Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Explainable Hybrid Deep Learning for AI-generated Text Detection

Anjani Kumar

akumar@cic.du.ac.in

University of Delhi

Article

Keywords: AI-generated text, Deep learning, LSTM, Transformer, CNN, SHAP, LIME, Explainable AI

Posted Date: November 26th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11278494/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License. Read Full License

Additional Declarations: No competing interests reported.

Abstract

The rapid adoption of large language models (LLMs) has increased the need for reliable methods to distinguish human-written from AI-generated text. Although deep-learning detectors can achieve strong predictive performance, their decisions are often difficult to interpret. This study develops an explainable hybrid architecture combining bidirectional long short-term memory (BiLSTM), Transformer self-attention, and one-dimensional convolutional neural network (1D-CNN) layers. SHAP and LIME are incorporated to provide global and instance-level explanations. Using a paired corpus of human-written and AI-generated texts, the model is evaluated on a held-out test set of 15,000 samples. The reported results show 99.15% accuracy and a ROC AUC of 0.9991, with precision, recall, and F1-score of approximately 0.99 for both classes. The findings indicate strong discrimination on the evaluated dataset while providing an interpretable analysis layer. However, cross-dataset and adversarial generalization are not established and require further validation.

1. INTRODUCTION

The rapid development of large language models (LLMs) has transformed the production and dissemination of textual information. Systems such as GPT-class models can generate fluent and coherent content at scale, creating opportunities in education, communication, software development, and content production. At the same time, the increasing availability of machine-generated text creates a need for reliable methods that can assess text provenance, particularly in settings where authenticity and accountability matter.

AI-generated text detection is challenging because the linguistic differences between human and machine writing are neither fixed nor universal. Detection performance may vary with the topic, writing domain, generation model, prompting strategy, decoding procedure, and post-processing applied to generated text. Recent studies have demonstrated strong performance under controlled conditions while also showing substantial degradation under distribution shift and adversarial perturbations (Mitchell et al., 2023; Verma et al., 2024; Dugan et al., 2024). These observations motivate models that learn complementary textual representations and are evaluated with explicit attention to generalization.

Hybrid neural architectures provide one possible solution because different components can model different aspects of language. Recurrent networks can capture sequential dependencies, Transformer attention can model contextual relationships over longer spans, and convolutional layers can identify local lexical and stylistic patterns. However, predictive accuracy alone is insufficient when a detector is used to support consequential decisions. Explainable artificial intelligence (XAI) methods such as LIME and SHAP can provide complementary local and feature-attribution views of model behavior (Ribeiro et al., 2016; Lundberg and Lee, 2017).

This study therefore develops and evaluates a hybrid detector that integrates BiLSTM, Transformer attention, and 1D-CNN components, together with SHAP and LIME explanations. The contribution is twofold: first, the study combines recurrent, attention-based, and convolutional representations in a single detection pipeline; second, it adds an explicit interpretability layer intended to clarify the linguistic evidence associated with individual predictions. The evaluation is performed on a held-out test partition of a paired human/AI corpus. Cross-dataset transferability and adversarial robustness are treated as limitations rather than claimed outcomes of the present experiment.

2. RELATED WORK

Early approaches to machine-generated text detection relied on statistical and stylistic cues, including lexical frequency, syntactic regularity, and language-model characteristics (Badaskar et al., 2008). As language models became more capable, learned detectors and neural classifiers became increasingly important. Bidirectional recurrent architectures have been effective for text classification because they capture information from both preceding and following context (Zhou et al., 2016), while Transformer architectures introduced attention mechanisms that provide a flexible way to model contextual dependencies (Vaswani et al., 2017).

More recent work has explored detection without requiring direct access to the generator. DetectGPT uses probability curvature as a zero-shot detection signal (Mitchell et al., 2023), whereas Ghostbuster combines features derived from multiple language models and evaluates generalization across domains and generation settings (Verma et al., 2024). Watermarking represents a different strategy in which detectable signals are deliberately embedded during generation (Kirchenbauer et al., 2023). These approaches demonstrate that detection is possible through multiple mechanisms, but they also emphasize that evaluation conditions strongly affect reported performance. RAID, a large shared benchmark, found that detectors can be vulnerable to changes in generation strategies, unseen models, and adversarial attacks (Dugan et al., 2024).

