APPLIED INFORMATION TECHNOLOGY AND COMPUTER SCIENCE
e-ISSN: 2773-5141

AITCS

Vol. 6 No. 2 (2025) 494-508
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

DEEPhide: A Text Steganography Encoder and Decoder Tool using Deep Learning

Soo Kai Zhao[1], Shamsul Kamal Ahmad Khalid[1]*

[1] *Fakulti Sains Komputer dan Teknologi Maklumat,*
Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA

*Corresponding Author: shamsulk@uthm.edu.my
DOI: https://doi.org/10.30880/aitcs.2025.06.02.027

Article Info

Received: 12 July 2025
Accepted: 19 November 2025
Available online: 30 November 2025

Keywords

Text Steganography, Encoder, Decoder, Deep Learning.

Abstract

Text steganography is crucial in ensuring secure communication, but existing tools often lack efficiency in decoding concealed messages. This project presents DEEPhide, a deep learning-based encoder and decoder using CNN and RNN models to improve decoding accuracy and speed. A literature review highlights the need for automation and reliability in current methods. DEEPhide, developed using Agile methodology, was validated through unit, performance, and user testing. Results demonstrate high encoding/decoding success with fast performance and moderate robustness, showing DEEPhide's potential as a more secure and intelligent steganography solution.

1. Introduction

Text steganography is a field of study in information security that focuses on hiding the existence of a message within a text as its media. Unlike cryptography, a technique to encrypt the message to make it unreadable and hard to decrypt unless using the correct method, steganography conceals itself, so it is imperceptible to the reader but detectable by intended receivers [1]. As communication methods grow side by side with modern technology, ensuring the privacy and security of sensitive data during transmission becomes more crucial.

Existing text steganography techniques rely on predefined patterns. Traditional techniques such as word spacing and font manipulation are vulnerable to detection by modern steganography techniques [2]. This limitation exposes the hidden information to be discovered, affecting the confidentiality of the data.

Traditional text steganography techniques have shown limitations in handling large amounts of data while predefined patterns of existing techniques lead to the exposure of the hidden message. Existing systems are also constrained by the need for manual configuration, which results in inefficiency in the encoding and decoding of the text[3].

To overcome the limitations of traditional text steganography, DEEPhide is proposed. DEEPhide is a text steganography encoder and decoder tool that is installed with deep learning models. Deep learning is a division of machine learning, and it excels at many tasks with lower error rates compared to humans, making it suitable for automation [4]. Deep learning allows the system to intelligently encode and decode hidden information in the text, making it more resilient to detection. The automation for text steganography also increases efficiency while securing the confidentiality of the information [5]. This project contributes by developing DEEPhide with a hybrid CNN-RNN model for method prediction, custom-generated dataset for training, multiple encoding and decoding method provided, and simple yet user-friendly GUI for practical use in secure communication applications.

2. Related Work

This section presents a literature review of existing text steganography systems and deep learning-based approaches, focusing on their features, strengths, and limitations. The objective is to identify gaps in automation, decoding accuracy, and security robustness, and to position DEEPhide as an enhanced alternative that integrates modern deep learning techniques for improved performance.

2.1 Text Steganography

Text Steganography is a technique that hides secret information within text to ensure secure communication without drawing attention. Text steganography is less conspicuous than image or audio steganography as it embeds data in readable text. Most of the research on steganography utilizes cover media that includes videos, images and audios [1]. Unlike mentioned steganography techniques, text steganography is usually not given priority because of its difficulties identifying redundant bits in a text file [1]. However, the challenges met in text steganography can be improved or even overcome by automation using deep learning algorithms. Fig. 1 shows the traditional method of encoding a message using text steganography.

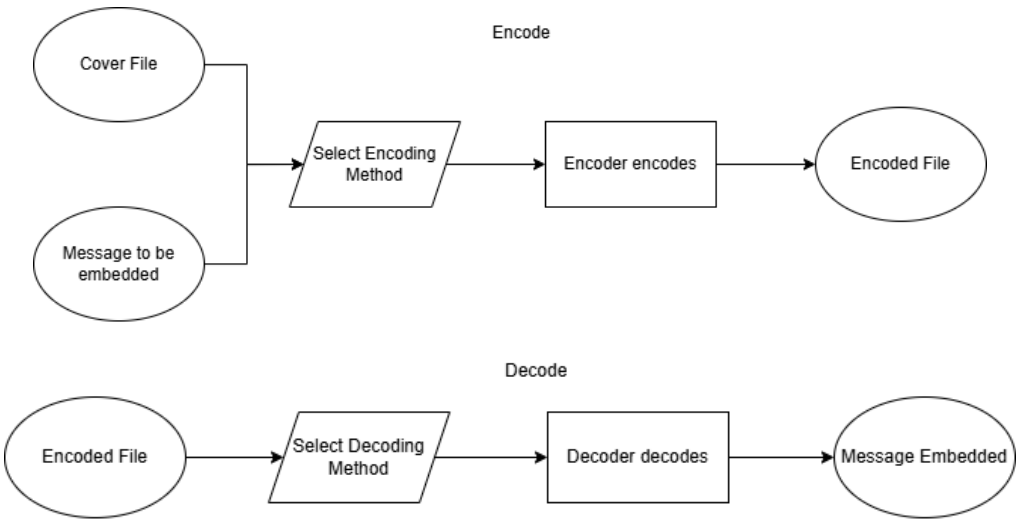


Fig. 1 *Traditional Text Steganoraphy Process [8]*

2.2 Types of Text Steganography

Text steganography can be split into three general classes: format-based method, linguistic method, and random and statistical generation [6]. In format-based method of text steganography, the physical features of text symbols are used, and they are altered in such a manner that the human eye cannot sense them [6]. In linguistic method of text steganography, it generates a steganographic paragraph with a specific topic based on knowledge graphs (KGs) that provides data about relevant topics and context to generate coherent multi-sentence text for better concealment [6]. In random and statistical generation, the statistical characteristics of a language are obtained before employed to generate cover text [7].

