.

[8] Bao, G., Zhang, Y., Wang, Z., Li, Y., 2024. Drift-aware continual learning for ai-generated text detection. Proceedings of NeurIPS 2024.

[9] Beltagy, I., Peters, M. E., Cohan, A., 2020. Longformer: The long-document transformer. arXiv preprint arXiv:2004.05150.

[10] Bengio, Y., Hinton, G., Yao, A., Song, D., Abbeel, P., Darrell, T., Harari, Y. N., Zhang, Y.-Q., Xue, L., Shalev-Shwartz, S., et al., 2024. Managing extreme ai risks amid rapid progress. Science 384 (6698), 842–845.

[11] Bhattacharjee, A., Liu, H., mar 2024. EAGLE: A domain generalization framework for ai-generated text detection. arXiv.
URL https://arxiv.org/abs/2403.15690

[12] Breiman, L., 2001. Random forests. Machine learning 45, 5–32.

[13] Chen, Y., 2015. Convolutional neural network for sentence classification. Master's thesis, University of Waterloo.

[14] Chen, Y., Kang, H., Zhai, V., Li, L., Singh, R., Raj, B., 2023. Gpt-sentinel: Distinguishing human and chatgpt generated content. arXiv preprint arXiv:2305.07969.

[15] Cheng, Z., Zhou, L., Jiang, F., Wang, B., Li, H., April 2025. Beyond binary: Towards fine-grained llm-generated text detection via role recognition and involvement measurement. Proceedings of the ACM Web Conference 2025 (WWW '25).
URL https://arxiv.org/abs/2410.14259

[16] Corston-Oliver, S., Gamon, M., Brockett, C., 2001. A machine learning approach to the automatic evaluation of machine translation. In: Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics. pp. 148–155.

[17] Croce, D., Zelenkauskaite, A., Castellucci, G., 2020. GAN-BERT: Generative adversarial learning for bert-based text classification. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 432–443.

[18] Devlin, J., Chang, M.-W., Lee, K., Toutanova, K., 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). pp. 4171–4186.

[19] Devlin, J., Chang, M.-W., Lee, K., Toutanova, K., 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. Proceedings of NAACL-HLT 2019, 4171–4186.

[20] Fagni, T., Falchi, F., Gambini, M., Martella, A., Tesconi, M., 2021. Tweepfake: About detecting deepfake tweets. Plos one 16 (5), e0251415.

[21] Fu, J., Zhang, Y., Tan, M., Lin, C., Lee, S., Schicchi, D., Bollegala, D., Wang, Y., He, P., sep 2025. DetectAnyLLM: Towards generalizable and

robust detection of machine-generated text across domains and models. arXiv.
URL https://arxiv.org/abs/2509.14268

[22] Gallé, M., Rozen, J., Kruszewski, G., Elsahar, H., 2021. Unsupervised and distributional detection of machine-generated text. arXiv preprint arXiv:2111.02878.

[23] Gambini, M., Fagni, T., Falchi, F., Tesconi, M., 2022. On pushing deepfake tweet detection capabilities to the limits. In: Proceedings of the 14th ACM Web Science Conference 2022. pp. 154–163.

[24] Gehman, S., Gururangan, S., Sap, M., Choi, Y., Smith, N. A., 2020. Realtoxicityprompts: Evaluating neural toxic degeneration in language models. Findings of ACL: EMNLP 2020.

[25] Gehrmann, S., Strobelt, H., Rush, A. M., 2019. Gltr: Statistical detection and visualization of generated text. arXiv preprint arXiv:1906.04043.

[26] Graves, A., Fernández, S., Schmidhuber, J., 2005. Bidirectional lstm networks for improved phoneme classification and recognition. In: International conference on artificial neural networks. Springer, pp. 799–804.

[27] Guo, B., Zhang, X., Wang, Z., Jiang, M., Nie, J., Ding, Y., Yue, J., Wu, Y., 2023. How close is chatgpt to human experts? comparison corpus, evaluation, and detection. arXiv preprint arXiv:2301.07597.

[28] Guo, B., Zhang, X., Wang, Z., Liu, M., Nie, J., Ding, Y., Zhang, J., Cao, H., jan 2023. How close is chatgpt to human experts? comparison corpus, evaluation, and detection. arXiv.
URL https://arxiv.org/abs/2301.07597

[29] Hamed, A. A., Wu, X., 2023. Improving detection of chatgpt-generated fake science using real publication text: Introducing xfakebibs a supervised-learning network algorithm.

[30] He, P., Liu, X., Gao, J., Chen, W., 2022. Multi-scale feature fusion for text classification and detection. Findings of EMNLP 2022.