3. METHODOLOGY

The proposed methodology uses a hybrid neural architecture that combines recurrent, attention-based, and convolutional components to learn complementary representations of human- and AI-generated text. The pipeline consists of text preprocessing, tokenization and embedding, BiLSTM sequence modeling, Transformer-based contextual feature extraction, 1D-CNN local feature extraction, classification, and post-hoc explanation using SHAP and LIME.

Model architecture

The model is designed specifically for binary text classification. Unlike the unrelated tabular/medical-dataset formulation present in the earlier draft, the architecture here operates directly on tokenized text sequences. The selected components are intended to capture sequential dependencies, long-range contextual relationships, and local n-gram-like patterns within the same representation pipeline.

3.1.1. Text vectorization and embedding
Input texts are tokenized using TensorFlow's TextVectorization layer with a vocabulary size of 20,000 and a maximum sequence length of 1,024 tokens. Sequences exceeding this limit are truncated and shorter sequences are padded. Each token is mapped to a 64-dimensional trainable embedding representation. Let V denote the vocabulary size and d the embedding dimension; the embedding matrix therefore has dimensions V × d, where V = 20,000 and d = 64.

$$(1)$$

The embedding representation converts discrete token identifiers into continuous vectors that can be processed by the subsequent sequence-modeling layers.
3.1.2. Bidirectional LSTM

A bidirectional LSTM (BiLSTM) is used to capture contextual dependencies in both temporal directions. For an input sequence, the forward LSTM processes tokens from left to right, while the backward LSTM processes them from right to left. The two hidden states are concatenated at each time step to produce a context-enriched representation. This design allows the model to use information from both preceding and following tokens when classifying a sequence.

$$\begin{aligned} \overrightarrow{\varvec{h}_t} &= \mathrm{LSTM}_f(\varvec{e}_t, \overrightarrow{\varvec{h}_{t-1}}), \\ \overleftarrow{\varvec{h}_t} &= \mathrm{LSTM}_b(\varvec{e}_t, \overleftarrow{\varvec{h}_{t-1}}), \end{aligned} \tag{2}$$

The concatenated representation is passed to the Transformer block for further contextual modeling.

$$\varvec{h}_t = \left[\overrightarrow{\varvec{h}_t}; \overleftarrow{\varvec{h}_t}\right]. \tag{3}$$

This representation provides contextual information from both directions of the input sequence.

3.1.3. Transformer block

Long-range dependencies are modeled using a Transformer block with multi-head self-attention. For queries Q, keys K, and values V, scaled dot-product attention is defined by the standard attention formulation (Vaswani et al., 2017). The model uses two attention heads, followed by residual connections and layer normalization. A feed-forward sublayer with dimension 64 further transforms the contextual representation.

$$\mathrm{Attention}\left(Q, K, V\right) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{4}$$

Multi-head attention combines the outputs of the individual attention heads and projects the concatenated representation into the model dimension.

$$\mathrm{MultiHead}\left(X\right) = \mathrm{Concat}\left(\mathrm{head}_1, \ldots, \mathrm{head}_h\right)\varvec{W}^O. \tag{5}$$

With two attention heads and a key/query dimension of 64, the block uses residual connections and layer normalization to improve optimization and stabilize the learned representation.

$$\begin{aligned} X_1 &= \mathrm{LayerNorm}(X + \mathrm{MultiHead}(X)), \\ X_2 &= \mathrm{LayerNorm}(X_1 + \mathrm{FFN}(X_1)). \end{aligned} \tag{6}$$

3.1.4. Convolutional feature extraction

A one-dimensional convolutional layer with 128 filters and a kernel size of 3 is applied to the Transformer output. The convolutional operation is intended to capture short-range lexical, syntactic, and stylistic patterns that may complement the longer-range dependencies learned by the attention mechanism.

$$\varvec{c}_i = \mathrm{ReLU}\left(\varvec{W} \cdot \varvec{x}_{i:i+k-1} + \varvec{b}\right). \tag{7}$$

