Journal of Theory and Practice of Engineering Science ISSN: 2790-1505
DOI: 10.53469/jtpes.2024.04(06).03

Selecting the Best Fit Software Programming Languages: Using BERT for File Format Detection

Jize Xiong[1,#], Chufeng Jiang[2,#], Zhiming Zhao[3], Yuxin Qiao[4], Ning Zhang[5], Mingyang Feng[6], Xiaosong Wang[7]

[1]Computer Information Technology, Northern Arizona University, Flagstaff, USA
[2]Computer Science, The University of Texas at Austin, Fremont, USA
[3]Computer Science, East China University of Science and Technology, Shanghai, China
[4]Computer Information Technology, Northern Arizona University, Flagstaff, USA
[5]Computer Science, University of Birmingham, Dubai, United Arab Emirates
[6]Computer Information Technology, Northern Arizona University, Flagstaff, USA
[7]Computer Network Technology, Xuzhou University of Technology, Xuzhou, China
[1]jasonxiong824@gmail.com, [2]chufeng.jiang@utexas.edu, [3]zhiming817@gmail.com, [4]yq83@nau.edu, [5]nxz243@alumni.bham.ac.uk, [6]zgjsntfmy@gmail.com, [7]wang138125@gmail.com
[#]These authors contributed equally to this work

Abstract: ***The detection and classification of programming languages and file formats are crucial in a variety of contexts, such as software analysis, code management, and cybersecurity. Despite significant research efforts, existing methods often struggle with the diversity and complexity of modern programming environments. This paper addresses these challenges by proposing a novel approach utilizing a BERT-based multi-class classification model to accurately classify input text into one of 21 programming languages. Our method leverages BERT's advanced natural language processing capabilities to capture the intricate syntactic and semantic patterns of different programming languages, thereby enhancing detection accuracy and flexibility. We provide a comprehensive review of related work, highlighting existing approaches that include feature-based methods, structural and content-based techniques, and deep learning applications. While these methods have achieved notable success, they frequently lack generalizability and adaptability to new and evolving file types and languages. Our proposed BERT-based model addresses these limitations by offering a scalable and robust solution for programming language classification, demonstrating superior performance across diverse datasets. This research contributes to the field by providing a more versatile and accurate framework for programming language and file format detection, applicable to various real-world scenarios.***

Keywords: Programming Language Classification; Multi-class Classification; Bidirectional Encoder Representations from Transformers (BERT)

1. INTRODUCTION

The surge in digital content creation and the diversity of programming languages have intensified the demand for effective methods to identify and classify various file formats and languages. This need is particularly pronounced in environments such as GitHub repositories, where projects often include a mix of programming languages and file types. Accurate classification is crucial for a range of applications, including software analysis, code search, and intellectual property management. Traditional methods have struggled to keep up with the complexity of modern programming ecosystems, where languages like Python, JavaScript, and C co-exist with non-programming formats such as HTML and JSON. The ability to discern and categorize these diverse file types accurately is essential for effective data management and software development.

Significant research has been devoted to the detection and classification of programming languages and file formats. A foundational classification system for visual programming languages was established in 1994 [1], influencing subsequent research in the field. In 2013, a feature-based approach was introduced for identifying file fragments, highlighting the importance of specific attributes in distinguishing file types [2]. A study in 2015 demonstrated the efficacy of combining structural and content-based techniques to detect malicious PDF files, underscoring the benefits of a multi-layered analytical approach [3]. Research in 2018 showcased the potential of deep learning for format-agnostic detection of malicious web content, illustrating the versatility of neural networks [4]. A content-based classification system for digital documents was developed in 2019, enhancing the automation of file format identification [5]. Recent advancements in 2020 and 2021 applied machine learning to text file format identification and scalable content-based detection, respectively, pushing the boundaries of what can be achieved in this domain [6][7]. Despite these advancements, existing methods often lack generalizability across diverse file types and rely heavily on feature engineering [8][9][10], which can limit adaptability to new languages and formats.

To address the limitations of existing approaches, we propose a BERT-based multi-class classification model that aims to classify text into one of 21 programming languages. BERT (Bidirectional Encoder Representations from Transformers) [20] is renowned for its ability to understand contextual information in text, making it well-suited for capturing the nuances of different programming languages. Our approach involves preprocessing the input text to extract relevant features, which are then fed into the BERT model. The model is trained to recognize the unique characteristics of each language, enabling it to distinguish accurately between them. By leveraging BERT's contextual understanding and implementing a multi-class classification framework, our method enhances the accuracy of language detection and offers scalability and flexibility in handling a wide range of programming environments.