[31] Ippolito, D., Kriz, R., Sedghi, H., Awadalla, H., Kiela, D., 2019. Automatic detection of generated text: Methods, datasets, and evaluations. arXiv preprint arXiv:1911.00693.

[32] Jacovi, A., Shalom, O. S., Goldberg, Y., 2018. Understanding convolutional neural networks for text classification. arXiv preprint arXiv:1809.08037.

[33] Jha, S., Chaurasia, A., Sudhakar, A., Singh, A. K., 2017. Reference scope identification for citances using convolutional neural networks. In: Proceedings of the 14th international conference on natural language processing (ICON-2017). pp. 23–32.

[34] Joulin, A., Grave, E., Bojanowski, P., Mikolov, T., April 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. Association for Computational Linguistics, pp. 427–431.

[35] Kalinichenko, L. A., Korenkov, V. V., Shirikov, V. P., Sissakian, A. N., Sunturenko, O. V., 2003. Digital libraries: Advanced methods and technologies, digital collections. D-Lib Magazine 9 (1), 1082–9873.

[36] Kelly, M. L., Wood, P., 2023. 'everybody is cheating': Why this teacher has adopted an open chatgpt policy. Accessed: 2023-01-26.
URL https://www.npr.org/2023/01/26/1151499213/chatgpt-ai-education-cheating-classroom-wharton-school

[37] Kelly, S. M., 2023. Chatgpt passes exams from law and business schools. Accessed: 2024-01-26.
URL https://www.cnn.com/2023/04/13/tech/chatgpt-exams-law-business-schools/index.html

[38] Kemker, R., McClure, M., Abitino, A., Hayes, T., Kanan, C., 2018. Measuring catastrophic forgetting in neural networks. In: Proceedings of the AAAI conference on artificial intelligence. Vol. 32.

[39] Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al., 2017. Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences 114 (13), 3521–3526.

[40] Kumarage, T., Garland, J., Bhattacharjee, A., Trapeznikov, K., Ruston, S., Liu, H., 2024. Stylometric detection of ai-generated text in twitter timelines. arXiv preprint arXiv:2303.03697.

[41] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., Jackel, L. D., 1989. Backpropagation applied to handwritten zip code recognition. Neural computation 1 (4), 541–551.

[42] Li, Y., Li, Q., Cui, L., Bi, W., Wang, L., Yang, L., Shi, S., Zhang, Y., 2023. Deepfake text detection in the wild. arXiv preprint arXiv:2305.13242.

[43] Li, Y., Li, Q., Cui, L., Bi, W., Wang, Z., Wang, L., Yang, L., Shi, S., Zhang, Y., 2023. Mage: Machine-generated text detection in the wild. arXiv preprint arXiv:2305.13242.

[44] Li, Z., Wu, Y., Zhang, R., Wang, H., 2024. Hierarchical multi-scale fusion for robust ai text detection. Proceedings of ACL 2024.

[45] Liu, Y., Li, J., Zhang, R., Wang, D., 2023. Discourse-aware detection of ai-generated long-form text. Proceedings of EMNLP 2023.

[46] 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. arXiv preprint arXiv:1907.11692.

[47] 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. arXiv preprint arXiv:1907.11692.

[48] Liu, Y., Shao, Z., Hoffmann, N., 2021. Global attention mechanism: Retain information to enhance channel-spatial interactions. arXiv preprint arXiv:2112.05561.

[49] Liu, Y., Zhang, Z., Zhang, W., Yue, S., Zhao, X., Cheng, X., Zhang, Y., Hu, H., 2023. Argugpt: evaluating, understanding and identifying argumentative essays generated by gpt models. arXiv preprint arXiv:2304.07666.

[50] Liu, Z., Yao, Z., Li, F., Luo, B., 2023. Check me if you can: Detecting chatgpt-generated academic writing using checkgpt. arXiv preprint arXiv:2306.05524 2.

[51] McCloskey, M., Cohen, N. J., 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In: Psychology of learning and motivation. Vol. 24. Elsevier, pp. 109–165.

[52] Mitchell, A., 2022. Professor catches student cheating with chatgpt:'i feel abject terror'. new york post.

[53] Mitchell, E., Lee, Y., Khazatsky, A., Manning, C. D., Finn, C., 2023. Detectgpt: Zero-shot machine-generated text detection using probability curvature. In: International Conference on Machine Learning. PMLR, pp. 24950–24962.

[54] Mitchell, E., Lee, Y., Khazatsky, A., Manning, C. D., Finn, C., 2023. Detecting textual adversarial examples through the lens of uncertainty. arXiv preprint arXiv:2304.11268.