2.2.1 Line Shift Coding

Line Shift manipulates the line spacing of lines in a text file to hide data that are converted into ASCII coding before converting into binary form [8]. The shifting process can represent the binary "1" or "0" by slightly move up or down the text line. For example, the letter 'S' has ASCII coding of 83, which are 1010011 in binary, can be embedded in the text file by slightly moving up the line on the second, fourth and fifth line while slightly moving down on the rest of the lines. However, this method may easily fail if the embedded text is rewritten using different text processors or uses Optical Character Recognition (OCR) which uses the technology to distinguish printed or handwritten text characters [8].

2.2.2 Word Shift Coding

Word Shift Coding is a technique that alters the ordinary text's word spacing [8]. This method slightly moves to the right or to the left to conceal "0" or "1" just like Line Shift Coding. This method is simple for masking but it also easy to identify the masking process by using a different text editor, uses Optical Character Recognition (OCR) technology, or by comparing the original text to the embedded text [8].

2.2.3 Whitespace Steganography

Whitespace steganography conceals secret messages within the whitespace characters of a text file [9]. By altering the frequency or arrangement of whitespace character, the hidden messages in binary form can be embedded within a text file. These alterations cannot be detected by naked human eyes but those aware of the encoding technique can easily retrieve the concealed information.

2.2.4 Text Encoding with Special Fonts

By designing new fonts with special properties, it can be used to hide secret information. [13] Properties such as capacity, Size Increasing Ratio (SIR) and invisibility can be altered to create a unique font to embed into a plain text[8]. Capacity refers to the amount of secret data that can be hidden within the text using the custom font; Size Increasing Ratio (SIR) indicates the changes of the font size when the secret message is embedded; Invisibility refers to how the embedded message unnoticed within plain text.

2.2.5 Invisible Characters

Invisible Unicode characters can hide secret data in its cover text [15]. Invisible transmission of confidential information in such a way that no one can detect it. True invisible characters are ideal for steganography purposes as it enable embedding of information without altering text appearance.

2.3 Deep Learning Algorithm

Deep Learning (DL) is a part of machine learning (ML), which is also known as Artificial Intelligence (AI). Machine learning is used in data analysis and computing to allow the application to function intelligently [16]. Deep learning is inspired by the structure of the human that works by teaching computers and devices logical functioning that are similar to machine learning [17]. Deep learning uses artificial neural networks (ANNs), computational algorithms that are used to model data by using a set of processing neurons [18]. It recognizes patterns and learns automatically in deep architectures in order to make decisions [17]. Deep learning algorithms are able to learn from huge amounts of data to be used for big data analysis [17]. Fig. 2 shows the decoding method after implementing deep learning in the encoded decoding file for text steganography.

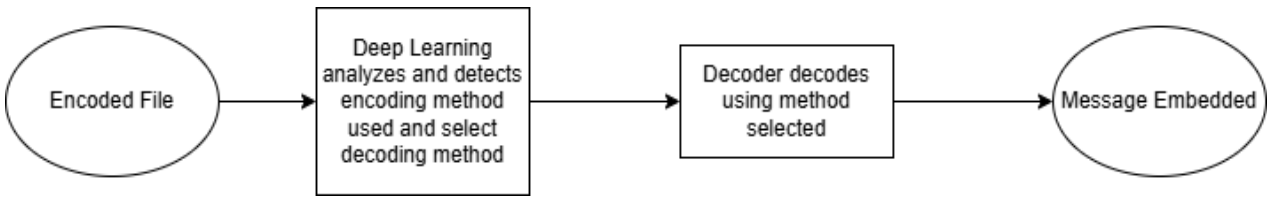


Fig. 2 *Decoding Process of Text Steganography using Deep Learning [17]*

2.3.1 Encoder-Decoder Architecture

The encoder-decoder model, also known as Sequence-to-Sequence model (seq2seq) is often used for machine translation, text summarization and other tasks [19]. By processing data in a two-stage setup: first encoding information into a hidden representation, then decoding it into readable output, this architecture is suitable for sequential prediction tasks[19].

2.3.2 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are primarily associated with image processing. It made many impressive achievements in many areas which includes but not limited to computer vision and natural language processing [20]. However, they have been used to detect complex patterns within structured text data. CNNs are effective in encoding additional information because of their high pattern recognition capabilities, essential for embedding data without affecting text structure or readability.

2.3.3 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks are ideal for sequential data, which are suitable for handling text in steganography. It can effectively model the dependencies within language, which is crucial for creating coherent steganographic text output if the state information over text sequences is well maintained. RNNs offer process of inferring on long sequences but they are hard to optimize and slow to learn something [21].

2.3.4 Comparison Between Existing Tools and DEEPhide

Table 1 compares the advantages and disadvantages of each existing tools and DEEPhide. Recurrent Neural Networks are ideal for sequential data, which are suitable for handling text in steganography. It can effectively model the dependencies within language, which is crucial for creating coherent steganographic text output if the state information over text sequences is well maintained. RNNs offer process of inferring on long sequences but they are hard to optimize and slow to learn something [22].

Table 1 *Comparison Between Existing Tools and Proposed Tool*

Features Tools	StegCloak[15]	OpenStego[16]	OpenPuff[17]	DEEPhide
Steganography Type	Text-based	Image-based	Multi-format (image, audio, video)	Text-based
Data Capacity	Low	Moderate	High	Moderate
Ease of use	Very easy, CLI-based	GUI, simple	Moderate, format understanding	Moderate
Support Format	Text only	Images	Image, Audio, Video, etc.	Text only
Robustness	Low	Low	High	Medium
Platform support	Node.js (Cross-platform)	Java (Cross-platform)	Windows only	Python-based
Complexity	Low	Low	High	Medium
Main Limitation	Low capacity, weak robustness	Limited to images	High complexity	Limited data capacity, limited supported file type
Advantages	Simple, Text-based, Quick, Unobtrusive	Image-based and Support password protection	Support various file types (images, audio, and video). Robust encryption capabilities	Text-based, Improved security, Improved embedding capacity
Disadvantages	Limitation in terms on data capacity and robust against modification	Minimal support for text-based steganography	More complex. Require familiarity with different media formats. Offers limited ease of use	Limited data capacity and limited detection vulnerability

3. Methodology

The software development life cycle selected as the guide for developing DEEPhide is the Agile model. Agile methodology has the benefits of increasing efficiency, improving quality, reducing the risks, improving collaboration, and increasing agility [23].