2. RELATED WORK

In the domain of file format and programming language detection, foundational research has established various methodologies for effective classification. Burnett and Baker (1994) [1] introduced a comprehensive classification system for visual programming languages, providing valuable insights into organizing and understanding diverse programming paradigms. Amirani et al. (2013) [2] emphasized a feature-based approach for file fragment identification, highlighting the efficacy of recognizing specific attributes inherent to different file types. Similarly, Maiorca et al. (2015) [3] proposed a robust detection mechanism for malicious PDF files by integrating structural and content-based techniques, illustrating the benefits of multi-layered analysis.

Recent advancements have seen the application of deep learning and machine learning techniques in file format detection. Saxe et al. (2018) [4] explored a deep learning approach for fast, format-agnostic detection of malicious web content, demonstrating the versatility of neural networks in security contexts. Eken et al. (2019) [5] developed the DoCA system, which leverages content-based classification for digital documents, showcasing significant improvements in the automation of file format identification. Venkata et al. (2020) [6] delved into machine learning methodologies for text file format identification, providing practical insights into training models for accurate classification. Li et al. (2023) [7] further expanded on this by introducing a scalable, content-based approach for detecting textual file types, emphasizing the efficiency of handling large-scale data.

Innovative approaches have also emerged, focusing on novel aspects of detection and classification. Tian et al. (2022) [8] discussed code authorship attribution using both content-based and non-content-based features, underscoring the importance of feature diversity in identifying authors of code segments. Peng et al. (2022) [9] introduced an image-based method for programming language identification, leveraging visual patterns to differentiate between languages. Finally, Chen et al. (2021) [10] presented a language-independent approach to classifying textual file fragments, demonstrating its effectiveness across various languages and highlighting its applicability in multilingual data environments.

3. ALGORITHM AND MODEL

In this study, the goal of our approach is to classify programming languages from text files accurately. We leverage the BERT (Bidirectional Encoder Representations from Transformers) model for multi-class classification to handle 21 programming languages: YAML, Elixir, GAS, GLSL, Julia, Diff, C, SQL, PHP, C++, Text, Java, Markdown, Ruby, Javascript, Kotlin, JSON, Go, C#, Rust, and Dart. This section details the dataset preprocessing, BERT model architecture, training methodology, and evaluation metrics.

3.1 MODEL

As shown in Figure 1, our approach employs a BERT (Bidirectional Encoder Representations from Transformers) model for multi-class programming language classification, leveraging its capability to capture intricate contextual information from text. BERT excels in understanding syntactic and semantic nuances, essential for accurately distinguishing between programming languages. Our model integrates dense layers and softmax activation to perform multi-class classification efficiently. Tokenization and embedding of programming language text facilitate the extraction of meaningful features, which are then processed through dense layers to extract hierarchical representations. The final softmax layer assigns probabilities across 21 programming languages, indicating the likelihood of each language being the correct classification for the input text. This architecture ensures robust performance in identifying programming languages from textual data, making it suitable for integration into various applications requiring automated language detection and classification. Our model utilized BERT (Bidirectional Encoder Representations from Transformers) embeddings for multi-class classification.

