applied sciences

MDPI

Article

A Deep Learning Approach to Classify AI-Generated and Human-Written Texts

Ayla Kayabas [1], Ahmet Ercan Topcu [2,*], Yehia Ibrahim Alzoubi [3,*] and Mehmet Yıldız [1]

[1] Department of Computer Engineering, Faculty of Engineering and Architecture, Kirsehir Ahi Evran University, 40100 Kirsehir, Turkey; ayla.kayabas@ahievran.edu.tr (A.K.); mehmet.yildiz.jap@gmail.com (M.Y.)
[2] College of Engineering and Technology, American University of the Middle East, Egaila 54200, Kuwait
[3] College of Business Administration, American University of the Middle East, Egaila 54200, Kuwait
* Correspondence: ahmet.topcu@aum.edu.kw (A.E.T.); yehia.alzoubi@aum.edu.kw (Y.I.A.)

Abstract: The rapid advancement of artificial intelligence (AI) has introduced new challenges, particularly in the generation of AI-written content that closely resembles human-authored text. This poses a significant risk for misinformation, digital fraud, and academic dishonesty. While large language models (LLM) have demonstrated impressive capabilities across various languages, there remains a critical gap in evaluating and detecting AI-generated content in under-resourced languages such as Turkish. To address this, our study investigates the effectiveness of long short-term memory (LSTM) networks—a computationally efficient and interpretable architecture—for distinguishing AI-generated Turkish texts produced by ChatGPT from human-written content. LSTM was selected due to its lower hardware requirements and its proven strength in sequential text classification, especially under limited computational resources. Four experiments were conducted, varying hyperparameters such as dropout rate, number of epochs, embedding size, and patch size. The model trained over 20 epochs achieved the best results, with a classification accuracy of 97.28% and an F1 score of 0.97 for both classes. The confusion matrix confirmed high precision, with only 19 misclassified instances out of 698. These findings highlight the potential of LSTM-based approaches for AI-generated text detection in the Turkish language context. This study not only contributes a practical method for Turkish NLP applications but also underlines the necessity of tailored AI detection tools for low-resource languages. Future work will focus on expanding the dataset, incorporating other architectures, and applying the model across different domains to enhance generalizability and robustness.

Keywords: deep learning; AI-generated content; human-generated content; text generation

Academic Editor: Douglas O'Shaughnessy

Received: 12 April 2025
Revised: 7 May 2025
Accepted: 13 May 2025
Published: 15 May 2025

Citation: Kayabas, A.; Topcu, A.E.; Alzoubi, Y.I.; Yıldız, M. A Deep Learning Approach to Classify AI-Generated and Human-Written Texts. *Appl. Sci.* **2025**, *15*, 5541. https://doi.org/10.3390/app15105541

1. Introduction

While AI has been widely implemented in numerous beneficial ways and applications, it also has the potential to be misused, particularly in creating deceptive and misleading content. Among these concerns is the emergence of AI-generated text, which can produce fake or fabricated content that closely mimics human writing [1]. This poses significant challenges, as such text can be employed maliciously, including spreading misinformation, generating fake news, or even creating fraudulent documents. The focus on AI-generated text stems from its ability to accurately replicate linguistic patterns, often making it indistinguishable from human-written text [2]. This capability, while impressive, raises ethical concerns and necessitates the development of tools and methodologies to spot the misuse of AI-generated content [3].

The misuse of AI-generated content can lead to the spread of misinformation, manipulation of public opinion, academic dishonesty, and the erosion of trust in digital communication [4]. For instance, AI-generated texts can be used to create fake news articles, impersonate individuals online, or produce fraudulent academic work, which can have severe consequences for society at large [5]. These risks necessitate the development of effective tools and methodologies to identify and mitigate such misuse. From a social perspective, empowering institutions, educators, journalists, and the public with reliable AI detection tools fosters transparency, safeguards intellectual integrity, and promotes the responsible use of AI technologies [6]. Especially in low-resource language settings like Turkish, ensuring that communities are not disproportionately vulnerable to AI misuse is both an ethical and practical imperative [7]. Thus, this study contributes not only to technological development but also to the broader goal of maintaining ethical standards and social trust in the age of generative AI.

One subset of machine learning is deep learning [3]. It focuses on algorithms known as artificial neural networks, which are modeled after the composition and operations of the human brain [8]. As deep learning continues to evolve, LSTM networks remain a cornerstone in the toolkit of researchers and practitioners, driving innovation in diverse fields, from language processing to predictive analytics [9]. LSTM networks have demonstrated their versatility across a wide range of applications. In NLP, LSTM is employed for text classification, sentiment analysis, and machine translation [4]. In finance, LSTM is used for market trend prediction and price stocking. Their ability to process sequential data makes them suitable for speech recognition, handwriting recognition, and bioinformatics. Using LSTM networks to detect AI-generated text offers unique advantages since it can capture temporal dependencies, understand context, and handle non-linear patterns in sequential data. LSTMs can distinguish between fake and original texts by identifying unique patterns, such as unnatural transitions or repetitive structures [9]. Their adaptability to diverse text types and integration with pre-trained models enhances performance, making them particularly effective for natural language processing tasks. By leveraging these capabilities, LSTMs provide robust and reliable tools for highly precise text classification [2].

Consequently, this study aims to address the following research question: can LSTM-based models accurately detect AI-generated Turkish text, and what are their limitations and strengths in this context? This study mitigates the above constraints by creating advanced detection systems explicitly tailored for Turkish AI-generated text. Despite substantial advances in detecting AI-generated content, especially in English, the field still faces ongoing challenges, particularly regarding language diversity and generalization [10,11]. Many existing models focus exclusively on high-resource languages, leaving underrepresented languages like Turkish largely unaddressed. This lack of focus creates a pressing gap in the development of effective and equitable AI detection tools. The main contributions of this paper are as follows:

- The formulation of an LSTM-based detection framework tailored to the Turkish language;
- The development of a novel dataset comprising both human-written and ChatGPT-generated Turkish texts;
- A comprehensive experimental evaluation, varying key parameters such as dropout rate, epochs, embedding size, and patch size;
- The demonstration of high detection accuracy, offering practical implications for real-world applications in academia, media, and cybersecurity.

The remainder of this paper is structured as follows: Section 2 presents the research background and a review of the relevant literature. Section 3 covers the research methodology employed in this study. Section 4 details the experimental results and subsequent

discussion. Section 5 highlights the research limitations, future directions, and the implications of the findings. Finally, Section 6 provides the concluding remarks.

2. Background and Related Literature

2.1. Overview of LSTM Technique

Fundamentally, deep learning processes extract patterns from data by distinguishing input, hidden, and output layers by identifying levels of interdependent neurons or nodes [12]. The capacity of deep learning to learn hierarchical representations—each layer extracting progressively abstract features from the input data—is its main strength [13]. Deep learning has taken center stage in contemporary AI research and applications thanks to the availability of big datasets, sophisticated computing power, and frameworks like TensorFlow v2.18.0 and PyTorch v2.6.0 [8]. Unlike standard machine learning algorithms, deep learning techniques automatically extract raw data characteristics, eliminating the demand for laborious, detailed feature engineering. These capabilities have made significant progress in several fields, such as autonomous systems, image recognition, and natural language processing.

One type of recurrent neural network (RNN), called an LSTM network, is designed to process and forecast data sequences [14]. In contrast to conventional RNNs, which have issues with vanishing or ballooning gradients when working with lengthy sequences, LSTMs control the information flow using memory cells and gates (input, output, and forget gates) [15]. Because of their architecture, LSTMs can retain information over extended time steps, making them ideal for activities such as recognizing speech, language modeling, and time series forecasting [12]. LSTMs differ from other neural network architectures in their capacity to retain context across long sequences [16].