3.1.5. Pooling and output layer

Global max pooling is applied to the convolutional feature maps to retain the most salient activation from each filter. The pooled vector is then passed to a fully connected layer with 128 ReLU units.

$$\varvec{p} = \max_i \left(\varvec{c}_i \right). \tag{8}$$

A dropout rate of 0.5 is applied after the dense layer to reduce overfitting. The final layer uses a sigmoid activation to estimate the probability that the input text belongs to the AI-generated class.

$$\varvec{z} = \mathrm{ReLU}\left(\varvec{W}_d \varvec{p} + \varvec{b}_d \right), \tag{9}$$

followed by dropout (rate 0.5). The final sigmoid output ŷ (the probability of being AI-generated) is

$$\hat{y} = \sigma\left(\varvec{w}_o^T \varvec{z} + b_o \right). \tag{10}$$

Training procedure

The model is optimized using Adam with an initial learning rate of 0.001 and binary cross-entropy loss. Training is performed for up to 10 epochs with a batch size of 32. Early stopping with a patience of three epochs, learning-rate reduction by a factor of 0.2, validation-based checkpointing, and dropout regularization are used to improve generalization.

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \left[y_i \log\left(\hat{y}_i\right) + \left(1 - y_i\right) \log\left(1 - \hat{y}_i\right) \right]. \tag{11}$$

The reported implementation uses TensorFlow 2.13 + and GPU acceleration. The tf.data pipeline uses caching, parallel mapping, and prefetching to improve data throughput. Experiment configuration is maintained through a JSON-based specification, while TensorBoard is used for monitoring training behavior.

Explainability Framework

SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations) are used as post-hoc explanation methods. SHAP assigns an attribution value to each feature based on its contribution to the model output (Lundberg and Lee, 2017). For text classification, token-level attributions can be aggregated to identify features that move a prediction toward either class. LIME explains an individual prediction by generating perturbed versions of the input and fitting a locally interpretable surrogate model around the original instance (Ribeiro et al., 2016).

$$\phi_i = \sum_{S \subseteq F\{i\}} \frac{|S|!\left(|F| - |S| - 1\right)!}{|F|!} \left[f\left(S \cup \{i\}\right) - f\left(S\right) \right], \tag{12}$$

In this study, SHAP is used to provide feature-attribution information and LIME is used to provide instance-level explanations. Because both methods are post-hoc approximations, their outputs should be interpreted as explanations of model behavior rather than as direct evidence of causal linguistic features.

$$\xi\left(x\right) = \underset{g \in G}{\mathrm{argmin}} \mathcal{L}\left(f, g, \pi_x\right) + \Omega\left(g\right), \tag{13}$$

4. RESULTS AND DISCUSSION

Dataset

The study uses a paired corpus containing human-written and AI-generated text. The data are divided into training, validation, and testing partitions. The held-out test partition contains 15,000 samples: 9,281 human-written samples and 5,719 AI-generated samples. All texts undergo normalization, tokenization, and fixed-length transformation using the TensorFlow TextVectorization pipeline with a vocabulary size of 20,000 and a maximum sequence length of 1,024. The supplied manuscript does not document the dataset name, source domains, generating models, prompting procedure, or deduplication protocol; these details should be added before submission because they are important for reproducibility and for interpreting generalization claims.

Table 1
Dataset Statistics

Split	Total Samples	Human (0)	AI (1)
Training	49,999	31,369	18630
Validation	15,000	9,285	5,715
Testing	15,000	9,281	5,719
Implementation details			

The final implementation uses a 64-dimensional embedding layer, a BiLSTM with 32 units per direction, a Transformer block with two attention heads and a feed-forward dimension of 64, a 1D-CNN with 128 filters and kernel size 3, and a 128-unit dense projection. Dropout is set to 0.5. The vocabulary size and maximum sequence length are 20,000 and 1,024, respectively.

Training configuration is maintained in a JSON-based experiment specification. TensorBoard is used to monitor loss convergence, gradient behavior, and internal activations. These settings are intended to support reproducibility, although the exact random seed, hardware specification, and software package versions beyond TensorFlow 2.13 + should also be reported in the final submission.