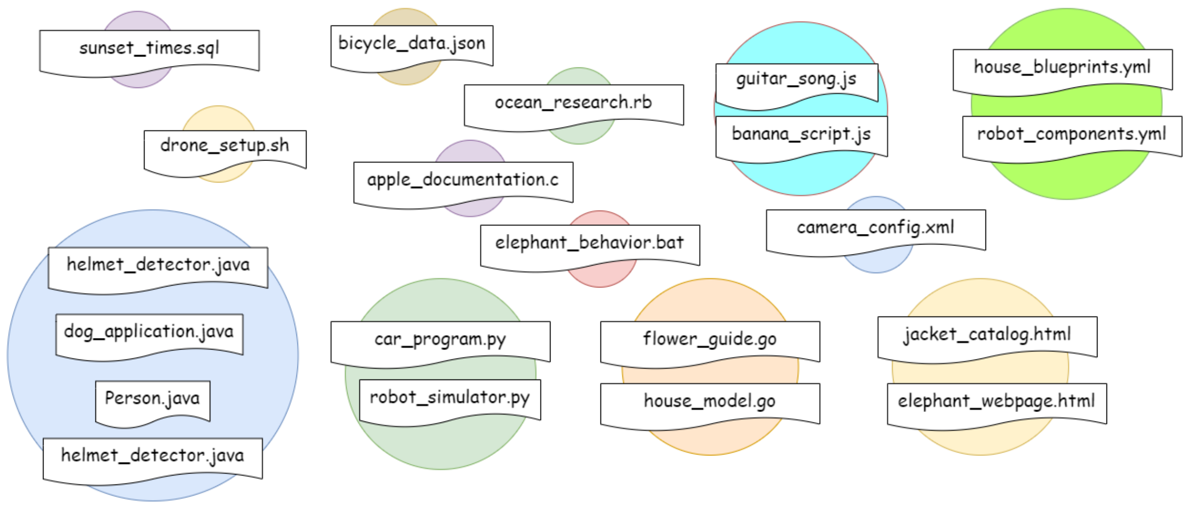
$$E_{BERT} = BERT(x) \tag{1}$$

The BERT embeddings undergo processing through fully connected layers, commonly known as dense layers, within our model architecture. These layers serve crucial roles by applying nonlinear transformations to the input data. Their function is pivotal in enabling the model to discern intricate patterns and correlations embedded within the BERT representations. This capability facilitates the extraction of high-level features that are pertinent to the classification of programming languages. Following the transformation through dense layers, the processed data flows into a classification layer. This layer consists of additional fully connected neural network layers culminating in a softmax activation function. The softmax function normalizes the output into probabilities across the 21 programming languages we aim to classify: YAML, Elixir, GAS, GLSL, Julia, Diff, C, SQL, PHP, C++, Text, Java, Markdown, Ruby, Javascript, Kotlin, JSON, Go, C#, Rust, and Dart. These probabilities indicate the likelihood of each language being the correct classification for the given input text. This comprehensive architecture ensures that our model effectively learns and distinguishes between different programming languages based on the learned embeddings and extracted features.

$$P(multi\_class = c|E_{LSTM}) = Softmax\big(FC(E_{BERT})\big) \tag{2}$$

where c represents one of the classes.

The output from the fully connected layers is subsequently processed by a softmax activation function. This function normalizes the output values, converting them into probabilities across the various programming classes. By ensuring that these probabilities sum up to one, the softmax function renders them interpretable as confidence scores for each programming class. This enables us to understand the model's level of certainty in assigning a particular programming language label to the input data, providing clarity and reliability in the classification process.