Understanding the context and word sequence is essential in natural language processing, one of the key areas where LSTMs are used. In machine translation, for example, an LSTM network can record a sentence's structure in one language and produce the same structure in another. Similarly, LSTMs can accurately convert audio signals into text in voice recognition by modeling the temporal connections of uttered words. They are a key component of many cutting-edge AI systems due to their resilience in managing sequential dependencies. Transformers, which utilize attention mechanisms, have largely supplanted LSTMs in many NLP tasks due to their efficiency and scalability [4]. However, LSTMs remain relevant in specific domains, primarily where computational resources are limited, or the data are naturally sequential and not excessively long [16].

2.2. Related Literature

The detection of AI-generated text has gained considerable research attention in recent years due to the growing concerns around authenticity, misinformation, and ethical misuse. Studies such as [17] have highlighted that while AI-generated text is advancing rapidly, it still lacks the nuanced coherence and complexity found in human-authored content. These limitations have led to a diverse set of detection approaches spanning traditional machine learning, deep learning, and hybrid methods. Table 1 offers a concise overview of previous research findings, emphasizing the distinct contributions of our study in addressing existing gaps. Collectively, these studies showcase the efficacy of various deep learning models in identifying AI-generated content.

Table 1. Findings from previous studies.

Category	Study	Focus	Deep Learning Technique Used	Findings
Using deep learning models	[15]	Proposed a new model for text classification.	DRNN	Test accuracy of 88.52% using a dataset of 900 samples.
	[14]	Combined existing datasets, real-time tweets, and synthetic data generated by AI models.	CNN, ANN, RNN, LSTM, GRU, BiLSTM, and BiRNN	BiLSTM achieved the highest F1 score of 98.77%.
	[2]	LSTM-based approach for classification.	LSTM	Achieved accuracy, precision, recall, and F1 scores above 98%.
	[9]	Introduced the TSA-LSTM-RNN.	TSA-LSTM-RNN	Achieved 93.17% (accuracy for human dataset) and 93.83% (ChatGPT 4o dataset).
Alternative approaches	[18]	Classification using statistical methods.	SVM	Binary classification achieved 99.87% accuracy; accuracy for rephrased texts was 87.58% (classification) and 84.91% (statistical).
	[10]	DeepFake text detection.	TCN and four modified TCN models	Modified TCN models outperformed the baseline, achieving F1 scores exceeding 98% and accuracies between 94% and 99%.
This study		Differentiate and classify AI and human-generated text.	LSTM	F1 score for both AI and human generated text was best achieved for model 1 (20 epochs and 128 patch size), achieving above 98%. Other models achieved above 96%.

Early detection tools have focused on linguistic cues and metadata. For example, the authors in [19–21] used features like syllable count, sentence structure, perplexity, and semantic similarity to differentiate between human and AI texts. Although these models achieved notable accuracy (up to 93%), their reliance on surface-level features makes them susceptible to evasion when AI models improve their mimicry of human style. Tools like GPTZero and ZeroGPT, though widely used, were found to underperform when compared with feature-enhanced models. Studies also integrated explainable AI (xAI) tools (e.g., LIME, SHAP) to interpret model decisions, thus increasing transparency [19]. However, while feature-based methods benefit from interpretability, they often struggle to generalize across domains and newer generative models [17,22].

Several researchers have assessed traditional ML algorithms for text classification. The authors in [23] found that NuSVC outperformed other algorithms like random forest and KNN on a balanced dataset, while the authors in [24] compared decision trees, SVM, and logistic regression. These studies reveal the trade-off between simplicity and accuracy: while ML models offer fast computation and easier training, they tend to underperform on nuanced textual patterns, especially for large-scale and multilingual data.

Recent literature has pivoted toward deep learning due to its superior performance in handling sequential data and capturing contextual nuances. The authors in [15] introduced a Deep Recurrent Neural Network (DRNN), while the authors in [14] conducted a broad comparison across CNN, ANN, RNN, LSTM, GRU, BiLSTM, and BiRNN. Among these, BiLSTM achieved the highest F1 score (98.77%), demonstrating its robustness in identifying subtle differences between AI-generated and human text. Similarly, the authors in [2,9] showed that LSTM-based models, including hybrid variants like TSA-LSTM-RNN, achieved high accuracy (above 93%), although were still somewhat behind the best BiL-

STM configurations. These deep learning models offer high adaptability but often lack interpretability and require large datasets for optimal performance.

Several studies have explored combining deep learning with traditional or optimization techniques. The authors in [18] applied SVM and linguistic analysis to reach an accuracy of 99.87% on binary classification tasks, outperforming many neural models. The authors in [10] employed temporal convolutional networks (TCNs), creating modified variants that reached F1 scores above 98%. These advanced methods highlight a trend toward building hybrid models that balance interpretability, performance, and robustness.

A major gap in the literature is the linguistic diversity of research. The overwhelming majority of existing models have been developed and tested on English datasets (e.g., [9,14]. Few studies have considered low-resource languages, such as Turkish. Moreover, many of the datasets used are either small scale or artificially constructed, limiting generalizability. The review by [1], covering 17 detection tools, emphasized this issue and recommended the integration of multiple evaluation methods, including human review, for enhanced reliability. Despite the growing body of research, two critical gaps persist: (1) the limited application of deep learning techniques to Turkish language detection, (2) the absence of high-quality, native Turkish datasets that include both human-written and AI-generated texts.

To address these gaps, our research proposes an LSTM-based framework specifically trained and evaluated on Turkish texts. Our study introduces a novel, domain-specific Turkish dataset composed of original and ChatGPT-generated content. The model achieved a testing accuracy and F1 score exceeding 97%, outperforming previous LSTM approaches (e.g., [2,9]) and closely matching top performers like [10,14]. This positions our study as a pioneering contribution to multilingual AI detection research and offers practical implications for educational, journalistic, and governmental use cases in Turkish-speaking contexts.

3. Materials and Method

LSTMs' capacity to model both short-term and long-term dependencies makes them uniquely positioned to handle the subtleties of distinguishing AI-generated content. This capability adds unique value by enabling the fine-grained analysis of textual features that may go unnoticed in simpler models. Their interpretability and performance in sequence-based tasks make LSTMs a powerful choice for this application, providing reliability and depth of results. This section outlines the materials and methods used to train and test the LSTM models that distinguish between AI-generated and human-written text. It provides a detailed discussion of the dataset creation process, data preprocessing steps, model training procedures, and approaches for evaluation and validation. The model was trained on a system equipped with an NVIDIA RTX 2060 GPU (NVIDIA, Santa Clara, CA, USA), an AMD Ryzen 5 7600X processor (AMD, Santa Clara, CA, USA), and 16 GB of RAM. Due to these hardware constraints, the LSTM architecture was chosen over more resource-demanding models like Transformers, and the dataset size was kept within manageable limits to optimize performance.

3.1. Dataset Creation

The dataset was created by combining sentences generated by AI with human-generated sentences. AI-generated sentences were produced using OpenAI's ChatGPT-4o [25], Gemini–2.0 Flash [26], and standard Microsoft 365 Copilot Chat (Free version) [27], spanning various academic topics. The dataset used in this study consisted of 4000 statements deliberately curated to represent a broad spectrum of academic disciplines, including engineering, medicine, social sciences, and natural sciences. This approach ensured a comprehensive and diverse dataset, reflecting the varied linguistic structures and domain-

specific language encountered in scholarly writing. The AI-generated content was produced using three leading language models: OpenAI's ChatGPT, Google's Gemini, and GitHub's Copilot. The prompts for these models were carefully crafted to elicit high-quality, scholarly responses across multiple subjects, ensuring that the generated text mirrors the complexity and depth expected in academic discourse. This rigorous design process aimed to produce contextually rich samples that accurately reflect real-world academic writing.