Table 2
Model Hyperparameters

Parameter	Value
Embedding Dimension	64
LSTM Units (per direction)	32
Transformer Heads	2
Feed-forward Dimension	64
CNN Filters	128
CNN Kernel Size	3
Dense Units	128
Dropout Rate	0.5
Vocabulary Size	20,000
Sequence Length	1,024
Training and evaluation protocol	

The model is trained with Adam at an initial learning rate of 10^-3 and a batch size of 32 using binary cross-entropy loss. Early stopping with a patience of three epochs and learning-rate reduction by a factor of 0.2 are applied when validation performance stops improving. The best checkpoint is selected using validation accuracy.

Final performance is reported on the separate 15,000-sample test set. The evaluation includes accuracy, precision, recall, F1-score, ROC AUC, and the confusion matrix. No statistical significance test is reported in the supplied experimental results; therefore, claims of statistical significance have been removed.

Quantitative results

On the held-out test set, the hybrid model achieves an overall accuracy of 99.15% and a ROC AUC of 0.9991. Precision, recall, and F1-score are approximately 0.99 for both the human and AI-generated classes. These results indicate strong discrimination on the evaluated test distribution. They should not, however, be interpreted as evidence of cross-dataset generalization because the study evaluates a single dataset split.

Table 3
Test Set Performance Metrics

Metric	Class 0 (Human)	Class 1 (AI)
Precision	0.99	0.99
Recall	0.99	0.99
F1-Score	0.99	0.99
Support	9,281	5,719
Accuracy	0.9915	
ROC AUC	0.9991	

The confusion matrix further illustrates the classification behavior.

The model correctly classifies 9,224 human-written samples and 5,649 AI-generated samples. It misclassifies 57 human-written samples as AI-generated and 70 AI-generated samples as human-written. These counts yield 14,873 correct predictions out of 15,000 test instances, corresponding to approximately 99.15% accuracy.

Performance analysis

The two classes show similar reported precision, recall, and F1-score values, suggesting that the model does not exhibit a large class-specific performance imbalance on this test set. However, the test set itself is moderately imbalanced, with human-written samples representing approximately 61.9% of the data. Consequently, accuracy should be considered together with class-wise metrics and ROC AUC rather than used as the sole indicator of performance.

Model behavior interpretation

The architecture provides three complementary representation mechanisms. The BiLSTM can model sequential dependencies in both directions, the Transformer block can capture contextual relationships through self-attention, and the 1D-CNN can emphasize local patterns. The reported performance is consistent with the hypothesis that combining these mechanisms can produce a discriminative representation. Nevertheless, the present results do not isolate the contribution of each component; ablation experiments would therefore be required to determine whether the hybrid architecture significantly outperforms its individual components.

Explainability insights

SHAP and LIME are included to provide complementary explanations of model predictions. SHAP provides token-level attribution information that can be aggregated across samples, whereas LIME focuses on local explanations for individual texts. These explanations can be used to inspect whether predictions are driven by plausible linguistic patterns or by potentially spurious lexical cues.

The supplied results section does not provide quantitative aggregates of SHAP values, representative SHAP plots, or a systematic LIME case study. Accordingly, the present manuscript treats explainability as a methodological component rather than claiming a quantified explanation-quality improvement. For a stronger

submission, at least two representative correctly classified cases and one misclassified case should be presented with both SHAP and LIME explanations.

LIME can be particularly useful for examining borderline or misclassified instances because it shows how perturbing individual tokens changes the local decision boundary. Such examples should be reported with the original text, predicted probability, highlighted influential tokens, and a concise interpretation.

Interpretability benefits

The integration of SHAP and LIME improves the inspectability of individual predictions and can support model debugging and error analysis. In applications where automated detection may affect academic or professional judgments, such explanations can help users understand that a model prediction is probabilistic rather than a definitive determination of authorship. The explanations should therefore be used as supporting evidence rather than as proof of whether a human or an AI system produced a text.

Limitations and challenges

Several limitations remain. First, the evaluation is restricted to one dataset and one held-out test partition; cross-dataset transferability is therefore not demonstrated. Second, the supplied manuscript does not specify the source domains, LLM families, prompts, decoding settings, or post-processing procedures used to construct the corpus, which limits reproducibility and external validity. Third, no ablation study is reported, so the individual contributions of the BiLSTM, Transformer, and CNN components cannot be quantified. Fourth, SHAP and LIME add computational cost and provide post-hoc approximations rather than causal explanations. Finally, detector performance may change as generation models and prompting strategies evolve, and robustness to paraphrasing or adversarial attacks has not been evaluated.