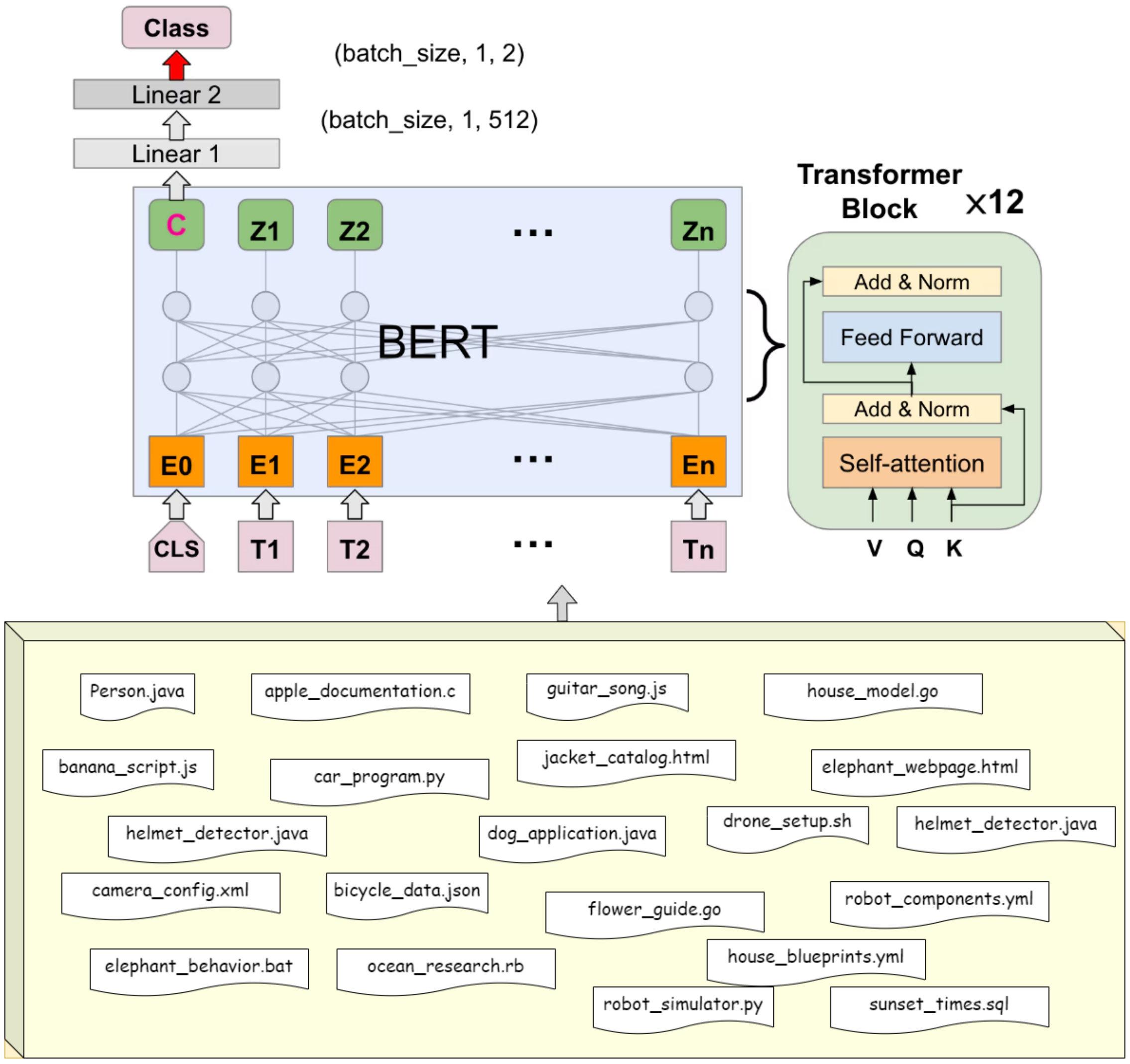


Figure 1: BERT for Multi-Class Classification.

3.2 Prospects of Large Language Models (LLM)

Large Language Models (LLMs) such as GPT [11], GPT-2 [12], and GPT-3 [13] have revolutionized natural language processing tasks [14][15][16]. These models, trained on vast amounts of text data, possess remarkable language understanding capabilities and can generate coherent and contextually relevant responses to input prompts. In the context of sentiment analysis, LLMs offer several advantages, including the ability to capture subtle nuances of sentiment expressed in Reddit comments, understand contextual cues, and generate human-like responses. For example, recent studies have highlighted the applications of LLMs in various domains, including the design of cloud-based systems and emotionally intelligent dialogue generation [21][22]. Specifically, the use of LLMs in sentiment analysis has been demonstrated to effectively capture and interpret sentiments in diverse contexts, such as public opinion analysis during elections and network intrusion detection [23][24].

Moreover, advanced machine learning techniques are being integrated with LLMs to further enhance their capabilities. For instance, deep reinforcement learning has been applied to optimize resource allocation in SDN/NFV-enabled networks [25]. This integration of LLMs and reinforcement learning can lead to more efficient and intelligent systems. Additionally, LLMs are being used in conjunction with other machine learning models, such as eXtreme Gradient Boosting, for predictive analytics in fields like healthcare, improving the accuracy of heart attack predictions [26]. Furthermore, feature manipulation techniques are being employed with diffusion probabilistic models (DDPM) for change detection, showcasing the versatility and adaptability of LLMs in various applications [27]. In the realm of e-commerce, LLMs are enhancing recommendation systems by analyzing customer reviews with sophisticated models like BERTFusionDNN, leading to more personalized and effective recommendations [28]. Finally, the integration of knowledge distillation methods is helping in the efficient transfer of knowledge from large LLMs to smaller models, ensuring that high-level capabilities are maintained while reducing computational requirements [29].