3.1 Software Development Workflow

The development of DEEPhide is distributed into six phases: requirements phase, design phase, development phase, testing phase, deployment phase, and review phase. These phases work as a guideline while developing DEEPhide. Table 2 shows the workflow of software development.

Table 2 *Software Development Workflow*

Phase	Task	Output
Requirement	1) Gather and analyze the user needs and software specifications 2) Identify functional and non-functional requirements 3) Decide core functionalities 4) Plan a timetable for the development of the software	1) Proposal 2) Gantt Chart

Table 2 (cont.)

Phase	Task	Output
Design	1) Create software architecture	1) Software architecture design
	2) Create user interface (UI) and user experience (UX) mockups	2) User interfaces
	3) Develop case diagrams, and user sequence diagram of the software	3) Use case diagram, and user sequence diagram of the software
Development	1) Study and implement deep learning algorithms	1) DEEPhide encoding and decoding tool using deep learning
	2) Write source code on interfaces of the software	
Testing	1) Alpha testing	1) Test results
	2) Beta Testing	2) Feedback
	3) Software Testing	3) Fine-tuned encoder and decoder tool
	4) Benchmarking Testing	
	5) Adjustment of DEEPhide	
Deployment	1) Push DEEPhide through GitHub platform	1) Deployment of software
	2) Manual attached with deployed software	2) User training and manual
Review	1) Gather feedback	1) Ideas to improve proposed software
	2) Analyze feedback	

3.2 Software Architecture

Fig. 3 depicts the architecture of DEEPhide tool, which is divided into two key modules: Encoder Module and Decoder Module. Encoder modules implement non-deep learning-based text-technology techniques. While Decoder Module employs a combination of deep learning algorithms including CNNs for feature extraction, RNNs for sequence learning and Transformer for classifcation and message extraction. Deep learning will be used to analyze the method used in the encoded files and learn the pattern in order to select the most suitable decoder to extract message.The encoding process starts with the user uploading a cover file (docx) and a message file (.txt). The selected encoding method manipulates the cover file and saves the encoded file (docx) in a designed location.The decoding process involves uploading the encoded file, where the system will use trained model to analyze the steganography techniques used. The upload file undergoes preprocessing, feature extraction using CNNs, sequence analysis using RNNs, and classification with a Transfomer decoder to extract the hidden message and save in the designed location as a text (.txt) file. Fig. 3 shows the DEEPhide Tool architecture ensures seamless integration of both traditional encoding techniques and advanced decoding mechanisms.

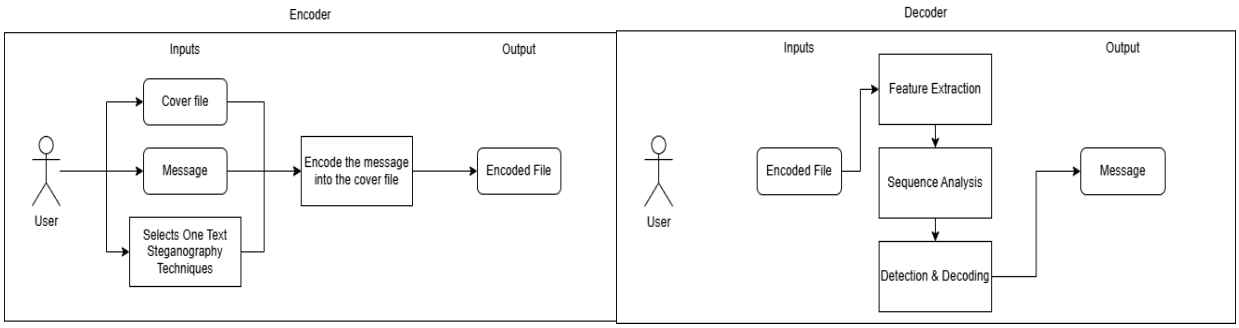


Fig. 3 *DEEPhide Tool Architecture*

The training process for DEEPhide starts by generating their own dataset using the encoders implemented in DEEPhide. This is because there are no ready datasets in the field of text steganography suitable for model training. Data preprocessing cleans and prepares data for training. The usage of deep learning is in feature extraction, sequence learning and model training. Feature extraction uses Convolutional Neural Network (CNN) to learn feature representations from the input data. Sequence learning uses Recurrent Neural Network (RNN) to understand sequential relationships in the features. Model training uses Transformer decoder to classify the encoding type and extract the message. Validation monitors model performance during training to prevent overfitting, while testing evaluate the trained model on unseen data. Model evaluation finds out the accuracy of

the model, and performance evaluation shows the efficiency of the model. Finally, the deployment finalizes the trained model for use in the decoding module. Fig. 4 is the process of the model training for DEEPhide.

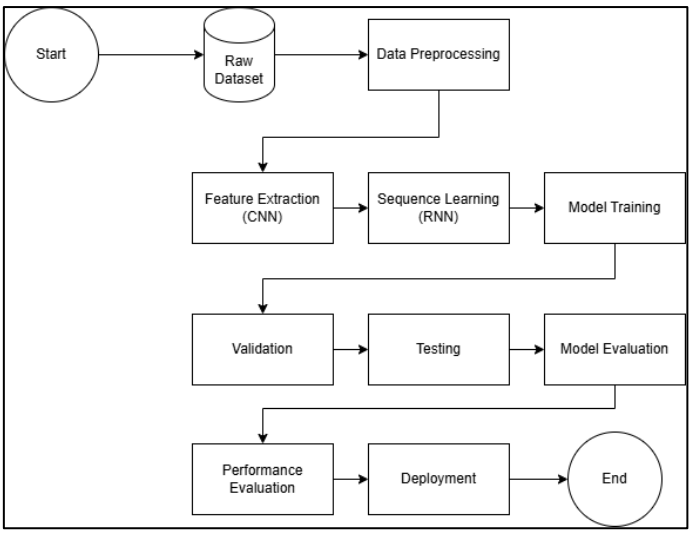


Fig. 4 *Model training process for DEEPhide*

3.3 System Requirements

The requirements of the DEEPhide tool are derived by considering the needs of users who want to encode and decode messages in DOCX files using text steganography techniques. The requirements ensure the tool is both functional and user friendly while meeting performance, and scalability expectations. Table 3 shows the functional requirements, Table 4 shows the non-functional requirements, Table 5 is the software specifications, and Table 6 is the hardware specifications of DEEPhide tool.