Potential improvements

Future experiments should include cross-dataset evaluation, model-family holdout testing, domain transfer, paraphrase and adversarial stress tests, and ablation studies of the hybrid architecture. Threshold calibration should also be reported using validation data, with precision-recall curves and class-specific error analysis where appropriate. Reporting confidence intervals across multiple random seeds would provide a more reliable estimate of performance stability. For explainability, representative SHAP and LIME case studies should be added, including difficult and misclassified examples.

Comparative analysis

Direct numerical comparison with other detectors is not appropriate without matching datasets, generation settings, and evaluation protocols. Nevertheless, the present framework is conceptually aligned with prior work that emphasizes complementary representations and robust evaluation of AI-generated text detectors (Mitchell et al., 2023; Verma et al., 2024; Dugan et al., 2024). Its principal distinguishing feature is the explicit integration of post-hoc explanation methods into the hybrid classifier pipeline.

5. CONCLUSION

This study presents an explainable hybrid deep-learning framework for detecting AI-generated text. The model combines BiLSTM sequence modeling, Transformer self-attention, and 1D-CNN feature extraction, and incorporates SHAP and LIME to improve the interpretability of individual predictions.

On the reported held-out test set of 15,000 samples, the model achieves 99.15% accuracy and a ROC AUC of 0.9991, with approximately 0.99 precision, recall, and F1-score for both classes. These results demonstrate strong discriminative performance on the evaluated distribution. However, they do not establish robustness to unseen datasets, domains, language models, prompts, or adversarial modifications.

The main practical value of the proposed framework is therefore the combination of predictive modeling and model inspection. Future work should prioritize transparent dataset documentation, cross-dataset and cross-model validation, ablation studies, adversarial evaluation, and systematic reporting of SHAP and LIME explanations. Such extensions would provide stronger evidence for the reliability and practical deployment of AI-generated text detection systems.

Declarations

Funding:

No funding was received to assist with the preparation of this manuscript.

Data availability

The datasets analyzed during the current study are available upon author reasonable request.

Author Contribution

The author has contributed from stage 1 to the end of the stage of preparing this manuscript.

Acknowledgments

The authors acknowledge the individuals and institutions that contributed to the development and evaluation of this study.

Data Availability

The datasets analyzed during the current study are available upon author reasonable request.

References

1. Badaskar, S., Agarwal, S. & Arora, S. 'Identifying real or fake articles: Towards better language modeling', Proceedings of the Third International Joint Conference on Natural Language Processing, pp. 1–8. (2008).

2. Dugan, L. et al. 'RAID: A shared benchmark for robust evaluation of machine-generated text detectors', Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics, pp. 12011–12035. (2024).
3. Kirchenbauer, J. et al. 'A watermark for large language models', Proceedings of the 40th International Conference on Machine Learning, 202, pp. 17061–17084. (2023).
4. Lundberg, S. M. & Lee, S. I. 'A unified approach to interpreting model predictions'. *Adv. Neural. Inf. Process. Syst.*, 30. (2017).
5. Mitchell, E., Lee, Y., Khazatsky, A., Manning, C. D. & Finn, C. 'DetectGPT: Zero-shot machine-generated text detection using probability curvature', Proceedings of the 40th International Conference on Machine Learning, 202, pp. 24950–24962. (2023).
6. Ribeiro, M. T., Singh, S. & Guestrin, C. 'Why should I trust you? Explaining the predictions of any classifier', Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1135–1144. (2016).
7. Solaiman, I. et al. 'Release strategies and the social impacts of language models', arXiv preprint arXiv:1908.09203. (2019).
8. Vaswani, A. et al. 'Attention is all you need'. *Adv. Neural. Inf. Process. Syst.*, 30. (2017).
9. Verma, V., Fleisig, E., Tomlin, N. & Klein, D. 'Ghostbuster: Detecting text ghostwritten by large language models', Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics, pp. 81–97. (2024).
10. Zhou, P. et al. 'Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling', Proceedings of COLING 2016, pp. 3485–3495. (2016).