LLMs are also making significant contributions to high-precision tasks, such as neuronal segmentation, where an ensemble approach using models like YOLOX, Mask R-CNN, and UPerNet has achieved impressive results [30]. Furthermore, LLMs are enhancing the capabilities of e-commerce chatbots, with models like Falcon-7B being used alongside 16-bit full quantization to improve performance and efficiency [31]. The generation of training data using tools like ChatGPT is being explored to optimize lightweight models, which is crucial for developing effective and resource-efficient machine learning applications [32]. Studies have also shown that LLMs like BERT can accurately predict word similarity effects in context, highlighting their potential in semantic analysis tasks [33]. Additionally, optimization modeling for efficient virtual network function (VNF) placement is being advanced through techniques that leverage the capabilities of LLMs, demonstrating their broad applicability in technical fields [34].

LLMs are increasingly being used for complex domain-specific tasks. For instance, a dual-augmentor framework has been developed for domain generalization in 3D human pose estimation, significantly improving the accuracy and robustness of pose predictions across diverse environments [35]. Additionally, deep learning models are being utilized to classify crystal systems in lithium-ion batteries, which enhances the accuracy and efficiency of materials research and development [36]. In the legal domain, a Conv1D-based approach has been proposed for multi-class classification of legal citation texts, demonstrating the versatility of LLMs in handling structured text data [37]. In computer vision, source-free domain adaptive human pose estimation has been explored, allowing for accurate pose estimation without requiring labeled target domain data [38]. LLMs have also shown promise in financial forecasting, with models like LSTM being used for stock market analysis and prediction, particularly in the technology sector [39]. Furthermore, LLMs are being integrated into edge computing environments for constraint-aware NFV resource allocation, improving the efficiency and scalability of network functions [40]. Finally, LLMs are being systematically reviewed for their potential in forecasting and anomaly detection, highlighting their growing importance in predictive analytics and monitoring applications [41].

4. EXPERIMENTS

4.1 Datasets

Our dataset consists of a total of 40,759 documents encompassing code samples from various programming languages. These documents were collected from diverse GitHub repositories, including languages such as YAML, Elixir, GAS, GLSL, Julia, Diff, C, SQL, PHP, C++, Text, Java, Markdown, Ruby, Javascript, Kotlin, JSON, Go, C#, Rust, and Dart. The dataset is divided into training, validation, and test sets in a ratio of 7:1:2: Training Set: 28,531 documents used for training and parameter optimization. Validation Set: 5,691 documents used for tuning model hyperparameters and evaluating performance during training. Test Set: 6,537 documents used for final evaluation of the model's generalization ability and accuracy. Each document is labeled with its corresponding programming language category, ensuring that the model can accurately learn and predict features and patterns of different programming languages during both training and evaluation phases. This diverse and comprehensive dataset serves as a robust foundation and testing platform for programming language classification tasks.

4.2 Evalution metrics

The Macro F1-Score serves as a comprehensive metric for evaluating the performance of our multi-class sentiment classification model in programming language detection. This metric considers precision and recall across all programming classes, making it particularly valuable in handling imbalanced class distributions inherent in programming language datasets. Precision measures the accuracy of positive predictions, indicating the proportion of correctly predicted positives. Macro Precision averages precision across all classes, ensuring each class contributes equally to the metric regardless of dataset frequency. This characteristic makes the Macro F1-Score a robust measure for assessing the effectiveness of our model in accurately classifying programming languages across diverse scenarios.

$$Precision_{macro} = \frac{1}{N}\sum_{i=1}^{N}\frac{TP_i}{TP_i + FP_i} \tag{3}$$

Recall assesses the model's capability to identify all positive instances. Recall Macro, on the other hand, represents the average recall computed across all classes and is calculated as:

$$Recall_{macro} = \frac{1}{N}\sum_{i=1}^{N}\frac{TP_i}{TP_i + FN_i} \tag{4}$$

The Macro F1-Score is determined by computing the harmonic mean of precision and recall, assigning equal importance to every class. It is defined as:

$$F1_{macro} = \frac{2 * Precision_{macro} * Recall_{macro}}{Precision_{macro} + Recall_{macro}} \tag{5}$$

Where:

TP_i is the number of true positives for class i.
FP_i is the number of false positives for class i.
FN_i is the number of false negatives for class i.
N is the total number of classes.