Table 3 *Functional Requirements for DEEPhide*

Functional Requirement	Description
File Upload	Allow users to upload cover and message files for encoding or an encoded file for decoding.
Encoding Methods Selection	Provide options for users to select one of the provided encoding methods.
Encoding Process	Perform selected text steganography encoding and generate an encoded file.
Automatic Decoding	Automatically detect the encoding method and extract the hidden message using deep learning model.
File Saved	Enable the save of the decoded message extracted from encoded files.

Table 4 *Non-Functional Requirements for DEEPhide Tool*

Non-Functional Requirement	Description
Performance	Ensure encoding and decoding operations are completed within a very short period.
Usability	Provide a clear, simple and intuitive interface for users with minimal technical expertise.
Reliability	Ensure accuracy in encoding and decoding, maintaining consistency across all supported methods.
Scalability	Support increasing file sizes and complexity without degrading performance. Allow the addition of new text steganography techniques.

Table 5 *Software specifications*

Software	Function
Python 3.8+	Programming language for implementing the DEEPhide.
Tkinter	Graphical User Interface (GUI) framework for building the desktop application
Draw.io	Illustrate the architecture diagram, deep learning process, and Unified Modeling Language (UML) diagram
Figma	Visualize the user interface of DEEPhide

Table 6 *Hardware specifications*

Hardware	Specification
Processor	Intel i5 or higher
RAM	8GB or higher
GPU	NVIDIA CUDA-compatible GPU for deep learning computations
Operating System	Window 11

3.4 Unified Model Language (UML)

The Unified Modeling Language (UML) provides a set of diagrams to visualize the design and the interaction of the DEEPhide tool. These diagrams help communicate between the software's architecture and workflows effectively, ensuring all stakeholders have a clear understanding of the tool's functionalities and operations.

Fig. 5 shows the use case diagram illustrates the interaction between the user and the DEEPhide tool. It focuses on the core functionalities of encoding and decoding text steganography. Users can upload cover files and message files for encoding or encoded files for decoding. The tool processes the files based on the selected encoding method and automatically detects the decoding type using deep learning. The output is then provided to the user by saving it into a text file in designed location.

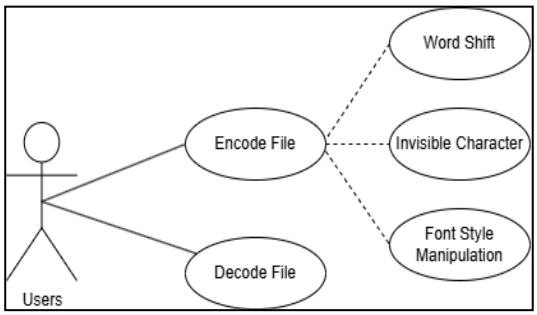


Fig. 5 *Use Case Diagram of DEEPhide Encoder & Decoder*

Fig. 6 shows the User Sequence Diagram provides a detailed flow of operations within the DEEPhide tool. It depicts how a user action (e.g., selecting files and clicking the "Encode" or "Decode" button) triggers the tool's backend processes.

Workflow for Encoding:

1) The user selects cover and message files.
2) The GUIManager passes the selected files to the Encoder.
3) The Encoder applies the chosen text steganography method and generates an encoded file.
4) The result will pop-up and the encoded file is saved in the designed directory.

Workflow for Decoding:

1) The user uploads an encoded file.
2) The GUIManager sends the file to the Decoder.
3) The Decoder uses deep learning models to detect the encoding type and extract the message.
4) The result will pop-up and decoded message is saved in a text file in the designed directory.

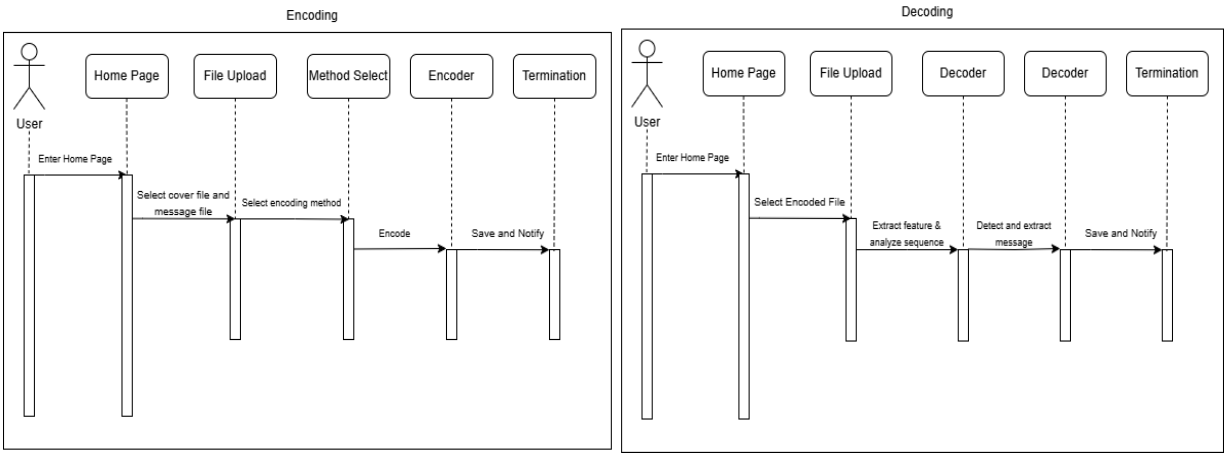


Fig. 6 *User Sequence Diagram of DEEPhide*

3.5 User Interface Design

The user interface (UI) for DEEPhide is designed for simplicity and efficiency. It ensures that users, regardless of technical expertise, can encode and decode files seamlessly. The home screen serves as the main navigation hub, providing access to encoding and decoding functionalities. Fig. 7 provides a sample user interface of DEEPhide. Users can easily select the desired operation:

1. Encoding Section: Upload fields for the cover file and message file, drop down menu to select an encoding method and "Encode" button to start processing.
2. Decoding Section: Upload field for the encoded file and "Decode" button to start the automatic detection and extraction process.