In contrast, the human-authored component of the dataset was sourced from peer-reviewed academic publications and samples of university-level writing. Sentences were meticulously selected and validated by domain experts to confirm their authenticity, coherence, and alignment with the stylistic and thematic conventions of scholarly communication. This verification process ensured that the human-written text maintained a high standard of linguistic quality, providing a robust baseline for model evaluation. To further strengthen this study's experimental reproducibility and evaluate the model's generalization capability, an external test set of 400 newly collected phrases was included. This test set comprised new AI outputs and human-written academic sentences not present in the original training data, allowing for a more rigorous assessment of the model's ability to distinguish between known and novel text samples.

Combining the AI and human-written datasets resulted in a labeled dataset, with AI-generated texts marked as "AI" and human-written ones as "Human". In this study, no distinct datasets were created for each specific LLM model. Instead, outputs generated by ChatGPT, Gemini, and Copilot were grouped together and uniformly labeled as "AI". The train_test_split tool from scikit-learn was used to divide the dataset into testing and training sets, with 70% for training and 30% for testing. Class distribution was balanced across both sets. The LSTM model was developed using TensorFlow. The core structure of the model is as follows:

- Embedding layer: Used to create numerical representations of text, with the vocabulary size reduced to a maximum of 10,000 words. In this study, the tokenizer vocabulary was capped at 10,000 words to achieve a balance between computational efficiency and effective text representation. Based on Zipf's Law, a small set of high-frequency words accounts for most of the text, making this limit sufficient for capturing essential semantics while avoiding the burden of rare words. Prior research [28] also supports that focusing on frequent terms improves model performance without significant semantic loss. Moreover, restricting the vocabulary size helps the model generalize better by reducing overfitting and unnecessary memorization.
- LSTM layer: composed of LSTM layers, with a kernel regularizer applied.
- Dropout layer: included to prevent overfitting.
- The test data ratio: the proportion of data used for testing training data. It was set to 30% for the first three tests.
- Batch normalization layer: added to normalize the distributions between layers.
- Dense layer: the output layer uses a sigmoid activation function to classify text as AI-generated or human-written.
- The Adam algorithm was used as the optimizer for training the model and the learning rate was set to 0.0005. The early stopping technique was applied and the binary_crossentropy loss function was utilized to construct the model.

3.2. Data Preprocessing

The dataset consisted of labeled sentences, categorized as either AI-generated or human-written. It was loaded from a CSV file and processed for text and label extraction. Labels were numerically encoded using LabelEncoder from the scikit-learn library, where "AI" was assigned 0 and "Human" was assigned 1. Text data were tokenized using

TensorFlow's Keras Tokenizer, configured to consider the 10,000 most frequent words. Each sentence was converted into sequences of integers, padded to a uniform length of 1000 tokens to fit the neural network input requirements. Figure 1 depicts the Pseudocode used to prepare the dataset for analysis.

```
BEGIN
// Load the dataset
    Load CSV file 'veri_seti.csv' into dataframe 'df'
// Separate texts and labels
    Extract first column of 'df' into 'texts'
    Extract second column of 'df' into 'labels'
// Encode labels (AI = 0, Human = 1)
    Initialize label encoder
    Convert 'labels' into numerical form using label encoder and store in 'encoded_labels'
// Convert texts to numerical sequences using Tokenizer
    Initialize tokenizer with a vocabulary size of 10,000 words
    Fit tokenizer on 'texts'
    Convert 'texts' into sequences of numbers
// Pad sequences to ensure equal length
    Set 'max_sequence_length' to 1000
    Pad 'sequences' to have a maximum length of 'max_sequence_length'
    Store the result in 'data'
// Split data into training and test sets (70% training, 30% test)
    Split 'data' and 'encoded_labels' into 'X_train', 'X_test', 'y_train', and 'y_test'
    with test size of 30% and random seed 42
// Convert labels to binary format
    Convert 'y_train' into a NumPy array
    Convert 'y_test' into a NumPy array
// Create model
    Initialize a sequential model
    Add an embedding layer with:
        - Input dimension: 10,000
        - Output dimension: 100
        - Input length: 'max_sequence_length'
    Add an LSTM layer with:
        - 64 units
        - No return sequences
        - L1 and L2 regularization with 0.001 each
    Add a dropout layer with 40% dropout rate
    Add batch normalization layer
    Add a dense output layer with:
        - 1 neuron
        - Sigmoid activation
        - L1 and L2 regularization with 0.001 each
// Define optimizer
    Initialize Adam optimizer with a learning rate of 0.0005
// Compile model
    Compile model with:
        - Loss function: Binary cross-entropy
        - Optimizer: Adam
        - Metric: Accuracy
// Define early stopping
    Initialize early stopping with:
        - Monitor validation loss
        - Patience of 5 epochs
        - Restore best weights if training stops

END
```

Figure 1. Data preparation pseudocode.

3.3. Model Training

Adam, an optimizer with a learning rate of 0.0005, was employed for training. The binary_crossentropy loss function was used to construct the model and an early halting

technique was used. The Adam optimizer was used to assemble the model because of its effectiveness and capacity for adaptive learning. Binary cross-entropy was selected as the loss function because it performs well for binary classification problems. During training, accuracy served as the performance indicator. An early halting mechanism was used to reduce overfitting. This system tracked the validation loss and stopped training after five epochs in which no progress was seen. To guarantee that the model's performance was maximized, the optimal weights were restored after termination. Figure 2 depicts the Pseudocode used to train the dataset. Also, Figure 3 shows a sample of the trained dataset.

```
BEGIN
// Train the model
    Train 'model' using:
        - Training data: 'X_train', 'y_train'
        - 50 epochs
        - Batch size: 32
        - Validation data: ('X_test', 'y_test')
        - Callback: 'early_stopping'
    Store training history in 'history'

// Plot training and validation loss
    Plot 'history' training loss over epochs
    Plot 'history' validation loss over epochs
    Label x-axis as 'Epoch'
    Label y-axis as 'Loss'
    Add legend: 'Loss of Training' and 'Validation Loss'
    Set title: 'Loss of Training and Validation'
    Display plot
END
```

Figure 2. Data training pseudocode.

	Sentences	ai or human
0	"Alzheimer hastalığı, beyin hücrelerinin ölümü...	ai
1	"Kripto para, blok zincir teknolojisiyle güven...	ai
2	"Sınıflar, nesnelerin özelliklerini ve işlevle...	ai
3	"Tekrarlayan sinir ağları, önceki bilgilerin b...	human
4	"Bir kitabı okuduğunuzda, o kitap size birçok ...	ai
5	"Bu görüntü ağdaki katmanlardan geçirilerek bi...	human
6	"Evrenin genişlemesi, kozmolojinin temel sorul...	ai
7	"Bu çalışma, derin öğrenme tekniklerinde LSTM ...	human
8	"Veri madenciliği, büyük veri kümelerinden anl...	ai
9	"Ana dili eğitimi ve öğretimi süreci dört ...	human
10	"Reading and Writing okuma ve yazma beceri...	human
11	"Bir sınıf, nesnelerin özelliklerini ve davran...	ai
12	"Bu derin ağı doğrusal olmayan bir yapıya sokm...	human
13	"Klorofil, bitkilerin güneş enerjisini kimyasa...	ai
14	"Bir liste, birden fazla elemanı sıralı bir şe...	ai
15	"Bu şekilde, MAE kaybının dağılımını ve örnek ...	human
16	"Şekil 2'de yazma alanyazınında en sık bir...	human
17	"Bu bir zorunluluk olduğu gibi gerçekleşti...	human
18	"Duygu analizi, insanların duygusal tepkilerin...	ai
19	"Bu çalışmalarda Türkiye'de yazma becerisi...	human

Figure 3. Sample of the trained dataset.