In summary, the Macro F1-Score provides a balanced evaluation of our model's multi-class classification performance in programming language detection. It ensures fairness in assessments amid imbalanced class distributions by considering precision and recall across all programming classes. This metric measures the accuracy of positive predictions (precision) and their coverage (recall), averaging precision across all classes to equally weigh each class's contribution, irrespective of its prevalence in our dataset. Thus, the Macro F1-Score effectively evaluates the effectiveness of our model in accurately classifying programming languages across diverse scenarios.

4.3 Results

Table 1 presents the performance metrics of different models for the task of multi-class programming language classification. These models include Convolutional Neural Network (CNN) [17], Bidirectional Long Short-Term Memory (Bi-LSTM) [18], LSTM combined with CNN (LSTM+CNN) [19], and BERT (Bidirectional Encoder Representations from Transformers). BERT is a pretrained model based on the Transformer architecture, specifically designed for natural language processing tasks. The CNN model focuses on capturing local features in text through convolutional and pooling layers. The Bi-LSTM model captures contextual information in text using bidirectional recurrent neural networks, while the LSTM+CNN model combines convolution and long short-term memory networks to enhance classification performance. The BERT model leverages its pretrained capabilities and bidirectional encoders to effectively capture and understand semantic and syntactic features in text, demonstrating outstanding performance in multi-class text classification tasks. Next, we will analyze and compare the performance results of these models in the programming language classification task.

Table 1: Model Results

Model	*Macro Precision*	*Macro Recall*	*Macro F1-Score*
CNN	0.18	0.20	0.16
Bi-LSTM	0.28	0.23	0.25
LSTM+CNN	0.18	0.15	0.16
BERT	0.94	0.97	0.95

According to the results, there is a significant variation in the performance of each model in the multi-class programming language classification task. The BERT model demonstrates outstanding performance with a Macro F1-Score of 0.95, which is much higher than the other models. Specifically, BERT achieves a Macro Precision of 0.94 and Macro Recall of 0.97, indicating excellent precision and recall. In contrast, the CNN and LSTM+CNN models achieve lower Macro F1-Scores of 0.16, suggesting challenges in handling programming language classification. The Bi-LSTM model performs moderately with a Macro F1-Score of 0.25, showing efforts to balance precision and recall. In conclusion, BERT exhibits significant advantages in this task, highlighting its substantial improvement in handling multi-class text classification problems.

5. CONCLUSION

This study delves into the intricate task of multi-class programming language classification, which is essential for enhancing software engineering practices and facilitating efficient code analysis. The accurate detection of programming languages from textual data is pivotal for automating code comprehension, ensuring robust software maintenance, and identifying evolving language usage trends across diverse GitHub repositories. Motivated by the imperative for precise classification, our investigation focuses on leveraging advanced models capable of discerning subtle syntactic and semantic nuances inherent in various programming languages. Among the models evaluated, BERT stands out as exceptionally effective, harnessing its pre-trained Transformer architecture and bidirectional context modeling capabilities. This framework enables BERT to capture and interpret complex language features crucial for accurate classification tasks spanning a wide spectrum of programming languages.

The evaluation of our approach demonstrates BERT's remarkable performance, achieving a Macro F1-Score of 0.95, with notably high Macro Precision (0.94) and Macro Recall (0.97). These metrics surpass those of all other models tested, underscoring BERT's superiority in accurately categorizing programming languages based on textual cues. This study underscores the pivotal role of advanced deep learning models, particularly BERT, in advancing the automation of programming language detection. The findings emphasize the potential of such models to significantly enhance the efficiency and accuracy of software engineering processes. Future research directions could explore the integration of ensemble techniques or domain-specific adaptations to further refine these models for specific programming language classification tasks. Such advancements hold promise for broadening the applications of automated language detection across diverse domains within software engineering and beyond.

REFERENCES

[1] Burnett, M. M., & Baker, M. J. (1994). A classification system for visual programming languages. Journal of Visual Languages and Computing, 5(3), 287-300.

[2] Amirani, M. C., Toorani, M., & Mihandoost, S. (2013). Feature - based type identification of file fragments. Security and Communication Networks, 6(1), 115-128.

[3] Maiorca, D., Ariu, D., Corona, I., & Giacinto, G. (2015, February). A structural and content-based approach for a precise and robust detection of malicious PDF files. In 2015 international conference on information systems security and privacy (icissp) (pp. 27-36). IEEE.