[55] Opara, C., 2024. Styloai: Distinguishing ai-generated content with stylometric analysis. In: International conference on artificial intelligence in education. Springer, pp. 105–114.

[56] Pagnoni, A., Graciarena, M., Tsvetkov, Y., 2022. Threat scenarios and best practices to detect neural fake news. In: Proceedings of the 29th International Conference on Computational Linguistics. pp. 1233–1249.

[57] Pu, Y., Wang, L., Zhang, X., Chen, J., 2023. Continual learning for cross-domain ai text detection. Proceedings of ACL 2023.

[58] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al., 2019. Language models are unsupervised multitask learners. OpenAI blog 1 (8), 9.

[59] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P. J., 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of machine learning research 21 (140), 1–67.

[60] Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T., Wayne, G., 2019. Experience replay for continual learning. Proceedings of NeurIPS 2019.

[61] Sanh, V., Debut, L., Chaumond, J., Wolf, T., 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108.

[62] Sarvazyan, A. M., González, J. Á., Rosso, P., Franco-Salvador, M., 2023. Supervised machine-generated text detectors: Family and scale matters. In: International Conference of the Cross-Language Evaluation Forum for European Languages. Springer, pp. 121–132.

[63] Schuster, T., Schuster, R., Shah, D. J., Barzilay, R., 2020. The limitations of stylometry for detecting machine-generated fake news. Computational Linguistics 46 (2), 499–510.

[64] Schwarz, J., Vinyals, O., Doulamis, K., Kavukcuoglu, K., Blundell, C., 2018. Progressive neural networks. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. pp. 132–140.

[65] Shah, A., Ranka, P., Dedhia, U., Prasad, S., Muni, S., Bhowmick, K., 2023. Detecting and unmasking ai-generated texts through explainable artificial intelligence using stylistic features. International Journal of Advanced Computer Science and Applications 14 (10).

[66] Sison, A. J. G., Daza, M. T., Gozalo-Brizuela, R., Garrido-Merchán, E. C., 2024. Chatgpt: More than a "weapon of mass deception" ethical challenges and responses from the human-centered artificial intelligence (hcai) perspective. International Journal of Human–Computer Interaction 40 (17), 4853–4872.

[67] Solaiman, I., Brundage, M., Clark, J., Askell, A., Herbert-Voss, A., Wu, J., Radford, A., Krueger, G., Kim, J. W., Kreps, S., et al., 2019. Release strategies and the social impacts of language models. arXiv preprint arXiv:1908.09203.

[68] Stamatatos, E., 2009. A survey of modern authorship attribution methods. Journal of the American Society for information Science and Technology 60 (3), 538–556.

[69] Tang, R., Liu, J., Zhang, Z., Li, Z., 2023. Token-level temporal signatures in llm-generated text. arXiv preprint arXiv:2306.14567.

[70] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al., 2023. Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971.

[71] Uchendu, A., Le, T., Shu, K., Lee, D., 2020. Authorship attribution for neural text generation. In: Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP). pp. 8384–8395.

[72] Uchendu, A., Ma, Z., Le, T., Zhang, R., Lee, D., 2021. Turingbench: A benchmark environment for turing test in the age of neural text generation. arXiv preprint arXiv:2109.13296.

[73] Vapnik, V., 1995. Support-vector networks. Machine learning 20, 273–297.

[74] Verga, P., Drozdov, A., McCallum, A., 2021. Non-stationary text generation in autoregressive models. Proceedings of NAACL 2021.

[75] Wang, F., Zhang, L., Chen, H., Liu, J., 2025. Lifelong learning for evolving ai text detectors: A multi-scale approach. Proceedings of AAAI 2025.

[76] Wang, Z., Cheng, J., Cui, C., Yu, C., 2023. Implementing bert and fine-tuned roberta to detect ai generated news by chatgpt. arXiv preprint arXiv:2306.07401.

[77] Yang, K., Chen, H., Liu, J., Zhang, L., 2025. Universal linguistic markers in large language model outputs. Proceedings of ICML 2025.

[78] Zhan, R., Gao, X., Wang, T., Xie, Y., Pan, Y., Yu, Y., Qiao, D., Liu, P., Liu, Y., oct 2024. DetectRL: Benchmarking llm-generated text detection in real-world scenarios. arXiv.
URL https://arxiv.org/abs/2410.23746

[79] Zhang, L., Zhao, Y., Wang, C., Liu, X., 2024. Temporal change detection in ai-generated text using multi-scale fusion. Proceedings of CVPR 2024 Workshops.

[80] Zhong, Y., Li, J., Chen, Q., Liu, Y., 2022. Multi-granular text analysis for ai detection via transformer fusion. Proceedings of COLING 2022.