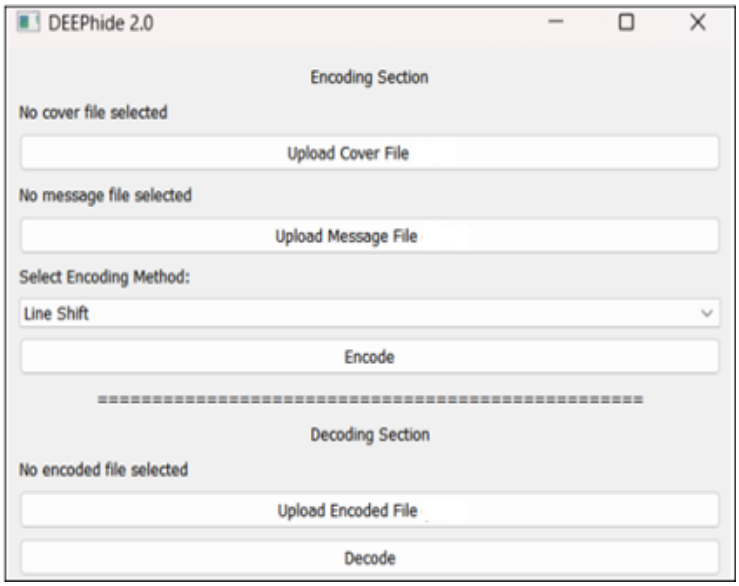


Fig. 7 *Home Page of DEEPhide with Encoding Section and Decoding Section*

4. Result and Discussion

This section conducts different types of testing on DEEPhide: A Text Steganography Tool Encoder and Decoder Tool using Deep Learning and record each result in the table.

4.1 Unit Testing

Unit testing is carried out to verify that individual components of DEEPhide such as functions, modules and others work correctly in isolation. For DEEPhide, the test units are encoding modules, decoding modules, deep learning models, and file input & output operations. The test results are shown in Table 7.

Table 7 *Unit Testing of DEEPhide Tool*

No	Test List	Result	Description
1	Encoding Modules	Pass	Functions responsible for preparing input messages, interacting with the selected steganographic algorithm and generating the steganographic output. The message is successfully encoded into the cover file using every algorithm provided in DEEPhide.
2	Decoding Module	Pass	Functions responsible for processing steganographic text, interacting with relevant algorithms and extracting the original hidden message. The encoded message is successfully retrieved from the cover file using subsequent algorithm provided in DEEPhide.
3	Deep Learning Model Module	Pass	Function that handles the loading of the pre-trained deep learning models, preprocessing of input data for the model, and post-processing of the model's input. The deep learning model module successfully predicts the encoding method used in the encoded file by algorithm provided in DEEPhide. The model also successfully predicts the method used in Stegano tool [24].
4	File Input & Output Operations	Pass	Functions ensuring for reading cover texts and writing steganographic files. The files are successfully uploaded and generated files are saved in desired location.
5	Input Validation	Pass	Functions that ensure user inputs meet the required criteria such as cover files length. The required criteria of DEEPhide are verified and error messages are shown if the files failed to fulfil the criteria.

4.2 Performance Testing

Performance testing is a evaluation step that ensures DEEPhide function both correctly and efficiently while providing satisfactory user experience. The objective of this testing is to measure the speed and responsiveness of DEEPhide's core steganographic operations, including encoding and decoding, to access the system resource consumption during the deep learning operations and others. Table 8 is the key performance indicators (KPIs) monitored and the test results.

Table 8 *Performance Testing of DEEPhide Tool*

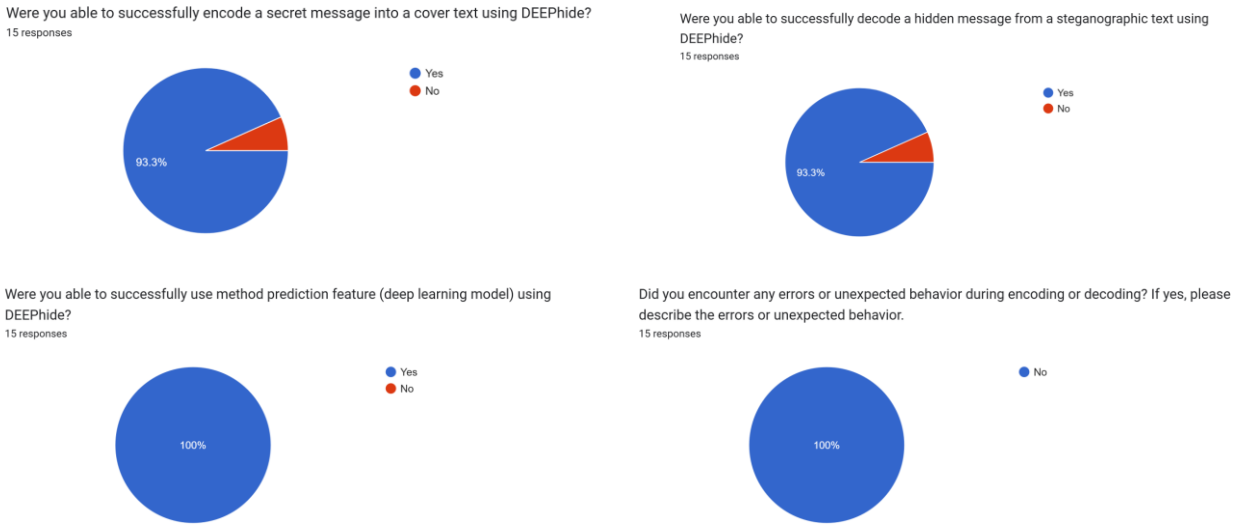
No	Test List	Result	Description
1	Encoding Time	Pass	The time taken to embed a secret message into a cover text and produce steganographic output file. The time taken for all 5 different methods is less than 5 seconds.
2	Decoding Time	Pass	The time taken to extract a secret message from a cover text and produce steganographic output file. The time taken for all 5 different methods is less than 5 seconds.
3	CPU Usage	Pass	Peak CPU utilization by DEEPhide application during prediction operations. The actual peak CPU usage is 17.7% during deep learning model inference.
4	Memory Usage	Pass	The peak RAM consumption by the application with particular to the memory footprint and utilizing deep learning model. The RAM usage without using deep learning is around 150MB, while peak RAM usage using deep learning comes around 180MB.
5	Tool Loading Time	Pass	The time taken to load any needed modules including deep learning models into memory before they can be used for steganographic operations. It takes less than 15 seconds to open DEEPhide tool before ready to use.
6	GUI Responsiveness	Pass	Qualitative assessment of the user interface's responsiveness during processing rasks, ensuring it remains interactive and does not freeze. The GUI remained responsive during encoding and decoding operations but takes longer for deep learning operations.