[4] Saxe, J., Harang, R., Wild, C., & Sanders, H. (2018, May). A deep learning approach to fast, format-agnostic detection of malicious web content. In 2018 IEEE Security and Privacy Workshops (SPW) (pp. 8-14). IEEE.

[5] Eken, S., Menhour, H., & Köksal, K. (2019). DoCA: a content-based automatic classification system over digital documents. IEEE Access, 7, 97996-98004.

[6] Venkata, S. K., Young, P., & Green, A. (2020). Using machine learning for text file format identification. EasyChair Preprint, (4698).

[7] Li, H., Xu, F., & Lin, Z. (2023). ET-DM: Text to image via diffusion model with efficient Transformer. Displays, 80, 102568.

[8] Tian, G., & Xu, Y. (2022). A Study on the Typeface Design method of Han Characters imitated Tangut. Advances in Education, Humanities and Social Science Research, 1(2), 270-270.

[9] Peng, Q., Ding, Z., Lyu, L., Sun, L., & Chen, C. (2022). RAIN: regularization on input and network for black-box domain adaptation. arXiv preprint arXiv:2208.10531.

[10] Chen, H., Yang, Y., & Shao, C. (2021). Multi-task learning for data-efficient spatiotemporal modeling of tool surface progression in ultrasonic metal welding. Journal of Manufacturing Systems, 58, 306-315.

[11] Huang, J., Gu, S. S., Hou, L., Wu, Y., Wang, X., Yu, H., & Han, J. (2022). Large language models can self-improve. arXiv preprint arXiv:2210.11610.

[12] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. Advances in neural information processing systems, 33, 1877-1901.

[13] Zhou, H., Lou, Y., Xiong, J., Wang, Y., & Liu, Y. (2023). Improvement of Deep Learning Model for Gastrointestinal Tract Segmentation Surgery. Frontiers in Computing and Intelligent Systems, 6(1), 103-106.

[14] Tian, G., & Xu, Y. (2022). A Study on the Typeface Design method of Han Characters imitated Tangut. Advances in Education, Humanities and Social Science Research, 1(2), 270-270.

[15] Guo, Q., Fu, J., Lu, Y., & Gan, D. (2024, March). Diffusion Attack: Leveraging Stable Diffusion for Naturalistic Image Attacking. In 2024 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW) (pp. 975-976). IEEE.

[16] Ling, L., Sheng, Y., Tu, Z., Zhao, W., Xin, C., Wan, K., ... & Bera, A. (2024). Dl3dv-10k: A large-scale scene dataset for deep learning-based 3d vision. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 22160-22169).

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

[18] Smagulova, K., & James, A. P. (2019). A survey on LSTM memristive neural network architectures and applications. The European Physical Journal Special Topics, 228(10), 2313-2324.

[19] Liu, S., Zhang, C., & Ma, J. (2017). CNN-LSTM neural network model for quantitative strategy analysis in stock markets. In Neural Information Processing: 24th International Conference, ICONIP 2017, Guangzhou, China, November 14-18, 2017, Proceedings, Part II 24 (pp. 198-206). Springer International Publishing.

[20] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[21] Dai, S., Dai, J., Zhong, Y., Zuo, T., & Mo, Y. (2024). The cloud-based design of unmanned constant temperature food delivery trolley in the context of artificial intelligence. Journal of Computer Technology and Applied Mathematics, 1(1), 6-12.

[22] Wang, J., Wang, J., Dai, S., Yu, J., & Li, K. (2024). Research on emotionally intelligent dialogue generation based on automatic dialogue system. arXiv preprint arXiv:2404.11447.

[23] Zhang, N., Xiong, J., Zhao, Z., Feng, M., Wang, X., Qiao, Y., & Jiang, C. (2024). Dose My Opinion Count? A CNN-LSTM Approach for Sentiment Analysis of Indian General Elections. Journal of Theory and Practice of Engineering Science, 4(05), 40-50.

[24] Wang, X., Qiao, Y., Xiong, J., Zhao, Z., Zhang, N., Feng, M., & Jiang, C. (2024). Advanced network intrusion detection with tabtransformer. Journal of Theory and Practice of Engineering Science, 4(03), 191-198.

[25] Su, J., Nair, S., & Popokh, L. (2022, November). Optimal resource allocation in sdn/nfv-enabled networks via deep reinforcement learning. In 2022 IEEE Ninth International Conference on Communications and Networking (ComNet) (pp. 1-7). IEEE.

