Jurnal Ilmu Komputer dan Informasi (Journal of Computer Science and Information)
18/2 (2025), 239-249. DOI: http://dx.doi.org/10.21609/jiki.v18i2.1530

# Context-Aware Detection of Deceptive Design Patterns in E-Commerce Websites Using Word Embedding Based Deep Learning Paradigms

Rukshika Premathilaka

Department of Information and Communication Technology, Faculty of Technological Studies,
Uva Wellassa University of Sri Lanka, Badulla, Sri Lanka

*E-mail: itt1617059@tec.rjt.ac.lk*

**Abstract**

Deceptive designs (DDs) are a hidden technological tactic that manipulates the user's consumer behavior in a way that benefits website vendors without them knowing. Proper identification of deceptive designs is essential to prevent users from being misled by hidden tactics. To fulfill this requirement, this study assesses Word2Vec word embedding based deep learning models for text based deceptive design detection. Models trained consist of Convolutional Neural Networks (CNN), Bidirectional Long Short-Term Memory (BiLSTM), and a hybrid model (CNN + BiLSTM) that combines the two aforementioned models. These four key score indices of accuracy, precision, sensitivity, and F1-score are computed to compare the performance of each proclaimed model. When compared to the existing DD detection techniques, all three of these approaches attain state-of-the-art performance. The results of this evaluation illustrate that the hybrid model achieves the highest accuracy of 95% in capturing the nuanced text context of deceptive designs. Furthermore, even when other metrics are considered, the hybrid model performs more effectively. To guarantee the independence and security of user activities, intelligent deep learning paradigms are integrated to identify hidden deceptive activities automatically. This allows for the accurate detection and classification of deceptive designs in intricate e-commerce environments.

**Keywords:** *deceptive design detection, word embeddings, CNN, BiLSTM*

## 1. Introduction

"Deceptive design" is a subset of User Interface (UI) design techniques used to influence user behavior on various digital platforms subtly. These misleading designs inadvertently undermine users' autonomy and capacity to make wise decisions. According to earlier research, deceptive designs can be found anywhere on digital platforms, such as social networking sites [1], e-commerce sites, and apps and cookies [2]. These design tactics have garnered more attention lately because many of them are illegal and have ethical effects on the user. Financial loss [3], deceiving users into disclosing a great deal of personal information [4], or causing compulsive and addictive behavior in both adults [5] and children [6] are the most catastrophic results that may arise from these deceitful designs.

Therefore, it is crucial to accurately recognize and comprehend these designs to mitigate the effects. The majority of earlier research concentrated on manually detection [7] or labelling [8] of deceptive design patterns. Currently, several recent studies have taken steps to automatically detect these designs using a few types of traditional machine learning models for feature engineering and classification [9]. The main goal of this research is to address the limitations raised by those earlier studies by offering deep learning model-based solutions.

This study aims to address two key questions to bridge this gap:

**Q1.** How effectively can different deep learning paradigms detect dark patterns in e-commerce websites using word embeddings?

**Q2.** Which deep learning algorithms provide the highest performance for key indicators?

To fill this research gap and enhance design detection performance, we implemented three deep learning paradigms on an improved dataset. Word embeddings have been employed by all models to capture contextual and semantic nuances in the data. The performance of the models was measured using four key indicators. They are *accuracy, precision, sensitivity*, and *F1 score*, respectively.

In summary, the contributions of this work are as follows:

1. *Convolutional Neural Network + Word2Vec*: First, we use a convolutional neural network in conjunction with the Word2Vec model to extract semantic features from textual context. The convolutional layers in this model finds n-grams of word patterns to collect the essential features required for deceptive designs detection.
2. *Bidirectional Long Short-Term Memory + Word2Vec*: Long-range dependencies and contextual linkages are extracted from textual context using this method, which combines BiLSTM layers with a Word2Vec pre-trained model. Because the model is bidirectional, it performs well on tasks that call for in-depth contextual understanding.
3. *Convolutional Neural Network + Bidirectional Long Short-Term Memory with Word2Vec:* The development of this method involved combining the strong elements of the two aforementioned approaches. By integrating the two architectures, it seeks to offer an all-encompassing strategy for identifying deceptive designs while striking a balance between effectiveness and contextual awareness.

Section 2 covers related work. The methodology, including the development of the DDs dataset and the strategy employed in this investigation, is explained in Section 3. Section 4 presents the implementation setup, detailing the performance metrics, confusion matrix and comparative analysis. Section 5 provides in-depth analysis of key findings and model performance. Finally, section 6 concludes the complete study with final remarks.

## 2. Literature

User experience designer Harry Brignull first coined these deceptive designs termed 'dark patterns (DPs)' in 2010. The detection and classification of dark patterns in user interfaces have garnered significant attention from researchers and practitioners in the fields of human-computer interaction (HCI), computer science, and digital ethics [9]. This section provides a comprehensive exploration of existing research efforts in the context of detecting hidden tactics of deceptive designs and identifies gaps that motivate my focus on scalable, generalizable solutions for real – world deployment.

### 2.1. Automated DPs Detection Techniques

Umar et al. [9] have provided a solution to detect DPs in user interfaces using Logistic Regression (LR) and Bag-of-Words Representation (BoWR). In another study, Mathur et al. [8] propose an automated systematic approach to identify dark patterns on large scale e-commerce websites. They utilized Bag of Words (BoW) representation, Principal Component Analysis (PCA), and the Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) algorithm for feature engineering with clustering. Soe et al. [2] suggest a machine learning workflow comprising feature engineering, parameter search, gradient boosted tree classifier training, and evaluation to forecast dark patterns on cookie banners. In a similar study carried out by Ramteke [10], the development of an automated approach that combines web scraping techniques with contextual understanding through fine-tuned BERT language models for identifying outlier dark patterns was highlighted. Inspired by the emerging machine learning techniques, Yada et al. [11] examined four machine learning models, including BERT, RoBERTa, ALBERT, and XLNet. At the end of the 5-fold cross-validation, it achieved superior accuracy with RoBERTa. Most subsequent studies focused on exposing taxonomies of dark patterns without large-scale evidence demonstrating their prevalence across a wide range of websites.