3.4. Model Evaluation and Validation

Following training, loss metrics, and accuracy were used to assess the model on the test set. A categorizing report that comprised measures like accuracy, precision, recall, F1 score, and support for each class was produced by comparing predictions made on the test data with the true labels. The distribution of true positives, true negatives, false positives, and false negatives was also examined using a confusion matrix. A user interaction module was created to display the model's capabilities. Users could enter a sentence into this module, which would be tokenized and padded to fit the model's input format. The model's prediction was converted into a readable label ('AI' or 'Human') and displayed to the user. Figure 4 depicts the Pseudocode used to run the tests.

```
BEGIN
// Evaluate the trained model
    TEST_LOSS, TEST_ACCURACY = Evaluate 'model' on (X_test, y_test)
    PRINT "Test Accuracy:", TEST_ACCURACY
// Predict class labels for test data
    Y_PRED = Predict using 'model' on 'X_test'
    Convert 'Y_PRED' to binary values (0 or 1) by rounding
// Print classification report
    PRINT "Classification Report:"
    PRINT Classification report for 'y_test' vs 'y_pred' with target names: ['AI', 'Human']
// Compute confusion matrix
    CONF_MATRIX = Compute confusion matrix for 'y_test' and 'y_pred'
    PRINT "Confusion Matrix:", CONF_MATRIX
// Extract True Negatives (TN), False Positives (FP), False Negatives (FN), and True Positives (TP)
    TN, FP, FN, TP = Extract values from CONF_MATRIX
// Function to classify user input text
    FUNCTION text_prediction (model, tokenizer, max_sequence_length):
        PRINT "Enter a long article:"
        USER_INPUT = Read user input
        Split 'USER_INPUT' into sentences using regex
        FOR each sentence in the list:
            Convert sentence to numerical sequence using 'tokenizer'
            Pad the sequence to 'max_sequence_length'
            Predict class using 'model'
            Determine predicted class:
                IF prediction < 0.5 THEN CLASS = 'AI'
                ELSE CLASS = 'Human'
            PRINT "Sentence", sentence
            PRINT "Predicted Class:", CLASS
// Call the function for user input classification
    CALL text_prediction (model, tokenizer, max_sequence_length)
END
```

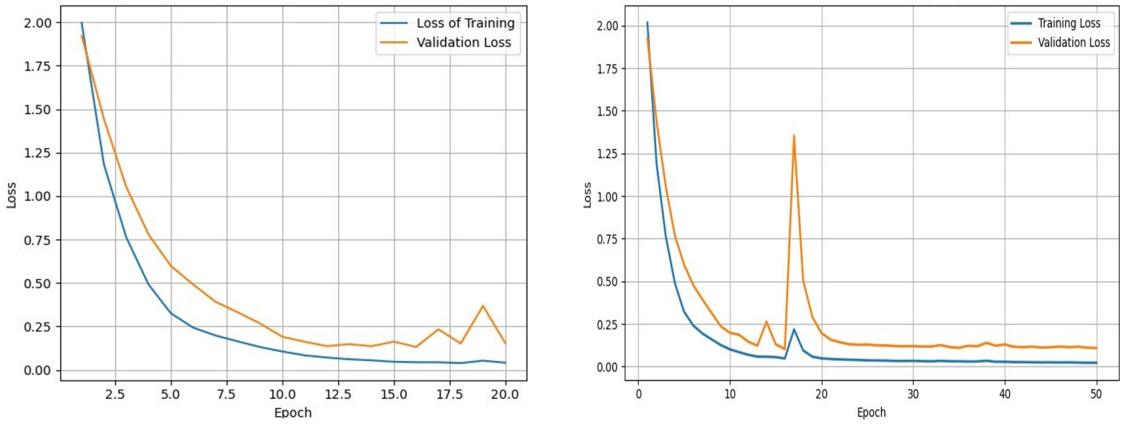
Figure 4. Test pseudocode.

4. Results and Discussion

This paper conducts several tests to identify the most accurate LSTM model. The first two tests involve varying the number of epochs, while the third test explores the impact of adjusting the dropout rate, number of epochs, embedding size, and patch size. This section presents and analyzes the results from all three tests in detail.

4.1. Test 1

The model was trained for a total of 20 epochs and 50 epochs. Figure 5 summarizes the results of test 1 for both 20 and 50 epochs. Based on the training results, the trends in the training and validation loss graphs were similar. The gap between training and validation loss was minimal, and there was no significant increase in validation loss, indicating that the model did not overfit. The early stopping strategy ensured that training was halted when the best validation loss was achieved, further contributing to overfitting prevention. Typical signs of overfitting, such as a rapid decrease in training loss alongside an increase in validation loss, were not observed in this study. Both training and validation losses decreased in a balanced manner. When interpreting the graph, we can analyze the model's performance by examining how the training and validation losses change per epoch. The graph shows the training loss and validation loss at the end of each epoch during training. However, during the 50-epoch training, there were imbalances and increases in the val_loss values. This suggests that the model overfitting led to a decrease in overall performance. Moreover, the 50-epoch model showed overfitting, adapting too much to the training data, which resulted in lower performance on the validation and test sets. Figure 5 depicts the training loss for test 1.



(**a**): 20 Epochs (**b**): 50 Epochs

Figure 5. Test 1 training loss validation.

Table 2 depicts a summary of results from test 1. The results demonstrate that this LSTM-based deep learning model can distinguish between human- and AI-generated texts with very high accuracy. The F1 scores and accuracy rates indicate that the model provides a balanced performance and successfully classifies both classes. Furthermore, the high and balanced F1 scores suggest that the model performed well. The model's overall performance is adequate and has the potential for application in various domains.

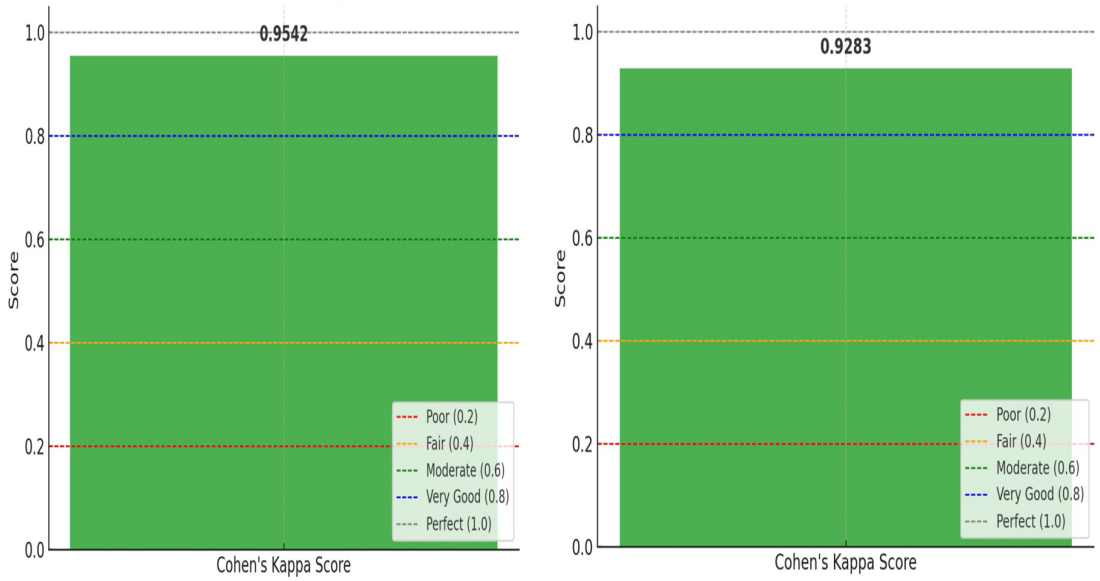
Table 2. Test 1 results.