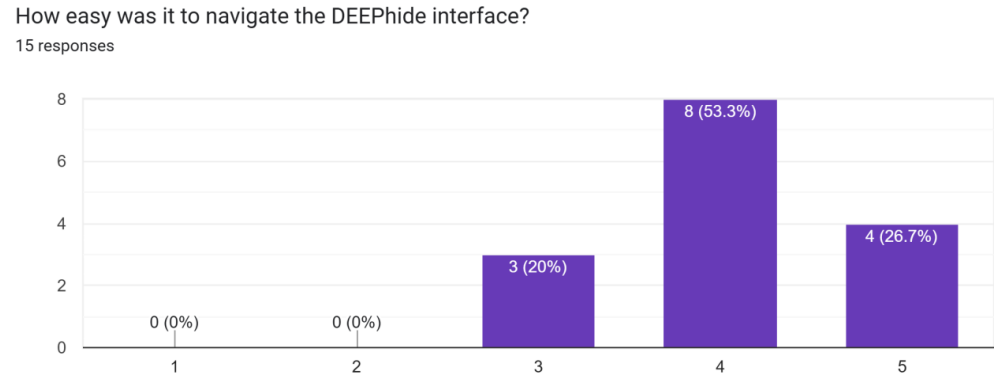
4.3 User Acceptance Testing

User Acceptance Testing (UAT) is the final phase of the software testing lifecycle, conducted to ensure that DEEPhide meets the requirements and expectations of its end users. The primary objective of UAT is to evaluate that the software is ready for deployment and practical use, confirming that all functional and non-functional requirements have been fulfilled. UAT also serves as an opportunity to gather feedback from real users in production-like environment. Table 9 is the key performance indicators (KPIs) monitored. The results of each question are shown as charts in Fig. 8.

Table 9 *User Acceptance Testing of DEEPhide Tool*

No	Question asked	Answer
1	Were you able to successfully encode a secret message into a cover text using DEEPhide?	Yes or No.
2	Were you able to successfully decode a hidden message from a steganographic text using DEEPhide?	Yes or No.
3	Were you able to successfully use method prediction feature (deep learning model) using DEEPhide?	Yes or No.
4	Did you encounter any errors or unexpected behavior during encoding or decoding? If yes, please describe the errors or unexpected behavior.	Yes or No.
5	How easy was it to navigate the DEEPhide interface?	Scales from 1(very difficult) to 5(very easy).
6	Were the instructions provided clear and helpful?	Scales from 1(very unclear) to 5(very clear).
7	Did you find it easy to select files and configure encoding/decoding options?	Yes or No.
8	Were error messages (if any) clear and helpful?	Not applicable (no errors), Yes or No.
9	How would you rate the speed of encoding and decoding operations?	Scales from 1(very slow) to 5(very fast).
10	How would you rate the speed of method prediction (deep learning model) operations?	Scales from 1(very slow) to 5(very fast).
11	Did the application remain responsive during operations?	Scales from 1(never) to 5(always).
12	How satisfied are you with our overall experience using DEEPhide?	Scales from 1(very dissatisfied) to 5(very satisfied).
13	What did you like most about DEEPhide?	Open-ended Answers.
14	What did you dislike or find confusing about DEEPhide?	Open-ended Answers.
15	Do you have any suggestions for improvement or additional features?	Open-ended Answers.
16	Any other comments or feedback?	Open-ended Answers.