In 2023, Constâncio et al. [12] conducted a systematic review of deception detection with machine learning, which highlights the performance of several machine learning techniques, including neural networks, support vector machines, random forests, decision trees, and K-nearest neighbors. Also, they employed monomodal, bimodal, and multimodal approaches for their study.

This study builds on these efforts by combining Word2Vec vector representations with deep learning architectures, focusing on capturing nuanced semantic features to detect deceptive design patterns [13].

### 2.2. Multimodal Approaches

Using computer vision and natural language processing techniques, another study [14] presented a new automated system dubbed "AidUI" that can identify the presence of a set of distinct DPs. They combined text analysis, color

analysis, and semantic analysis to achieve the final solution. Recently, a domain-independent holistic approach to deception detection was developed using deep learning architecture [15]. With an overall accuracy of 93%, their approach beat the State-of-the-Art (SOTA) performance of the majority of benchmark datasets.

This work differs in the use of a hybrid deep learning model to concatenate local and sequential semantic features. Furthermore, unlike the aforementioned multimodal approaches, this work is a text-centric approach.

### 2.3. Taxonomies and Real-World Prevalence

Gray et al. [16] and Di Geronimo et al. [17] proposed a taxonomy that classified DPs into distinct categories based on their characteristics and user impacts. Hidaka et al. [18] investigated DDs in Japanese apps using that taxonomy as a basis. Additionally, Zagal et al. [19] established the notion of dark patterns in games and discussed some of the nuances involved in detecting them that can be used to help guide the development and identification of future dark patterns. Another study [20] examined the frequency of dark patterns in mobile games that exploit players on a temporal, monetary, social, and psychological basis.

Two distinct scenario-based experiments were carried out by Kim [1] to examine the moderating effects of deceptive design techniques and social proof. The study also finds that dark patterns have a negative impact on consumer attitudes and perceptions of justice. By addressing a taxonomy that transitions from a problem-oriented perspective to a problem-solving framework, Saville [21] suggested a novel method for DP detection.

Same as these studies, our models aim for superior and scalable performance in dark pattern detection. Beyond that, in this study we concentrate on generalizable detection rather than taxonomy expansion.

This study explores methods and developments in identifying dark patterns (DPs) and deceptive designs using machine learning and deep learning approaches. However, understanding the presence and effects of DPs in the real world remains challenging due to limited empirical evidence and reliance on fictitious statistics. There is an urgent need to create sophisticated deep learning algorithms capable of detecting and neutralizing misleading designs in user interfaces, hence preserving user autonomy and decision-making authority.

## 3. Methodology

This research is conducted in two main parts: model training, which is the primary focus, and performance evaluation as the secondary aspect. A Jupyter notebook was utilized for all machine learning tasks. Figure 1 illustrates the structure of the method that is being used.

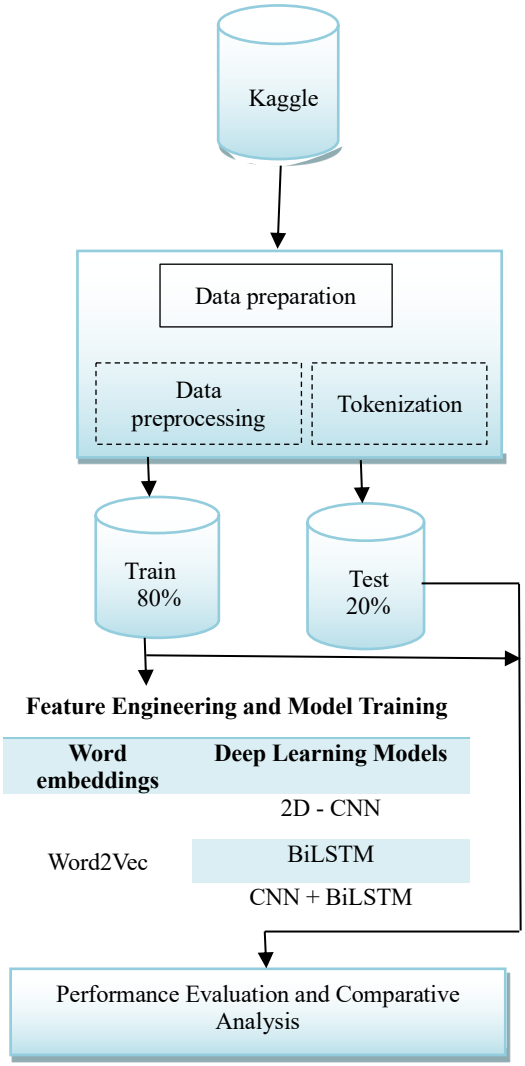


**Figure 1.** Framework for proposed methodology.

### 3.1. Dataset

The dataset in this study was sourced from previous research by Mathur et al. [9], which also utilized datasets from earlier studies [8], [11]. The initial dataset included 1818 deceitful texts from 1254 online retail websites, which were chosen based on their popularity in the online shopping market. The websites were chosen to represent a massive diversity of online websites. The dataset was built in 2019. By using the non-dark pattern texts scraped from the same e-commerce websites,

they balanced their dataset. They scraped text from websites using the puppeteer library. The absence of manipulative or deceptive features allowed the identification of non-deceptive content. They employed a manual validation step to ensure that selected texts did not have features equivalent to deceptive designs. In this dataset there are two labels in the class column as Dark Patterns (DP) and Non-Dark Patterns (non-DP).