Model Features	20-Epoch Model	50-Epoch Model
Patch size	128	128
Embedding size	300	300
Regularization (L1, L2)	0.0001	0.001
Dropout rate	20%	20%
Model's test accuracy	97.28%	95.99%
AI precision	97%	91%
Human precision	98%	92%
AI recall	98%	92%
Human recall	97%	91%
AI F1 Score	98%	87%
Human F1 Score	97%	88%

The model trained with 20 epochs showed a better performance. This indicates that the model received sufficient training after 20 epochs and that further training caused overfitting. For the second model, the precision and recall values for AI and Human classes are less than those of the first model. This comparison highlights that the first model performs better in accuracy and F1 score. In contrast, the second model shows some deterioration in performance, particularly with increased loss and changes in the confusion matrix. This suggests that the model overfitted to the training data, leading to a decrease in overall performance. Moreover, the second model showed overfitting, adapting too much to the training data, which resulted in lower performance on the validation and test sets.

As illustrated in Figure 6, the model trained with 20 epochs achieved a Kappa metric score of 0.9542, which is notably higher than the 0.9283 attained by the 50-epoch model. This metric, which measures the agreement between predicted and actual classifications while accounting for chance agreement, provides further evidence of model reliability. The superior Kappa score of the 20-epoch model reinforces our earlier findings, confirming that a shorter training duration—when coupled with appropriate regularization and architecture settings—can lead to more effective generalization and better classification consistency.

In contrast, the 50-epoch model, despite longer training, exhibited signs of overfitting, resulting in reduced agreement and lower robustness in its predictions. Overall, the 20-epoch model is more optimized, showing a more balanced performance and providing better test accuracy and loss values. Therefore, it has a higher generalization capacity. Moreover, effective use of the early stopping technique is important to prevent the model from over-learning and to determine the optimum number of epochs. Figure 6 depicts the Cohen's kappa scores for test 1.

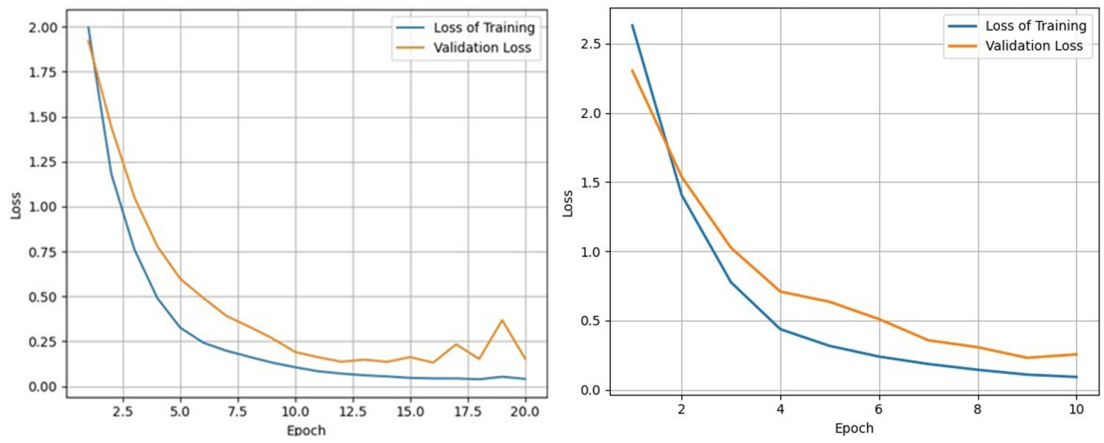


(**a**): 20 Epochs (**b**): 50 Epochs

Figure 6. Cohen's kappa scores for 20 epochs and 50 epochs.

4.2. Test 2

In this test, we trained the model for 20 epochs, using the same parameters as the first experiment. All other parameters were identical, but the second model was trained for 10 epochs rather than 50 epochs as in test one. Early stopping was used to optimize the validation loss value after 20 epochs of training the first model. The validation loss value for the second model dramatically dropped throughout ten training epochs. This model required fewer epochs and a faster training time. Figure 7 shows the validation loss analysis. Figure 7 depicts the training loss for test 2.



(**a**): 20 Epochs (**b**): 10 Epochs

Figure 7. Test 2 training loss validation.

A summary of the results of test 2 is depicted in Table 3. The first model, with a higher embedding size (300) and lower regularization coefficient ("l1, l2" = 0.0001), achieved more general learning and prevented the model from overfitting. In the second model, the use of a lower embedding size (100) and higher regularization ("L1, L2" = 0.001) led to insufficient learning, resulting in a decrease in overall performance. In the first model, a lower dropout rate (20%) minimized the loss of important information during learning. In contrast, the higher dropout rate (40%) in the second model caused more data to be lost, leading to decreased performance.

Table 3. Test 2 results.

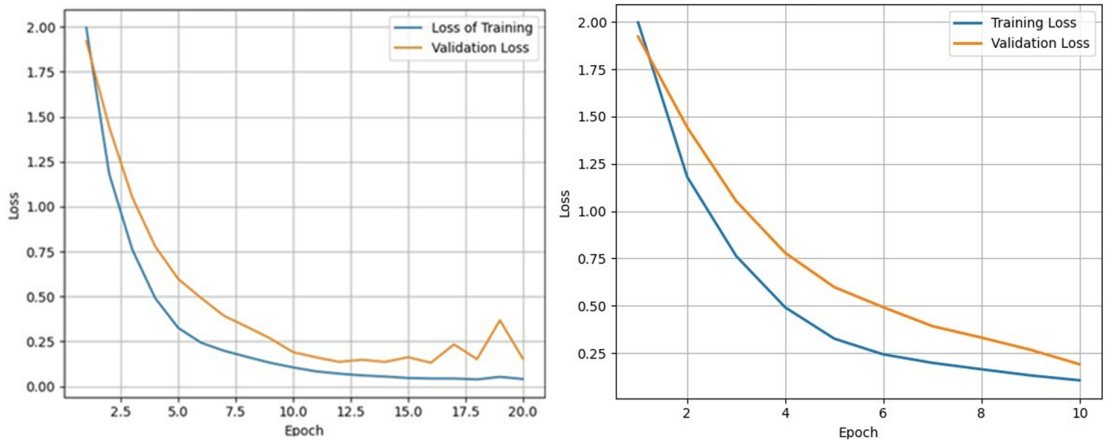
Model Features	20-Epoch Model	10-Epoch Model
Patch size	128	128
Embedding size	300	100
Regularization (L1, L2)	0.0001	0.001
Dropout rate	20%	40%
Model's test accuracy	97.28%	93.12%
AI precision	97%	99%
Human precision	98%	88%
AI recall	98%	87%
Human recall	97%	99%
AI F1 Score	98%	93.6%
Human F1 Score	97%	94%

With 97.28% accuracy, the first model outperforms the second model (93.12%). This indicates that the first model is more successful in accurately classifying the texts. The second model has a high precision for the AI class but a low recall. This suggests that the number of missed AI class examples increased. Again, here, the first model exhibited a more optimized performance overall with better accuracy, balanced precision, and recall values. In the second model, low embedding size and high regularization rates caused the model's performance to decrease.