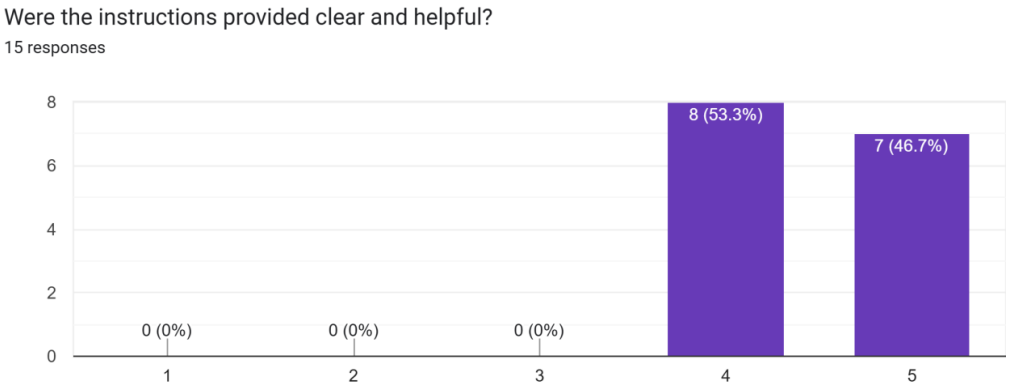


Fig. 8 *Results of User Acceptance Test*

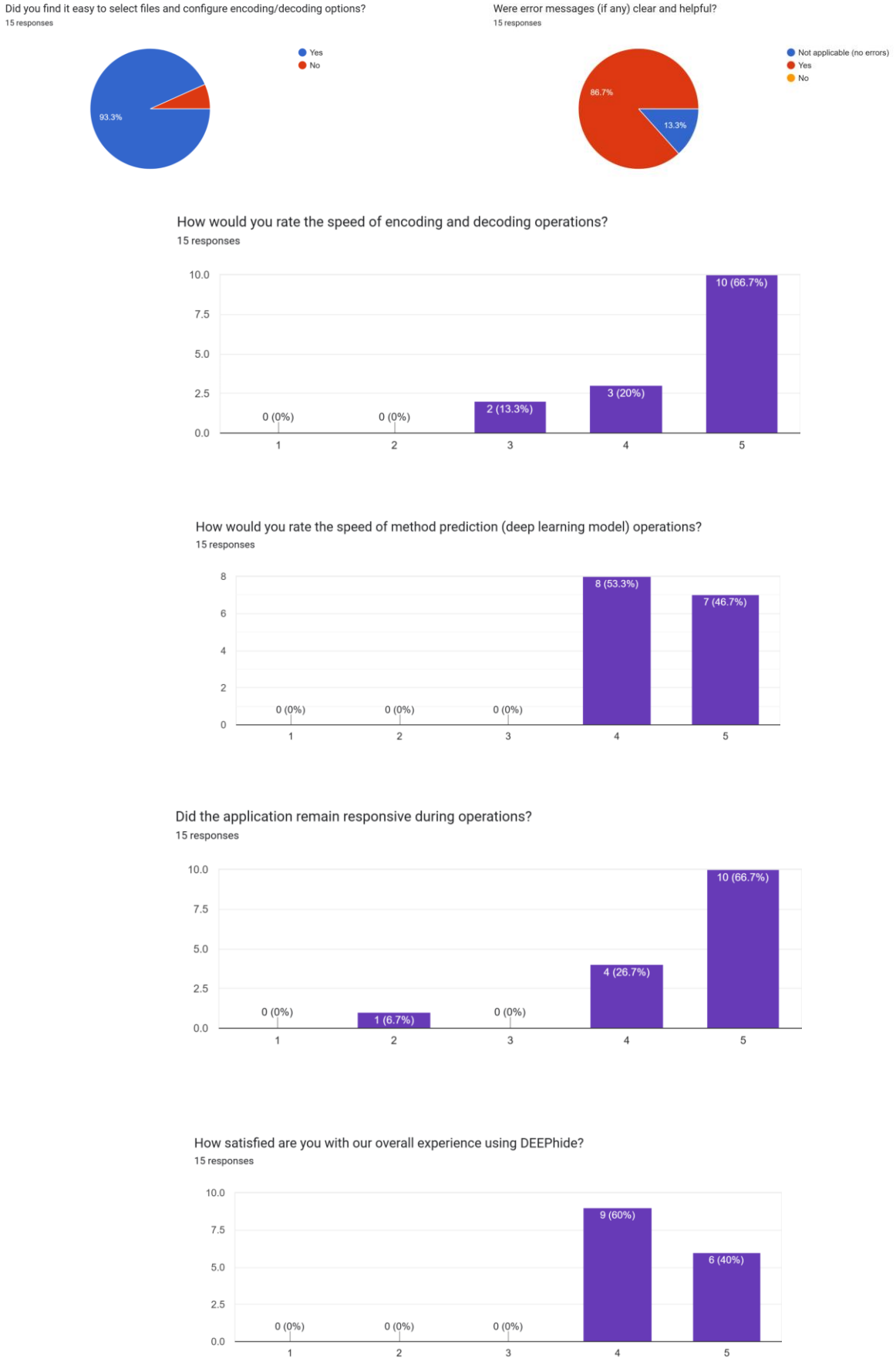


Fig. 8 *Results of User Acceptance Test(cont.)*

5. Conclusion

This section concludes with the development of DEEPhide. DEEPhide is a deep learning-based text steganography tool, consisting of traditional encoder and decoder, while implementing deep learning model to predict the method used. This section will revisit the project's objectives, summarizes the key achievements, evaluate the significance of the work conducted, and outlines possible directions for future research and development. The goal is to enhance the outcomes and assess their implications for the field of information security, particularly in text-based steganography.

5.1 Objective Achievement

The DEEPhide project successfully fulfilled its core objectives:

1. Design DEEPhide using deep learning models for method predictions. In DEEPhide, hybrid architecture combining CNNs, RNNs and Transformer models was implemented to accurately identify encoding methods and extract hidden messages from steganographic texts.
2. Development of an encoder and decoder tool. A fully functional GUI-based application was developed using Python and Tkinter. DEEPhide supports multiple encoding methods including LineShift, WordShift, Invisible Character, Font Variation, and Whitespace. Each encoding methods owns its decoding algorithm to extract secret message embedded in the cover file.
3. Benchmarking Testing. Through unit, performance, usability, and user acceptance testing, DEEPhide demonstrated reliable performance in encoding and decoding accuracy, operational speed and ease of use.

5.2 Project Significance

DEEPhide makes a significant contribution to the fields of steganography and secure communication. By integrating deep learning, DEEPhide can detect and predict the patterns of encoded files automatically. Its automation, adaptability, and robustness make it especially valuable in modern digital environments while stealth and security are crucial. DEEPhide is also platform-independent and moderate resource requirement that ensure accessibility for cybersecurity researchers, students, and forensics analysts.

5.3 Future Works

DEEPhide achieved its intended goals during this development. However, there are still room for improvement and expansion. The future work could focus on:

1. Enhancing Encoding Capacity. Investigate more advanced encoding strategies to increase the volume of data hidden without compromising text integrity.
2. Increasing Number of Encoding Method. DEEPhide only implements five different types of encoding method, which is the tip of an iceberg. The implementation of other encoding methods significantly increases the development of the tool, especially in the model training phase.
3. Support for Multilingual Texts. DEEPhide only support for alphabets cover files and English embedding message. Extend the support to non-English (e.g. Chinese) and multilingual texts to increase applicability in diverse communication scenarios.
4. Model Training with Better Datasets. DEEPhide was trained using a custom-generated dataset due to lack of publicly available datasets specific to text steganography. The dataset also does not include unencoded files, which make the tool fail to classify the file and show the correct prediction on the case. This limits the model's generalization and robustness. Future improvements should focus on acquiring or constructing a more comprehensive and diverse dataset that covers various steganographic patterns and natural language structures.

Acknowledgement

The authors would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia for its support.

Conflict of Interest

Authors declare that there is no conflict of interest regarding the publication of the paper.

Author Contribution

The authors confirm contribution to the paper as follows: **study conception and design**: *K. Z. Soo, S. K. Ahmad Khalid*; **data collection**: *K. Z. Soo, S. K. Ahmad Khalid*; **analysis and interpretation of results**: *K. Z. Soo, S. K. Ahmad Khalid*; **draft manuscript preparation:** *K. Z. Soo, S. K. Ahmad Khalid*. All authors reviewed the results and approved the final version of the manuscript.