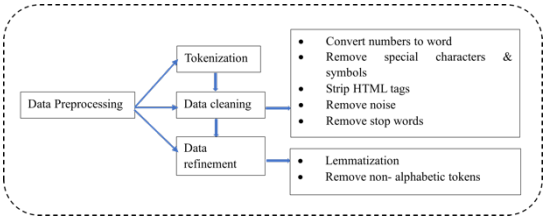


**Figure 2.** Preprocessing pipeline.

### 3.2. Data Preprocessing

Data preprocessing is a step in which we transform and prepare raw data into a form relevant to the data mining procedure. This stage aims to reduce the data size, find the relationship between data, remove outliers, and extract features for data [18]. Below, I describe the critical steps involved in this data preprocessing with the help of Figure 2.

1. *Tokenization:* Tokenization is the most substantial and initial stage in natural language processing (NLP). It entails dividing entire textual input into discrete units called tokens. The needs of the analysis will determine whether the tokens are words, phrases, or even characters. Tokenization was done in this study using ToktokTokenizer [22]. Transforming unstructured text into a structured format that deep learning models can handle is the main goal of tokenization.

2. *Data Cleaning:* The initial step in the data cleaning process is text normalization. In this stage, we transformed all the numbers within the row text into their textual representation using two libraries called re and inflect. This stage ensures it amalgamates numerical information into the textual data in a semantic manner. The subsequent step in the data cleaning phase considered adding required spaces before corresponding capitals. Here, regex recognized uppercase transitions and added spaces as applicable. The next phase of the data-cleaning process entailed the removal of symbols, specialized characters, and punctuation to convert text into a standardized form. This was kept only alphanumeric characters and a few essential punctuations (., /-) using regex text clean patterns [23]. Strip HTML tag removal was the next step in the data cleaning process, and the bs4 library was utilized to complete this task. This helped to filter the dataset, removing HTML tags and including only the scraped data that was most pertinent for analysis and interpretation. The next step addressed the removal of noise from the text. Using the re and bs4 libraries, this step executed a high-level function that integrates HTML tag stripping and square bracket removal. Stemming is a process of removing the associated affixes from a word to reduce it to its standard form, or root [19]. The NLTK's PorterStemmer was employed to filter out these root words, ensuring that only the most essential word forms remained [24]. The attention turned to eliminating common words (stopwords: is, the, and) from the textual data content after the remove stop words stage. This was achieved through the NLTK's stop words library, which identified and omitted common words that do not come up with much semantic value.
Then, any remaining inconsistencies in text formatting were addressed by converting all characters to lowercase and eliminating redundant spaces. At the completion of data preprocessing, the class column's categorical data were renamed as DP (Dark Patterns) and non-DP (Non-Dark Patterns) by using the replace function. The binary classification distinguishes between interfaces with deceptive patterns and non-deceptive patterns. An example of the preprocessed text obtained after completing all the preprocessing steps is described in Table 1.

**Table 1.** Example of preprocessed text.

| Input Text | Final Preprocessed Text |
|---|---|
| """This is an example of text data, including# numbers like 123 and abbreviations such= as e.g. or i.e. We need to preprocess this text for GAN-based text-to-speech conversion.""" | exampl text data, includ number like one hundr twenty-thre abbrevi exampl need preprocess text gan-bas text-to-speech convers. |

3. *Data Refinement*: As the concluding step, eliminate any words that are unable to contribute significant information to the context in the stage that deals with removing irrelevant terms by using specific techniques, including lemmatization and non-alphabetic token removal [24]. Additionally, the refinement step that handled both too-short and too-long textual inputs removed outliers that can skew analysis or hinder model performance. Handle these too-long and too-short words by using the min and max of word count. Figure 3 helps in understanding how the word count frequency is distributed after removing outliers.

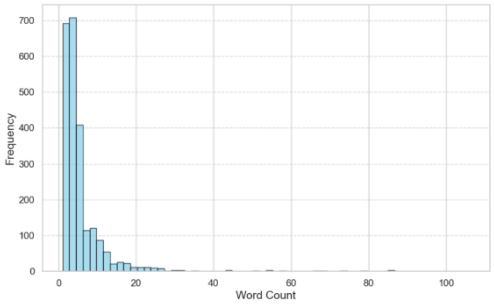


**Figure 3:** Word count frequency.

### 3.3. Exploratory Data Analysis (EDA)

This study's EDA step uses descriptive analysis to characterize the dataset's essential properties. Table 2 showcases the descriptive statistic parameters of a numerical variable in dataset. The following table briefly demonstrates statistical parameters including mean, standard deviation (std), minimum (min), percentage, and maximum (max). This section figures out some visual representations of data, including bar charts, heatmaps, bar plots, and histograms, highlighting trends, outliers, and relationships [25].

**Table 2**. Descriptive statistics for numeric variable.

| | page_id |
|---|---|
| Count | 2,356.00 |
| Mean | 904.33 |
| Std | 539.82 |
| min | 1.00 |
| 25% | 380.75 |
| 50% | 965.50 |
| 75% | 1,385.25 |
| max | 1,818.00 |

Figure 3 shows a bar chart showing the 30 most naturally occurring words in the deceptive texts in dark patterns. The analysis highlights the importance of threat-related and numerical words such as “left,” “one,” “stock,” “time,” and “limit.” Also, purchase-related words such as “purchase,” “offer,” “bought,” and “cancel” appear in high numbers, as these texts are likely to convey a transactional element. This bar chart is another examination of the DD practices, showing specific word patterns used to influence user manipulation. The distribution of words in a DD text is seen in Figure 4. It is evident that, in particular, the analyzed texts have fewer than 20 words, with the watershed being between 5-10 words.