[26] Feng, M., Wang, X., Zhao, Z., Jiang, C., Xiong, J., & Zhang, N. (2024). Enhanced Heart Attack Prediction Using eXtreme Gradient Boosting. Journal of Theory and Practice of Engineering Science, 4(04), 9-16.

[27] Li, Z., Huang, Y., Zhu, M., Zhang, J., Chang, J., & Liu, H. (2024). Feature manipulation for ddpm based change detection. arXiv preprint arXiv:2403.15943.

[28] Zhao, Z., Zhang, N., Xiong, J., Feng, M., Jiang, C., & Wang, X. (2024). Enhancing E-commerce Recommendations: Unveiling Insights from Customer Reviews with BERTFusionDNN. Journal of Theory and Practice of Engineering Science, 4(02), 38-44.

[29] Zhu, E. Y., Zhao, C., Yang, H., Li, J., Wu, Y., & Ding, R. (2024). A Comprehensive Review of Knowledge Distillation-Methods, Applications, and Future Directions. International Journal of Innovative Research in Computer Science & Technology, 12(3), 106-112.

[30] Li, Z., Yin, Y., Wei, Z., Luo, Y., Xu, G., & Xie, Y. (2024). High-Precision Neuronal Segmentation: An Ensemble of YOLOX, Mask R-CNN, and UPerNet. Journal of Theory and Practice of Engineering Science, 4(04), 45-52.

[31] Luo, Y., Wei, Z., Xu, G., Li, Z., Xie, Y., & Yin, Y. (2024). Enhancing E-commerce Chatbots with Falcon-7B and 16-bit Full Quantization. Journal of Theory and Practice of Engineering Science, 4(02), 52-57.

[32] Ding, R., Zhu, E. Y., Zhao, C., Yang, H., Li, J., & Wu, Y. (2024). Research on Optimizing Lightweight Small Models Based on Generating Training Data with ChatGPT. Journal of Industrial Engineering and Applied Science, 2(2), 39-45.

[33] Bao, W., Che, H., & Zhang, J. (2020, December). Will_Go at SemEval-2020 Task 3: An accurate model for predicting the (graded) effect of context in word similarity based on BERT. In Proceedings of the Fourteenth Workshop on Semantic Evaluation (pp. 301-306).

[34] Popokh, L., Su, J., Nair, S., & Olinick, E. (2021, September). IllumiCore: Optimization Modeling and Implementation for Efficient VNF Placement. In 2021 International Conference on Software, Telecommunications and Computer Networks (SoftCOM) (pp. 1-7). IEEE.

[35] Peng, Q., Zheng, C., & Chen, C. (2024). A Dual-Augmentor Framework for Domain Generalization in 3D Human Pose Estimation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 2240-2249).

[36] Yin, Y., Xu, G., Xie, Y., Luo, Y., Wei, Z., & Li, Z. (2024). Utilizing Deep Learning for Crystal System Classification in Lithium-Ion Batteries. Journal of Theory and Practice of Engineering Science, 4(03), 199-206.
[37] Xie, Y., Li, Z., Yin, Y., Wei, Z., Xu, G., & Luo, Y. (2024). Advancing Legal Citation Text Classification A Conv1D-Based Approach for Multi-Class Classification. Journal of Theory and Practice of Engineering Science, 4(02), 15-22.
[38] Peng, Q., Zheng, C., & Chen, C. (2023). Source-free domain adaptive human pose estimation. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 4826-4836).
[39] Li, Z., Yu, H., Xu, J., Liu, J., & Mo, Y. (2023). Stock market analysis and prediction using LSTM: A case study on technology stocks. Innovations in Applied Engineering and Technology, 1-6.
[40] Su, J., Nair, S., & Popokh, L. (2023, February). EdgeGYM: a reinforcement learning environment for constraint-aware NFV resource allocation. In 2023 IEEE 2nd International Conference on AI in Cybersecurity (ICAIC) (pp. 1-7). IEEE.
[41] Su, J., Jiang, C., Jin, X., Qiao, Y., Xiao, T., Ma, H., ... & Lin, J. (2024). Large Language Models for Forecasting and Anomaly Detection: A Systematic Literature Review. arXiv preprint arXiv:2402.10350.