Figures

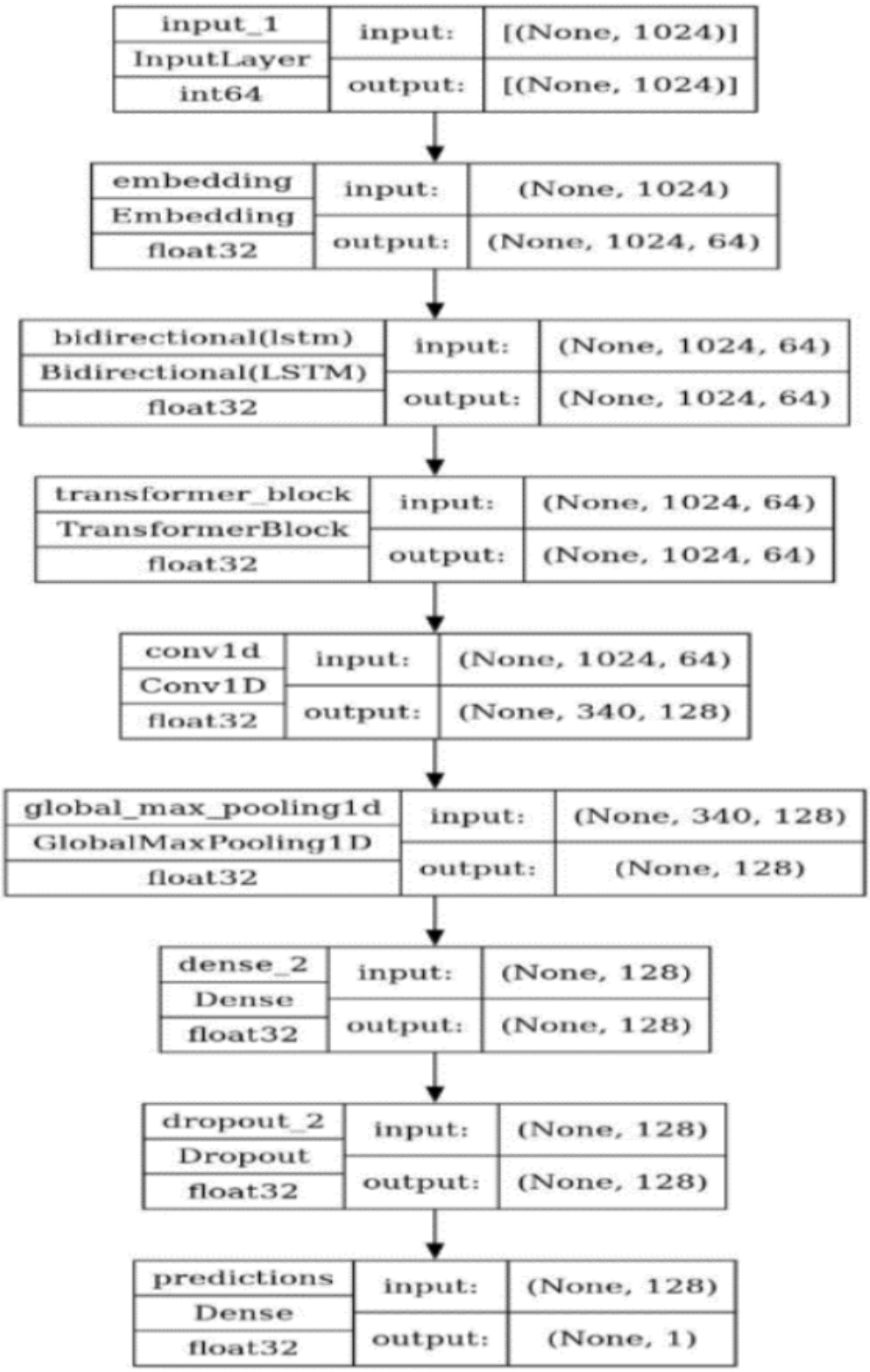

Figure 1

Architecture of the proposed hybrid model.