This finding implies that DDs are typically created as brief messages to immediately capture the user's attention or create pressure. In addition, the distribution is characterized by a long sentence structure, with a small number of sentences lasting up to 100 words. This rightward shift emphasizes the appropriate brevity about fraudulent patterns.

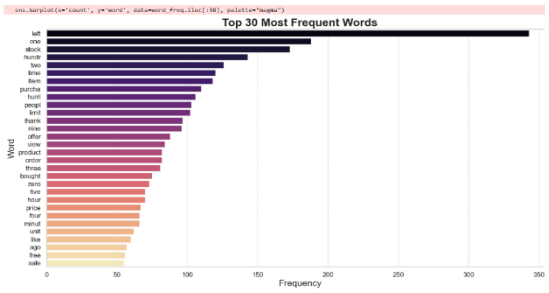


**Figure 3:** Top 30 Most Frequent Words in Deceptive Text

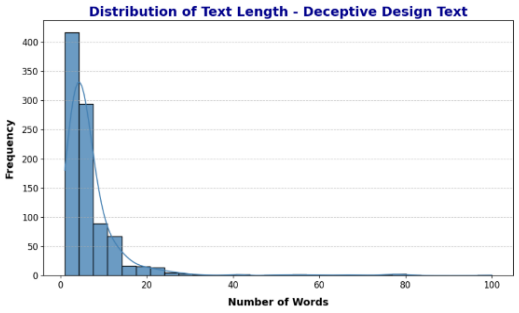


**Figure 4.** Distribution of Deceptive Text Length

### 3.4. Feature Engineering

In this study, Word2Vec embedding method was used to extract semantic features, with each word represented by a 300-dimensional feature vector. Pretrained Word2Vec embeddings from GoogleNews-vectors-negative300.bin were used, with embeddings loaded via the *Gensim* library. The dataset was tokenized with a maximum vocabulary size of 2000, and sequences were padded to ensure the model received a uniform input lengths for the model [26].

We preferred to employ Word2Vec in this study instead of BERT [10] due to its computational viability and ability to efficiently detect deceptive language patterns. Word2Vec is a lightweight word embedding technique that maps words into dense vector representations based on their co-occurrence in a corpus with much less computational power than BERT. Since deceptive design detection involves analyzing high-scale text-based patterns, BERT would have been necessary to use high GPU or TPU resources, which are not always readily available.

Although Word2Vec provides static word embeddings, like each word having an identical representation. While it lacks depth in contextual knowledge, it is effective in capturing semantic equivalences between words, an important feature to determine deceptive design patterns based on

intentional keyword-focused manipulation, like persistent insincere wording. Word2Vec embeddings are also more interpretable than transformer-based token representations from BERT so that the effects of which deceptive words and wording influence model decision-making most are more easily explored.

By combining Word2Vec embeddings with CNN and BiLSTM, we achieve balance in effectiveness and efficiency and thereby make it a viable choice for deceptive design detection and lay a good groundwork for subsequent studies.

### 3.5. Model Training

We employed three deep learning models - Convolutional Neural Network (CNN), Bidirectional Long Short-Term Memory (BiLSTM), and the combined model of these two models (CNN+BiLSTM). During the training phase, we split the dataset into a training subset and a test subset using Scikit-learn's train and test function [27]. Here, a training-to-test ratio of 80% is employed. The binary cross-entropy loss function and Adam optimizer were used to optimize these deep learning models [9].

$$Loss = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(p_i) + (1-y_i)\log(i-p_i)] \quad (1)$$

where $y_i$ is true label and $p_i$ is the predicted probability for the $i^{th}$ sample.

#### 3.5.1 Convolutional Neural Network (CNN)

Figure 5 shows a block diagram of the suggested Convolutional Neural Network (CNN) architecture for the task. The execution process steps of the DDs identification model using CNN are as follows:
The input layer receives a tokenized text sequence as input (length sequence_length).

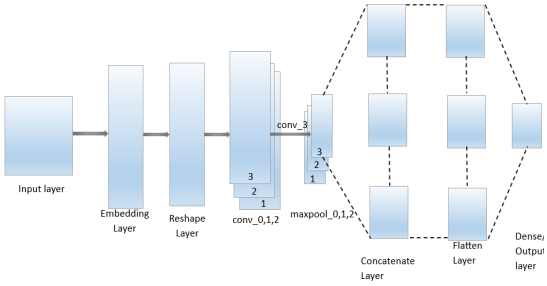


**Figure 5**. Block diagram for proposed Convolutional Neural Network

Using embedding sequences, the embedding layer turns these token indices into dense word vector representations. With EMBEDDING_DIM representing the word embedding dimension, the embedding layer's output shape is (sequence_length, EMBEDDING_DIM). The embedding tensor is changed by the reshaping layer so that the convolutional layers can use it. This is accomplished by adding a dimension for channel consistency.

$$R \in R^{(sequence\_length \times EMBEDDING\_DIM \times 1)} \quad (2)$$

This model's convolutional layers collect valuable n-gram features via embeddings. The model contains three transition layers with 100 filters each for the three, four, and five filter sizes. Each transition layer is designed to detect 100 unique properties. By applying *BatchNormalization* to each input, the convolutional layer's output is normalized, resulting in a stable and quick training process. The smooth, nonlinear Swish activation function is subjected to the normalized output:

$$Swish(x) = x.\sigma(x) \quad (3)$$

where:

$$\sigma(x) = \frac{1}{1+e^{-1}} \text{ is the sigmoid function} \quad (4)$$

The most important feature signals for each convolutional layer are recovered from the global maximum pooling layers by picking the highest value of each feature map. The model uses a concatenate layer to merge the outputs of the three global maximum convolutional layers into a composite feature representation. A flattened layer converts the resulting multi-dimensional tensor into a one-dimensional vector, which is then fed into the dense layer. A dropout technique is employed to avoid overfitting of the dense layer, and a sigmoid activation function is employed for binary classification, generating a probability for the two classes indicating the presence or absence of user manipulations.