4.3. Test 3

We trained the model for 20 epochs in this test, using the same parameters as the first two experiments. The second model in this test, trained for 10 epochs, maintained all the settings of the 10-epoch model from the second test, with the only modification being a reduction in patch size from 128 to 64. The first model was optimized using early stopping. The second model was trained for 10 epochs, during which the validation loss value significantly decreased. This model required fewer epochs and a faster training time. Figure 8 shows the validation loss patterns. Figure 8 depicts the training loss for test 3.

A summary of the results of test 3 is depicted in Table 4. In the first model (20 epochs), for both classes ("AI" and "Human"), F1 score, recall, and precision have values between 97 and 98%. In the second model, precision, recall, and F1 scores for both classes ("AI" and "Human") range between 87% and 99%. The second model used fewer epochs (10 epochs) and had a faster training time, which is particularly advantageous regarding time and resource usage. The first model exhibited a more optimized performance overall with better accuracy, balanced precision, and recall values. In the second model, low embedding size and higher dropout rate caused the model's performance to decrease.



(**a**): 20 Epochs (128) (**b**): 10 Epochs (64)

Figure 8. Test 3 training loss validation.

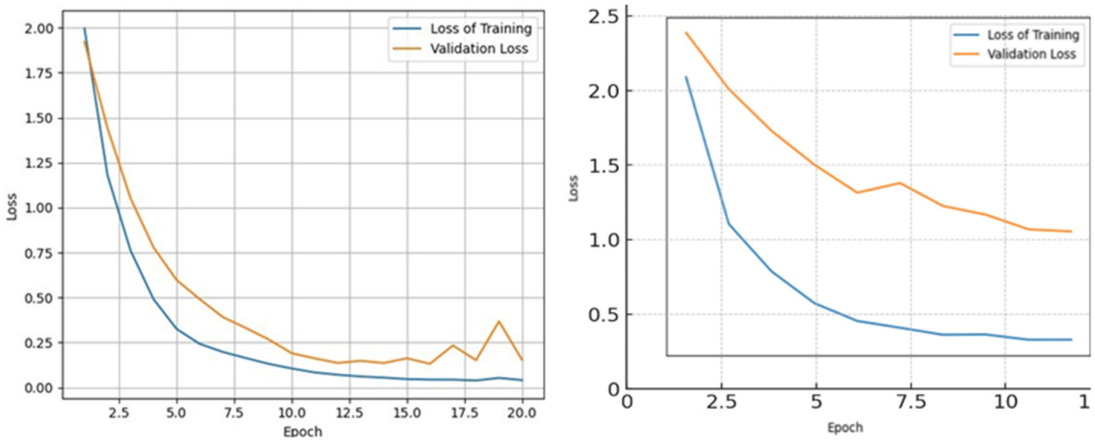
Table 4. Test 3 results.

Model Features	20-Epoch Model	10-Epoch Model
Patch size	128	64
Embedding size	300	100
Regularization (L1, L2)	0.0001	0.001
Dropout rate	20%	40%
Model's test accuracy	97.28%	93.12%
AI precision	97%	99%
Human precision	98%	88%
AI recall	98%	87%
Human recall	97%	99%
AI F1 Score	98%	96.19%
Human F1 Score	97%	96%

The higher number of LSTM units (128) and embedding size (300) in the first model may have allowed it to learn more features, enhancing classification success. However, this comes at the detriment of requiring more computational power for training. Using a lower embedding size (100) and a higher regularization in the second model caused the model to not learn enough, and its overall performance decreased. While the lower dropout rate (20%) in the first model minimized the loss of important information during learning, the higher dropout rate (40%) in the second model caused the model to lose more information and decreased performance.

4.4. Test 4

The effects of changing the test data ratio (test_size = 0.3 and test_size = 0.6) on an LSTM model were examined. The impact of this change in training and testing data ratios on the model's accuracy, loss values, classification report, and confusion matrix were analyzed comparatively. The ratio that gives more optimized results was evaluated, and its reasons and consequences were considered. This model required fewer epochs and a faster training time. As shown in Figure 9, although the training accuracy increased at a 60% test rate, a decrease in test accuracy was observed. Low training loss and high test loss indicate that the model is over-learning. On the other hand, at a 30% test rate, training and test performances are compatible, and no signs of over-learning are observed. Figure 9 depicts the training loss for test 4.



(**a**): 20 Epochs (30%) (**b**): 20 Epochs (60%)

Figure 9. Test 4 training loss validation.

A summary of the results of test 4 is depicted in Table 5. According to Table 5, the model shows a more optimized performance when the test rate is 30%. This rate provided a balanced and high accuracy in training and test data accuracy and did not show signs of over-learning. On the other hand, when the test rate was 60%, the test performance of the model decreased, and over-learning was observed. Therefore, the test rate of 30% was considered a more appropriate choice in terms of the generalizability of the model.

Table 5. Test 4 results.

Model Features	20-Epoch Model	20-Epoch Model
Patch size	128	128
Embedding size	300	300
Regularization (L1, L2)	0.0001	0.0001
Dropout rate	20%	20%
Test data ratio	30%	60%
Model's test accuracy	97.28%	99.89%
AI F1 Score	98%	91%
Human F1 Score	97%	90%

5. Discussion

This study aims to examine AI-generated text detection in Turkish using LSTM networks. We conduct four tests, changing the settings to select the best model (e.g., number of epochs, patch size, sample size, dropout rate, and testing data rate). This section details the limitations, future research directions, and implications of the findings.

5.1. Results Comparison

Table 2 demonstrates that the 20-epoch model consistently outperforms the 50-epoch model in terms of F1 Score and test accuracy. This performance gap validates the effectiveness of our initial model design, which prioritized a reduced regularization strength and a longer training period. These choices contributed to superior generalization and a balanced trade-off between precision and recall for both AI-generated and human-authored text, reinforcing the robustness of our approach. The incorporation of internal baselines further strengthens the credibility of these findings, underlining the innovative nature of our method.

Table 3 outlines the configuration of the 10-epoch model, where several key hyperparameters were deliberately modified to assess their influence on learning capacity and

overall performance. The embedding size was lowered from 300 to 100, limiting the model's ability to capture complex semantic relationships. A smaller embedding dimension restricts the available representational space, reducing the model's ability to discern intricate patterns and meaningful context within the training data. The regularization strength was elevated from 0.0001 to 0.001 (L1, L2), introducing a stronger penalty for large weights. While this can help control overfitting, it also reduces the model's flexibility, potentially restricting its capacity to learn nuanced features from the data. The dropout rate was increased from 20% to 40%. Although dropout is an effective mechanism to prevent overfitting by randomly omitting neurons during training, an excessively high dropout rate can disrupt learning by eliminating too much critical information, leading to fragmented learning and weakened model coherence. In contrast, the 20-epoch model exhibited superior performance in test 3, benefiting from its larger number of LSTM units (128) and a more expansive embedding size (300). These parameters significantly enhanced the model's representational capacity, enabling it to effectively capture complex semantic and sequential dependencies within the text. This configuration allowed the 20-epoch model to extract richer contextual insights, resulting in more precise and reliable classifications.