References

[1] Kaspersky, "What is steganography? Definition and explanation," Kaspersky, Feb. 10, 2023. [Online]. Available: https://www.kaspersky.com/resource-center/definitions/what-is-steganography.

[2] M. A. Majeed, R. Sulaiman, Z. Shukur, and M. K. Hasan, "A review on text steganography techniques," Mathematics, vol. 9, no. 21, p. 2829, Jan. 2021. doi: 10.3390/math9212829.

[3] E. Ardhianto, H. L. H. S. Warnars, B. Soewito, F. L. Gaol, and E. Abdurachman, "Improvement of Steganography Technique: A Survey," www.atlantis-press.com, Mar. 06, 2020. https://www.atlantis-press.com/proceedings/imcete-19/125935468 (accessed Jul. 07, 2023).

[4] N. Kang, "Introducing Deep Learning and Neural Networks — Deep Learning for Rookies (1)," Medium, Feb. 04, 2019. [Online]. Available: https://towardsdatascience.com/introducing-deep-learning-and-neural-networks-deep-learning-for-rookies-1-bd68f9cf5883.

[5] Amazon, "What is Deep Learning? Deep Learning Explained - AWS," Amazon Web Services, Inc., 2023. [Online]. Available: https://aws.amazon.com/what-is/deep-learning/.

[6] A. Majeed, R. Sulaiman, Z. Shukur, and M. K. Hasan, "A Review on Text Steganography Techniques," Mathematics, vol. 9, no. 21, p. 2829, Jan. 2021, doi: https://doi.org/10.3390/math9212829.

[7] "A novel approach for linguistic steganography evaluation based on Artificial Neural Networks," IEEE Access, vol. 9, pp. 120869–120879, Aug. 2021. doi: 10.1109/ACCESS.2021.3099104.

[8] "Text Steganography Techniques: A Review," ProQuest, 2023. doi: 10.47001/IRJIET/2023.711085.

[9] A. Kumar, S. Tandon, S. Deorari, and R. Kumar, "Steganography: Unveiling techniques and research agenda," IntechOpen eBooks, Jul. 2024. doi: 10.5772/intechopen.1005052.

[10] S. Zhang, Z. Yang, J. Yang, and Y. Huang, "Linguistic Steganography: From Symbolic Space to Semantic Space," IEEE Signal Processing Letters, vol. 28, pp. 11–15, 2021, doi: https://doi.org/10.1109/lsp.2020.3042413.

[11] Dr. N. Arora, "Types and Tools of Steganography," International Journal for Research in Applied Science and Engineering Technology, vol. 10, no. 6, pp. 2049–2053, Jun. 2022, doi: https://doi.org/10.22214/ijraset.2022.44279.

[12] Mohammed Majid Msallam and None Fayez Aldoghan, "Multistage Encryption for Text Using Steganography and Cryptography," Journal of techniques, vol. 5, no. 1, pp. 38–43, Mar. 2023, doi: https://doi.org/10.51173/jt.v5i1.1087.

[13] S. S. Baawi, D. A. Nasrawi, and L. T. abdulameer, "Improvement of 'Text Steganography Based on Unicode of Characters in Multilingual' by Custom Font with Special Properties," IOP Conference Series: Materials Science and Engineering, vol. 870, no. 1, p. 012125, Jun. 2020, doi: https://doi.org/10.1088/1757-899x/870/1/012125.

[14] G. Niess and R. Kern, "Stylometric Watermarks for Large Language Models," arXiv.org, 2024. https://arxiv.org/abs/2405.08400 (accessed Nov. 14, 2024).

[15] E. A. Khan, "A novel approach to secure communication in mega events through Arabic text steganography utilizing invisible Unicode characters," PeerJ Computer Science, vol. 10, pp. e2236–e2236, Aug. 2024, doi: https://doi.org/10.7717/peerj-cs.2236.

[16] I. H. Sarker, "Machine Learning: Algorithms, Real-World Applications and Research Directions," SN Computer Science, vol. 2, no. 3, pp. 1–21, Mar. 2021, doi: https://doi.org/10.1007/s42979-021-00592-x.

[17] H. Abdel-Jaber, D. Devassy, A. Al Salam, L. Hidaytallah, and M. EL-Amir, "A Review of Deep Learning Algorithms and Their Applications in Healthcare," Algorithms, vol. 15, no. 2, p. 71, Feb. 2022, doi: https://doi.org/10.3390/a15020071.

[18] M. G. M. Abdolrasol et al., "Artificial Neural Networks Based Optimization Techniques: A Review," Electronics, vol. 10, no. 21, p. 2689, Jan. 2021, doi: https://doi.org/10.3390/electronics10212689.

[19] Z. Wang, X. Su, and Z. Ding, "Long-Term Traffic Prediction Based on LSTM Encoder-Decoder Architecture," IEEE Transactions on Intelligent Transportation Systems, pp. 1–11, 2020, doi: https://doi.org/10.1109/tits.2020.2995546.

[20] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects," IEEE Transactions on Neural Networks and Learning Systems, vol. 33, no. 12, pp. 1–21, 2021, doi: https://doi.org/10.1109/tnnls.2021.3084827.

[21] A. Orvieto et al., "Resurrecting Recurrent Neural Networks for Long Sequences," PMLR, pp. 26670–26698, Jul. 2023, Accessed: Nov. 14, 2024. [Online]. Available: https://proceedings.mlr.press/v202/orvieto23a.html

[22] L. Heryanti, Wiga Maulana Baihaqi, Ariska Nurul Habibah, and B. A. Kusuma, "Comparative Analysis of Openpuff and Openstego Tools," JINAV: Journal of Information and Visualization, vol. 5, no. 1, pp. 12–20, 2024, doi: https://doi.org/10.35877/454RI.jinav2327.

[23] ThinkSys Inc., "Agile Software Development Life Cycle: Methodology, Examples," thinksys.com, Oct. 03, 2023. [Online]. Available: https://thinksys.com/development/agile-software-development-life-cycle/.
[24] C. Bonhomme, "Stegano: A pure Python steganography module," GitHub repository, 2025. [Online]. Available: https://github.com/cedricbonhomme/Stegano. [Accessed: Jun. 9, 2025].