#### 3.5.2 Bidirectional Long Short-Term Memory (BiLSTM)

The tokenized text sequence (with a sequence length of sequence_length) is sent to the initial layer of this BiLSTM model called the input layer. The embedding layer uses Word2Vec pre-trained embeddings to convert these token indices into dense word vector representations. This

embedding layer provides input to bidirectional LSTM (BiLSTM) layers. These BiLSTM layers require the model to process the text sequentially forward and backward in order to gather contextual information from the lines of text that come before and after. The BiLSTM layer helps the model consider word meaning by identifying the contextual meaning and helpful correlations of this text. Each BiLSTM in this model is followed by the use of dropout and batch normalization capabilities.

The output goes through fully connected dense layers after processing through the BiLSTM layers. The model can learn intricate patterns because of the Swish activation feature. For binary classification, the model's last layer is a dense layer with a sigmoid activation function. The probability ratio that the sigmoid function produces shows whether the user manipulation (dark patterns) associated with the two potential classes is present or not. The proposed BiLSTM model flows as Figure 6 illustrates.

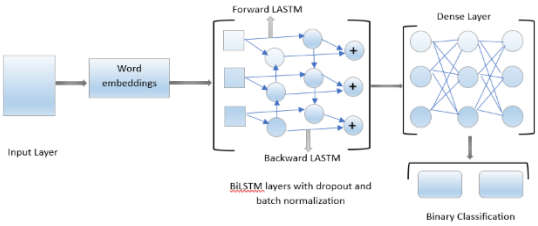


**Figure 6**. Block diagram for LSTM model baseline.

**Table 3.** Convolutional layers and their n-grams pattern detection capabilities.

| Convolutional Layer | Filter size/n-grams | Process | Output Shape |
|---|---|---|---|
| 01 | 3 | Detects patterns form 3 words collection | *(sequence_length-f+1,1, number of filters)* |
| 02 | 4 | Detects patterns form 4 words collection | |
| 03 | 5 | Detects patterns form 5 words collection | |

### 3.5.3 Hybrid model (CNN+BiLSTM)

According to Figure 7, the hybrid model performs both CNN and BiLSTM models simultaneously. In the combined model there were two input layers for the separate CNN block and BiLSTM block. Both inputs are sharing the same embedding layer.

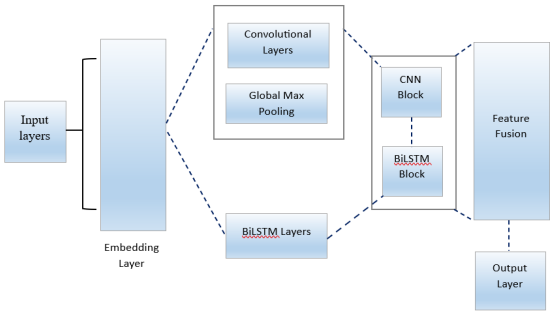


**Figure 7**. Block diagram for combined model baseline.

The convolutional operation is applied kernels of different filter sizes (k). Below are the equations for each kernel:

$$Z\_conv=Conv2D(Z\_embed,W\_conv,b\_conv) \quad (5)$$

where Z_embed stands for embedded input that has been reshaped. W_conv represents the learnable convolutional kernel of size k× embedding dimension. b_conv is representing a bias vector. Also, these convolutional layers contain a dropout function and batch normalization function. Batch normalization is computed through integration, as illustrated in the following equations:

$$Z_{norm}=\frac{Z_{conv}-\mu}{\sqrt{\sigma^2+\epsilon}}.\gamma+\beta \quad (6)$$

Global max pooling uses the following calculation to minimize the feature map after receiving input from the CNN block.

$$Z\_pool=max(Z\_norm,axis=1) \quad (7)$$

The following mathematical operation is used by the bidirectional LSTM block to identify contextual representation by operating the sequence in both forward and backward directions:

$$h_t=LSTM_f(x_t,h_{t-1})+LSTM_b(x_t,h_{t+1}) \quad (8)$$

Feature fusion performed concatenation of features extracted from the CNN block and the BiLSTM block using the below equation:

$$Z_{fused}=[Z_{pool}:h] \quad (9)$$

Sigmoid activation is utilized to construct the final classification using the fused features dense layer.

### 3.6. Performance Metrics

The trained deep-learning models were evaluated on a held-out test set to assess their performance in detecting DDs. We employed the standard evaluation metrics such as accuracy, precision, recall, and F1-score to measure the model's predictive performance.

*Accuracy:* Accuracy can be defined as the proportion of correctly identified samples to all samples [9].

$$Accuracy = \frac{Correctly\ classified\ samples}{Total\ number\ of\ samples} \quad (10)$$

*Precision:* Precision measures the ratio of correctly classified positive samples (true positives) to the total number of samples classified as positive (true positives + false positives) [9].

$$Precision = \frac{TP}{TP+FP} \quad (11)$$

Above,

The anticipated positive samples that correspond to the actual positive results are denoted as TP.

Predicted positive samples (FP) are those that don't match the actual negative results.