Collectively, these results underscore the importance of carefully tuning hyperparameters to achieve optimal performance, balancing model complexity, regularization, and expressive power to enhance both accuracy and generalization. This study effectively demonstrated the potential of LSTM models for detecting Turkish AI-generated academic writing, providing critical insights into how the model configuration impacts its performance and generalization. By systematically adjusting epoch count, embedding size, dropout rate, and regularization, we identified the optimal settings that significantly enhance detection accuracy. Specifically, the model trained with 20 epochs, a larger embedding size (300), and a lower regularization coefficient (0.0001) outperformed its counterparts, achieving better generalization and reduced overfitting. In contrast, the 50-epoch model, despite its higher training accuracy, exhibited signs of overfitting, as reflected in the widening gap between training and validation loss. Moreover, our findings underscore the critical balance between model complexity and generalization. While increasing complexity can improve learning, it also increases the risk of overfitting. However, incorporating strategies like early stopping and dropout effectively mitigated this risk, minimizing the discrepancies between training and validation performance, and ensuring robust, reliable classification.

To study the effect of different parameters, we tested three factors: epoch number, learning rate, and dropout, as shown in Figure 10. The comparative analysis of the two models (i.e., 20-epoch and 10-epoch) trained on identical datasets with modified hyperparameters reveals several critical insights:

- Epoch reduction (20 $\rightarrow$ 10): The reduction in epoch count from 20 to 10 had a comparatively minor impact. While it reduced training time, the overall structure and performance of the model remained largely intact. This suggests that, in this context, fewer epochs were sufficient for capturing the core patterns without significantly compromising accuracy, provided that other hyperparameters were well tuned.
- Dropout rate impact (20% $\rightarrow$ 40%): The most pronounced effect on model performance resulted from the increased dropout rate, which increased from 20% to 40%. This adjustment led to a substantial decline of approximately 10–11 points in the F1 score, indicating that the model experienced over-regularization. High dropout rates, while effective for reducing overfitting, can excessively disrupt the learning process, limiting the model's ability to capture complex patterns in the data. This aligns with the broader understanding that excessive dropout can degrade model capacity and stability.
- Learning rate adjustment (0.0001 $\rightarrow$ 0.001): The shift from a 0.0001 to 0.001 learning rate had the second most significant impact. While this increase accelerated the training

process, it also introduced the risk of unstable weight updates, potentially causing the model to oscillate around local minima or fail to converge effectively. This underscores the importance of selecting an optimal learning rate to balance training speed and model stability.

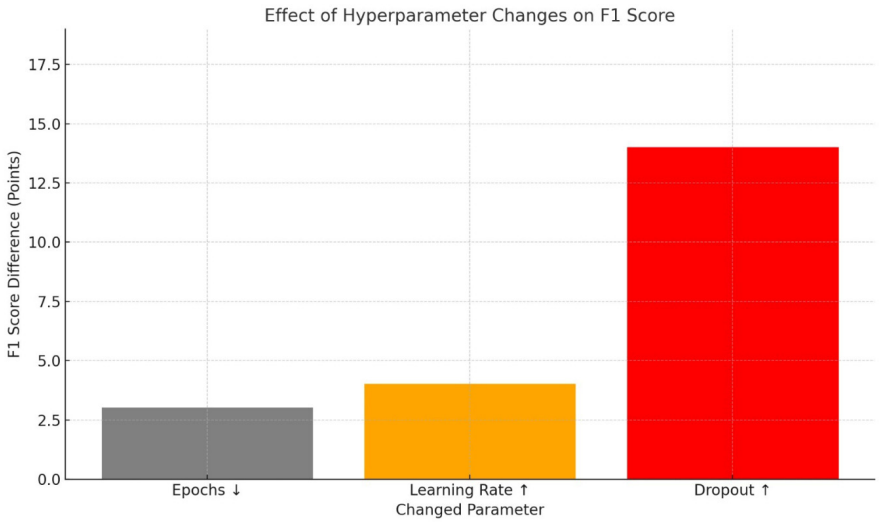


Figure 10. Parameter effect on the results.

5.2. Practical and Research Implications

By improving our knowledge of semantic characteristics, this study opens the door to creating more accurate screening methods to detect AI-generated content. This provides fresh evidence that can be investigated empirically in various ways. These findings also offer a starting point for further study to increase the precision of identification and identify more patterns exclusive to AI-generated text. Future research could improve these results to further our understanding of this field while tackling new opportunities and problems in detecting material created by AI. The findings also enable academics to establish more specific goals to further investigate humans' ability to identify AI-generated text. These results have applications in developing technologies to identify spam, fake news, and other unwanted content, potentially shielding people from deception. By enhancing the reliability of online communication, such solutions will guarantee more secure and dependable digital interactions.

Practically speaking, integrating mechanized and human recognition systems provides a potent composite solution that combines the accuracy of algorithms with the sophisticated discernment of human experience. Tasks like real-time text analysis can be accelerated with this cooperative method while retaining a high degree of contextual correctness. Examples include monitoring online conversations to guarantee prompt action while considering language communication difficulties. Moreover, this study tackles essential issues, including thwarting disinformation and enhancing public dialogue and decision-making procedures. It draws attention to the ethical ramifications of AI-driven communication and emphasizes the value of analytical methods to slow the spread of AI-generated content.

5.3. Limitations and Future Directions

Although we provide a reliable detection model using LLMs in this study, there are still certain restrictions and gaps. The dataset employed in this study is somewhat small, mainly due to limitations in computational resources. Using bigger datasets, future investigations may examine how well the suggested models function in various languages. Identifying Turkish AI-generated content in domain-specific corpora, such as those in education, is

an exciting approach. Additionally, efforts ought to be focused on creating instruments that improve communication and the quality of information while shielding consumers from being duped by false information. The results of this study may be expanded upon in future research by investigating context-aware detection techniques. ChatGPT is also the only tool used to generate text in the study sample. Furthermore, comparing in-context and out-of-context text production detection would yield insightful information and greatly enhance our knowledge of AI-generated content recognition. Finally, although 10-fold cross-validation is a widely used approach for assessing the generalization ability of machine learning models, it was not adopted in this study due to its substantial computational cost and the relatively small size of the dataset. Instead, a straightforward training test split was employed, providing a more practical and efficient means of evaluating model performance while minimizing the overfitting risks.

6. Conclusions

This study aimed to develop and evaluate a robust deep learning model capable of accurately distinguishing between human-written and AI-generated text in Turkish, a relatively underexplored linguistic context. To achieve this, we implemented LSTM networks and systematically examined the effects of various hyperparameters, including patch size, embedding size, number of epochs, dropout rate, and test–train data ratios. Our experimental results revealed that model configurations significantly influence detection performance. Specifically, models trained for fewer epochs (20) outperformed longer training runs (50 epochs), as extended training led to increased validation loss and overfitting. Similarly, models with a higher embedding size (300) and lower regularization (0.0001) demonstrated improved learning stability and generalization. Additionally, a 30% test data ratio yielded better overall performance than 60%, highlighting the impact of data distribution on model accuracy.

While increasing model complexity can enhance training accuracy, our findings emphasize the trade-off between accuracy and overfitting risk. The use of dropout and early stopping techniques successfully mitigated overfitting in optimized configurations, as indicated by stable validation losses and minimal training–validation divergence. Nonetheless, simpler models exhibited faster training times but struggled with classification precision, leading to increased false positives when detecting AI-generated text. The value of this research lies in its focus on Turkish-language AI-generated text detection—a niche area with limited prior studies—and in demonstrating that carefully tuned LSTM models can achieve high accuracy (above 97%) in this task. Moreover, this study contributes practical insights into the trade-offs between model complexity, training duration, and generalization performance. Future research should extend these findings by leveraging larger and more diverse Turkish-language datasets to improve robustness and linguistic coverage. Also, exploring hybrid deep learning models (e.g., BiLSTM-CNN, transformer-based architectures) could further enhance detection accuracy. Furthermore, the integration of semantic and contextual linguistic features to better handle rephrased or adversarial AI-generated texts could be explored. By addressing these avenues, future studies cannot only refine detection techniques but also contribute to broader efforts in combating the misuse of generative AI in education, journalism, and digital communication.