*Sensitivity:* Measures the proportion of accurately recognized true positives to all actual positive samples; also referred to as recall [9].

$$Sensitivity = \frac{TP}{TP+FN} \quad (12)$$

*F1-Score*: The F1-score offers a fair assessment of a model's performance since it is the harmonic mean of precision and sensitivity [9].

$$F1\text{-}score = \frac{2 \times Precision \times Sensitivity}{Precision + Sensitivity} \quad (13)$$

## 4. Implementation

### 4.1 Performance Evaluation

The performance of the three deep learning models on the tasks of deceptive design detection was rigorously evaluated using McNemar's paired test [28].This test is utilized to statistically compare the performance of three deep learning models. Figure 8 depicts how two models perform differently on the McNemar's test. Besides, it indicates that the hybrid outperforms the other two models. As shown there, the Bonferroni Threshold has been used for multiple comparisons in hypothesis testing to reduce the risk of false positives. In comparing the CNN and BiLSTM models separately, the p-value was greater than the Bonferroni value, indicating that their performance was not statistically significantly different. The hybrid model outperformed the CNN and BiLSTM models with p-values less than the Bonferroni value, which signifies a statistically significant improvement in performance.

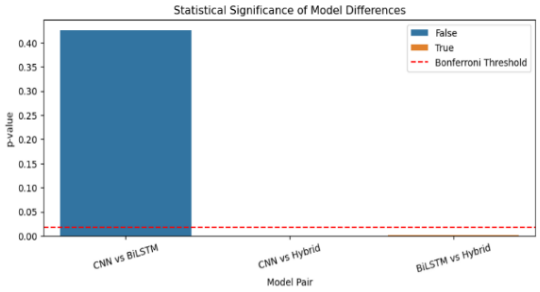


**Figure 8**. McNemar's paired test results.

Table 4 summarizes these evaluation metrics for each model on the utilized dataset. To assess the performance of proposed deep learning models, training loss was tracked across 30 epochs. It is proved that the hybrid model outperforms the others across all parameters, achieving superior superior performance. With an accuracy of 0.95 as opposed to 0.94 for BiLSTM and 0.93 for CNN, the hybrid model continuously beat the others. The hybrid model had the highest precision at 0.96, followed by BiLSTM at 0.93. A comparable pattern was seen in sensitivity, with the hybrid model reaching 0.95. These findings demonstrate the benefits of mixing sequential and convolutional designs to enhance classification performance.

**Table 4.** Overview of the model's performance.

| Model | Accuracy | Precision | Sensitivity | F1-score |
|---|---|---|---|---|
| CNN | 0.93 | 0.93 | 0.92 | 0.93 |
| BiLSTM | 0.94 | 0.93 | 0.92 | 0.93 |
| CNN+ BiLSTM | 0.95 | 0.96 | 0.95 | 0.95 |

### 4.2 Confusion Matrix

The confusion matrix provides a visual representation of the model's performance in

classifying instances as true positive (actual positive and predicted positive), false positive (actual negative and predicted positive), true negative (actual negative and predicted negative), and false negative (actual positive and predicted negative) [9]. The confusion matrices for all three model configurations are presented in the following figures.

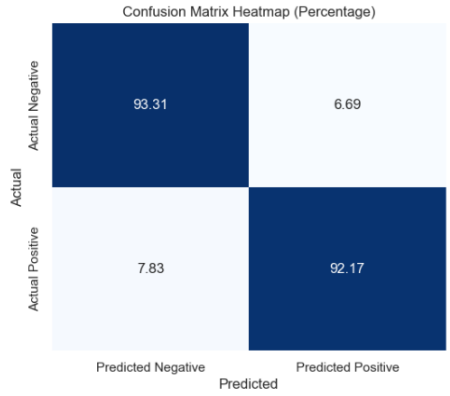


**Figure 9**. Confusion matrix for CNN.

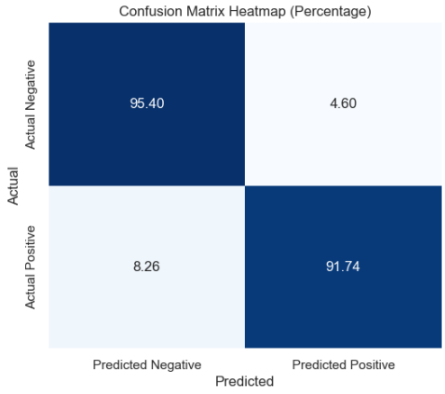


**Figure 10**. Confusion matrix for BiLSTM.

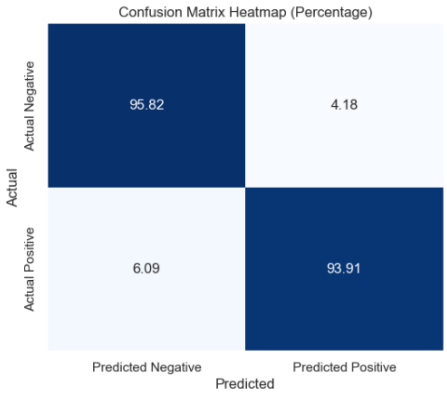


**Figure 11**. Confusion matrix for Hybrid model.

## 5. Discussion

In various domains like healthcare and financial, these deceptive design patterns can cause significant ethical and financial failures. As Figure 12 illustrates, with 1% accuracy over BiLSTM and 2% accuracy over CNN, the hybrid model is demonstrated to be better appropriate in this case for detecting deceptive designs. This approach works well in scenarios where real-time inference is not necessary, but it is less helpful for mobile apps, browser extensions, or edge devices due to its speed and performance constraints. Figure 13 directly addresses these efficiency trade-offs.

Additionally, this model might not be sensitive enough to social and cultural cues in misleading designs, which would prevent it from identifying subtle or indirect forms of language or dark pattern text. Furthermore, overfitting problems also occur, particularly with the hybrid model, whose test data accuracy decreases after multiple epochs, indicating that the model is overly focused on specific features in the training data. As a result, the model suffers when dealing with more varied data variants, like texts that are absent from the training set or have a different structure. More diversity in the training data, improved regularization methods, and enhanced preprocessing to accommodate more complicated and ambiguous texts are required to solve this problem [13].

A deceptive design strategy is not limited to textual context alone. However, we only take text-based designs into consideration in our study. In the future, we can investigate the use of a multi-fusion model in conjunction with other design strategies to detect false designs.

Second, this research solely uses a conventional word embedding technique. This is another study limitation. Future studies can examine how more sophisticated word and sentence embedding techniques function in the detection of deceptive designs.

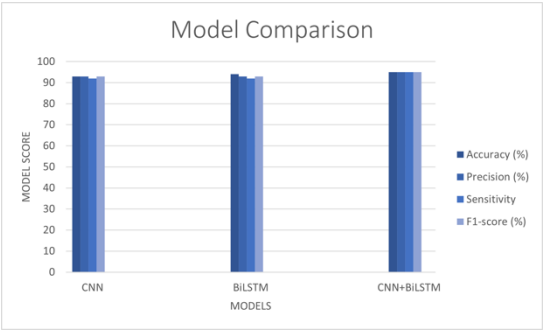


**Figure 12**. Model comparison.

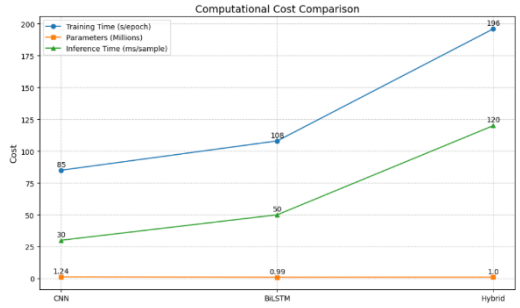


**Figure 13**. Computational cost comparison.

## 6. Conclusion

The primary goal of this study is to investigate how well deep learning models identify misleading designs. Using a hybrid model that blends CNN+BiLTM models, we have achieved notable performance by utilizing the Word2Vec model. Even while the CNN and BiLSTM models perform worse than the hybrid model when used alone, they are still very good.

The hybrid model proposed here combines Convolutional Neural Networks (CNN) and Bidirectional Long Short-Term Memory (BiLSTM) networks with Word2Vec embeddings to create a comprehensive approach to text classification that takes full advantage of both semantic and sequential feature extraction. The CNN part of the model is great at picking up local patterns and key phrases within text data, which makes it particularly effective at identifying fixed-length contextual dependencies. The BiLSTM part of the model, on the other hand, processes long-term dependencies really well and picks up on contextual features from both past and future sequences.

In summary, this research paper shows a method for finding deceptive designs in user interfaces using deep learning paradigms. By analyzing textual features and deep learning methods, the approach effectively identifies misleading design practices and helps users make intelligent decisions through online platforms. The results demonstrate that the mixed model successfully tells apart deceptive and non-deceptive UI patterns with strong metrics like accuracy, precision, recall, and F1-score. Analyzing feature importance sheds light on the language cues and patterns that signal misleading designs, aiding in understanding the model's predictions.

## References

[1] K. (Kathy) Kim, W. G. Kim, and M. Lee, “Impact of dark patterns on consumers’ perceived fairness and attitude: Moderating effects of types of dark patterns, social proof, and moral identity,” *Tourism Management*, vol. 98, p. 104763, Oct. 2023, doi: 10.1016/j.tourman.2023.104763.

[2] T. H. Soe, C. T. Santos, and M. Slavkovik, “Automated detection of dark patterns in cookie banners: how to do it poorly and why it is hard to do it any other way,” Apr. 21, 2022, *arXiv*: arXiv:2204.11836. doi: 10.48550/arXiv.2204.11836.

[3] A. R. Johnson, “Marketing Firm Affinion To Pay $30 Million in Multistate Settlement - WSJ.” Accessed: Jan. 22, 2025. [Online]. Available: https://www.wsj.com/articles/marketing-firm-agrees-to-30-million-settlement-1381441148

[4] Anon, “Deceptive Patterns (aka Dark Patterns) - spreading awareness since 2010.” Accessed: Jan. 12, 2025. [Online]. Available: https://www.deceptive.design/

[5] B. Shaffer, “Shopify Cracking Down On Fake Scarcity?,” Medium. Accessed: Jan. 22, 2025. [Online]. Available: https://benjshaffer.medium.com/shopify-cracking-down-on-fake-scarcity-e1509b11cb75

[6] D. Nazarov and Y. Baimukhambetov, “Clustering of Dark Patterns in the User Interfaces of Websites and Online Trading Portals (E-Commerce),” *Mathematics*, vol. 10, no. 18, Art. no. 18, Jan. 2022, doi: 10.3390/math10183219.

[7] C. M. Gray and S. S. Chivukula, “Ethical Mediation in UX Practice,” in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, in CHI ’19. New York, NY, USA: Association for Computing Machinery, May 2019, pp. 1–11. doi: 10.1145/3290605.3300408.

[8] A. Mathur *et al.*, “Dark Patterns at Scale: Findings from a Crawl of 11K Shopping Websites,” *Proc. ACM Hum.-Comput. Interact.*, vol. 3, no. CSCW, pp. 1–32, Nov. 2019, doi: 10.1145/3359183.

[9] A. Umar, M. Lawan, A. Lawan, A. Abdulkadir, and M. Dahiru, “Detecting Dark Patterns in User Interfaces Using Logistic Regression and Bag-of-Words Representation,” 2024. doi: 10.2139/ssrn.5050732.

[10] A. Ramteke, S. Tembhurne, G. Sonawane, and R. N. Bhimanpallewar, “Detecting Deceptive Dark Patterns in E-commerce Platforms,” May 27, 2024, *arXiv*: arXiv:2406.01608. doi: 10.48550/arXiv.2406.01608.

[11] Y. Yada, J. Feng, T. Matsumoto, N. Fukushima, F. Kido, and H. Yamana, “Dark patterns in e-commerce: a dataset and its baseline evaluations,” Nov. 12, 2022, *arXiv*: arXiv:2211.06543. doi: 10.48550/arXiv.2211.06543.