Author Contributions: Conceptualization, A.K.; methodology, M.Y. and A.E.T.; validation, A.E.T. and M.Y.; analysis, Y.I.A. and A.K.; writing—original draft preparation, A.K. and Y.I.A.; writing—review and editing, Y.I.A. and A.E.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research was not funded by any organization.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Chaka, C. Reviewing the performance of AI detection tools in differentiating between AI-generated and human-written texts: A literature and integrative hybrid review. *J. Appl. Learn. Teach.* **2024**, *7*, 115–126.
2. Chimata, S.; Bollimuntha, A.R.; Devagiri, D.; Puligadda, S. An Investigative Analysis on Generation of AI Text Using Deep Learning Models for Large Language Models. In Proceedings of the 2024 International Conference on Smart Systems for Electrical, Electronics, Communication and Computer Engineering (ICSSEECC), IEEE, Coimbatore, India, 28–29 June 2024; pp. 7–12.
3. Alzoubi, Y.I.; Mishra, A.; Topcu, A.E.; Cibikdiken, A.O. Generative artificial intelligence technology for systems engineering research: Contribution and challenges. *Int. J. Ind. Eng. Manag.* **2024**, *15*, 169–179. [CrossRef]
4. Topcu, A.E.; Alzoubi, Y.I.; Elbasi, E.; Camalan, E. Social media zero-day attack detection using TensorFlow. *Electronics* **2023**, *12*, 3554. [CrossRef]
5. Menard, P.; Bott, G.J. Artificial intelligence misuse and concern for information privacy: New construct validation and future directions. *Inf. Syst. J.* **2025**, *35*, 322–367. [CrossRef]
6. Zhou, J.; Müller, H.; Holzinger, A.; Chen, F. Ethical ChatGPT: Concerns, challenges, and commandments. *Electronics* **2024**, *13*, 3417. [CrossRef]
7. Sanchez, T.W.; Brenman, M.; Ye, X. The ethical concerns of artificial intelligence in urban planning. *J. Am. Plan. Assoc.* **2025**, *91*, 294–307. [CrossRef]
8. Alzoubi, Y.I.; Mishra, A.; Topcu, A.E. Research trends in deep learning and machine learning for cloud computing security. *Artif. Intell. Rev.* **2024**, *57*, 132. [CrossRef]
9. Katib, I.; Assiri, F.Y.; Abdushkour, H.A.; Hamed, D.; Ragab, M. Differentiating chat generative pretrained transformer from humans: Detecting ChatGPT-generated text and human text using machine learning. *Mathematics* **2023**, *11*, 3400. [CrossRef]
10. Boutadjine, A.; Harrag, F.; Shaalan, K. Human vs. Machine: A Comparative Study on the Detection of AI-Generated Content. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* **2024**, *24*, 1–26. [CrossRef]
11. Elkhatat, A.M.; Elsaid, K.; Almeer, S. Evaluating the efficacy of AI content detection tools in differentiating between human and AI-generated text. *Int. J. Educ. Integr.* **2023**, *19*, 1–17. [CrossRef]
12. Kayabaş, A.; Schmid, H.; Topçu, A.E.; Kiliç, Ö. TRMOR: A finite-state-based morphological analyzer for Turkish. *Turk. J. Electr. Eng. Comput. Sci.* **2019**, *27*, 3837–3851. [CrossRef]
13. Kayabaş, A.; Topcu, A.E.; Kiliç, Ö. A novel hybrid algorithm for morphological analysis: Artificial Neural-Net-XMOR. *Turk. J. Electr. Eng. Comput. Sci.* **2022**, *30*, 1726–1740. [CrossRef]
14. Wani, M.A.; Abd El-Latif, A.A.; ELAffendi, M.; Hussain, A. AI-based Framework for Discriminating Human-authored and AI-generated Text. *IEEE Trans. Artif. Intell.* **2024**, 1–15. [CrossRef]
15. Latif, G.; Mohammad, N.; Brahim, G.B.; Alghazo, J.; Fawagreh, K. Detection of AI-written and human-written text using deep recurrent neural networks. In Proceedings of the Fourth Symposium on Pattern Recognition and Applications (SPRA 2023), Napoli, Italy, 1–3 December 2023; pp. 11–20.
16. Zulqarnain, M.; Alsaedi, A.K.Z.; Ghazali, R.; Ghouse, M.G.; Sharif, W.; Husaini, N.A. A comparative analysis on question classification task based on deep learning approaches. *PeerJ Comput. Sci.* **2021**, *7*, e570. [CrossRef] [PubMed]
17. Sardinha, T.B. AI-generated vs human-authored texts: A multidimensional comparison. *Appl. Corpus Linguist.* **2024**, *4*, 100083. [CrossRef]
18. Kalra, M.P.; Mathur, A.; Patvardhan, C. Detection of AI-generated Text: An Experimental Study. In Proceedings of the 3rd World Conference on Applied Intelligence and Computing (AIC), IEEE, Gwalior, India, 27–28 July 2024; pp. 552–557.
19. Shah, A.; Ranka, P.; Dedhia, U.; Prasad, S.; Muni, S.; Bhowmick, K. Detecting and Unmasking AI-Generated Texts through Explainable Artificial Intelligence using Stylistic Features. *Int. J. Adv. Comput. Sci. Appl.* **2023**, *14*, 1043–1053. [CrossRef]
20. Mindner, L.; Schlippe, T.; Schaaff, K. Classification of human-and ai-generated texts: Investigating features for chatgpt. In *Artificial Intelligence in Education Technologies: New Development and Innovative Practices—AIET 2023—Lecture Notes on Data Engineering and Communications Technologies*; Schlippe, T., Cheng, E., Wang, T., Eds.; Springer: Singapore, 2023; Volume 190, pp. 152–170.
21. Schaaff, K.; Schlippe, T.; Mindner, L. Classification of human-and AI-generated texts for different languages and domains. *Int. J. Speech Technol.* **2024**, *27*, 935–956. [CrossRef]

22. Uzun, L. ChatGPT and academic integrity concerns: Detecting artificial intelligence generated content. *Lang. Educ. Technol.* **2023**, *3*, 45–54.
23. Vashistha, M.; Dhiman, I.; Singh, P.; Kumar, A.; Malhotra, D. A Comparative Study of Classification of Human-Written Text Versus AI-Generated Text. In *Proceedings of International Conference on Recent Innovations in Computing. ICRIC 2023. Lecture Notes in Electrical Engineering*; Illés, Z., Verma, C., Gonçalves, P., Singh, P., Eds.; Springer: Singapore, 2023; Volume 1195, pp. 197–206.
24. Alamleh, H.; AlQahtani, A.A.S.; ElSaid, A. Distinguishing human-written and ChatGPT-generated text using machine learning. In Proceedings of the 2023 Systems and Information Engineering Design Symposium (SIEDS), Charlottesville, VA, USA, 27–28 April 2023; pp. 154–158.
25. OpenAI. Open AI—ChatGPT. Available online: https://chatgpt.com/ (accessed on 7 March 2025).
26. Gemini. Google AI—Gemini. Available online: https://gemini.google.com/app (accessed on 8 March 2025).
27. Copilot. Microsoft Copilot: Your AI companion. Available online: https://copilot.microsoft.com/ (accessed on 9 March 2025).
28. Joulin, A.; Cissé, M.; Grangier, D.; Jégou, H. Efficient softmax approximation for GPUs. In Proceedings of the 34th International Conference on Machine Learning, PMLR, Sydney, Australia, 6–11 August 2017; pp. 1302–1310.