[12] A. S. Constâncio, D. F. Tsunoda, H. de F. N. Silva, J. M. da Silveira, and D. R. Carvalho, “Deception detection with machine learning: A systematic review and statistical analysis,” *PLOS ONE*, vol. 18, no. 2, p. e0281323, Feb. 2023, doi: 10.1371/journal.pone.0281323.

[13] R. Angger Saputra and Y. Sibaroni, “Multilabel Hate Speech Classification in Indonesian Political Discourse on X using Combined Deep Learning Models with Considering Sentence Length,” *Jurnal Ilmu Komputer dan Informasi*, vol. 18, no. 1, pp. 113–125, Feb. 2025, doi: 10.21609/jiki.v18i1.1440.

[14] S. H. Mansur, S. Salma, D. Awofisayo, and K. Moran, “AidUI: Toward Automated Recognition of Dark Patterns in User Interfaces,” Mar. 12, 2023, *arXiv*: arXiv:2303.06782. doi: 10.48550/arXiv.2303.06782.

[15] S. Shahriar, A. Mukherjee, and O. Gnawali, “A Domain-Independent Holistic Approach to Deception Detection,” in *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, R. Mitkov and G. Angelova, Eds., Held Online: INCOMA Ltd., Sep. 2021, pp. 1308–1317. Accessed: May 27, 2025. [Online]. Available:

https://aclanthology.org/2021.ranlp-1.147/

[16] C. M. Gray, Y. Kou, B. Battles, J. Hoggatt, and A. L. Toombs, "The Dark (Patterns) Side of UX Design," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, Montreal QC Canada: ACM, Apr. 2018, pp. 1–14. doi: 10.1145/3173574.3174108.

[17] L. Di Geronimo, L. Braz, E. Fregnan, F. Palomba, and A. Bacchelli, "UI Dark Patterns and Where to Find Them: A Study on Mobile Applications and User Perception," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, Honolulu HI USA: ACM, Apr. 2020, pp. 1–14. doi: 10.1145/3313831.3376600.

[18] S. Hidaka, S. Kobuki, M. Watanabe, and K. Seaborn, "Linguistic Dead-Ends and Alphabet Soup: Finding Dark Patterns in Japanese Apps," in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, Hamburg Germany: ACM, Apr. 2023, pp. 1–13. doi: 10.1145/3544548.3580942.

[19] J. Zagal, S. Björk, and C. Lewis, "Dark patterns in the design of games," presented at the International Conference on Foundations of Digital Games, 2013. Accessed: May 28, 2025. [Online]. Available: https://www.semanticscholar.org/paper/Dark-patterns-in-the-design-of-games-Zagal-Bj%C3%B6rk/19a241378b06d868eb5f6b76027172c3aaca86f4

[20] S. Niknejad, T. Mildner, N. Zargham, S. Putze, and R. Malaka, "Level Up or Game Over: Exploring How Dark Patterns Shape Mobile Games," in *Proceedings of the International Conference on Mobile and Ubiquitous Multimedia*, in MUM '24. New York, NY, USA: Association for Computing Machinery, Dec. 2024, pp. 148–156. doi: 10.1145/3701571.3701604.

[21] M. Potel-Saville and M. Da Rocha, "From Dark Patterns to Fair Patterns? Usable Taxonomy to Contribute Solving the Issue with Countermeasures," in *Privacy Technologies and Policy*, K. Rannenberg, P. Drogkaris, and C. Lauradoux, Eds., Cham: Springer Nature Switzerland, 2024, pp. 145–165. doi: 10.1007/978-3-031-61089-9_7.

[22] M. D. P. P. Goonathilake and P. P. N. V. Kumara, "Stance-Based Fake News Identification on Social Media with Hybrid CNN and RNN-LSTM Models," *Int J on Adv. in ICT for Emerging Countries*, vol. 16, no. 3, pp. 1–12, Dec. 2023, doi: 10.4038/icter.v16i3.7234.

[23] S. Biswas, D. Sengupta, R. Bhattacharjee, and M. Handique, "Text Manipulation Using Regular Expression," in *2016 IEEE 6th International Conference on Advanced Computing (IACC)*, Feb. 2016, pp. 62–67. doi: 10.1109/IACC.2016.22.

[24] D. Khyani, B. S. Siddhartha, N. M. Niveditha, and B. M. Divya, "An Interpretation of Lemmatization and Stemming in Natural Language Processing," Journal of University of Shanghai for Science and Technology. Accessed: May 28, 2025. [Online]. Available: https://jusst.org/an-interpretation-of-lemmatization-and-stemming-in-natural-language-processing/

[25] J. Simangunsong, M. S. Simanjuntak, and N. D. Simanjuntak, "Mental disorder classification with exploratory data analysis (EDA)," *idss*, vol. 7, no. 3, pp. 210–217, Jul. 2024, doi: 10.35335/idss.v7i3.252.

[26] H. Dinkel, M. Wu, and K. Yu, "Text-based depression detection on sparse data," Jul. 08, 2020, *arXiv*: arXiv:1904.05154. doi: 10.48550/arXiv.1904.05154.

[27] B. G. Bokolo and Q. Liu, "Advanced Comparative Analysis of Machine Learning and Transformer Models for Depression and Suicide Detection in Social Media Texts," *Electronics*, vol. 13, no. 20, Art. no. 20, Jan. 2024, doi: 10.3390/electronics13203980.

[28] M. Q. R. Pembury Smith and G. D. Ruxton, "Effective use of the McNemar test," *Behav Ecol Sociobiol*, vol. 74, no. 11, p. 133, Oct. 2020, doi: 10.1007/s00265-020-02916-y.