https://doi.org/149: 10.51153/kjcis.v6i1.149

A Smart Model for Categorization of GitHub Repositories

*[1]Muhammad Muneeb Aslam, [2]Muhammad Farhan, [3]Sana Yaseen,

[4]Ahmad Raza, [5]Muhammad Javed Iqbal, [6]Muhammad Munwar Iqbal

Abstract

There are several datasets of source code available on the World Wide Web. These files are usually grouped into application categories for programming languages. Various repositories of open-source code are now public on GitHub. Users can upload the source code they develop, distribute it to other users, allow other programmers to renew or change the program over time, and announce specific software applications. This study proposes a machine learning-based model for classifying source code. Machine learning algorithms are necessary to train and authenticate predictions of the required tasks. In the future, related software applications will be categorized into multiple source codes. Our machine-learning model trains you to organize your multiprogramming source code according to your problem. Training datasets can be obtained from GitHub. The main goal of this study is to learn different solutions for software classification. This research study aims to use source code to confirm the classification of multilingual software. The outcomes mentioned below in the result section appear to be quite encouraging. This method shows that the source code type can be completed using the earlier criteria. The software's great precision and quick response time make it ideal for the most realistic functions. The software had a 97 percent accuracy when identifying three programming languages.

Keywords: GitHub, Deep Learning, CNN, Categorization, Source Code

1. Introduction

Software classification and detection of related software make sense for several reasons, including replacing knowledge, realizing the application, and quick prototyping. The wide availability of open-source tasks and displaying those feature-based product updates on search engines requires automated classification and categories of similar techniques for software detection [1]. The open-source application repository, including SourceForge.net, contains many data source objects for coding and applications. It's easier to facilitate the search and exploration of these repositories, applications, and entities categorized into classifications, e.g., text editors, Antivirus, and warehouses [2]. Automatic ordering in several aspects helps with rapidly evolving software

[1]Department of Computer Science COMSATS University Islamabad, Sahiwal Campus l munbigee@gmail.com
[2]Department of Computer Science COMSATS University Islamabad, Sahiwal Campus l farhansajid@gmail.com
[3]Department of Computer Science COMSATS University Islamabad, Sahiwal Campus l sanayaseen009@gmail.com
[4]Department of Computer Science COMSATS University Islamabad, Sahiwal Campus l ahmadraza702905@gmail.com
[5]Department of Computer Science, University of Engineering and Technology, Taxila l javed.iqbal@uettaxila.edu.pk
[6]Department of Computer Science, University of Engineering and Technology, Taxila l munwar.iq@uettaxila.edu.pk

repositories, as mentioned above: Many similar applications can be grouped into one category for browsing convenience [3], for example, types of applications according to software foundries for related software, e.g., editors and databases.

As a result, it is difficult to know whether two APIs are accessible from two different APIs. Fourth, for various languages and compilers, the produced binary code is semantically and syntactically unique. As a result, current methods lack fewer cross-language identification capabilities than related systems. With the advancement in open-source platforms and various software programs, instruments that can identify and classify related applications across multiple programming languages are valuable [4].

Our research will explore the detection of source code types across multiple programming languages. We will have created such a model that will read source code as an input and determine which category this source code belongs to and the purpose it will be used. To create such an automated model, you need to download various source code models from GitHub and create a dataset. We will look at the source code manually to determine what tasks it performs and how it is written. With proper estimates, we shall put them in separate folders. We will give each folder a name when we separate folders, e.g., according to source code language and source code work. In this way, we would have prepared a data set. We will then submit that prepared data set into an automated model and put multiple algorithms on a data set to determine its accuracy. We will decide which algorithm gives more accuracy and will finalize that accurate algorithm.

One of the research gaps is classifying the source code repository with better accuracy and other statistical measures. If a programmer uploads a new source code into a GitHub repository, he only knows about the source code category and the purpose of the code. GitHub does not classify the uploaded code. It must be labeled using different keywords to train the machine learning models using keywords as categories and other features. Automatic classification can be performed using machine learning techniques. This research attempts to classify and identify the source code to determine the type and purpose.

2. Related Work

For source code categorization, different approaches have been suggested. Previous research has attempted to categorize the source code according to various patterns. Deep neural architectures (such as CNN and HAN) show the benefits of fully supervised text classification but cannot be adopted in some real-world scenarios due to the large amount of training data required. Numerous methods are mentioned in the literature review to resolve the difficulty of classifying the source code, as explained in Table 1.

Table 1: Related work of software categorization about authors' ideas and their datasets

Author	Idea	Datasets and Results
Katz, Deborah, et al., 2020 [5]	Software systems rely on no details other than the source code's automatically categories. The system automatically sets a software platform to be a part of several groups.	Implemented MUDA-Blue GUI, a program focused on categories. The framework of archive searching. MUDA-Blue makes browsing for a classified repository where several categories can belong to an application platform.
Altarawy et al., 2018 [6]	Automatic software categorization and related software searching are two main scenarios.	In their implementation, three well-annotated datasets were used. One was newly created, and the other two were from recent work. The 103application as a dataset is reused, and a freshly generated 5,220 applications were unlabeled data set.
Zhang, Jian, et al., 2019 [7]	This paper proposes a modern method that allows software projects to be classified without any source. Using code as attributes with a minimal number of API calls and even running a comprehensive scientific evaluation of alternatives to automated categorization.	The experimental data were gathered on two Java repositories. One was open-source, and the other was a closed-source repository containing 3,286 projects with 745 applications that did not have the source code.
Chen, Lingchao, et al., 2020 [8]	Introduce a different approach in this article for classifying software, called LACT, through open-source repository systems.	First, LACT has been tested to MUDABlue, for classifying 41 software systems in C into divisions of the problem area. LACT was extended in the second analysis to 43 information schemes carved in several languages for programs, such as C/C++, Java, C#, Perl, and PHP.
Komamizu, Takahiroet al, 2017 [9]	The introduction of a new recommendation is presented in this article. System for GitHub, servers which can be helpful in many ways, are looking for suitable options to create projects.	In this article, datasets of software mining repositories were used.

Izadi, Maliheh, et al., 2021 [10]	Automatic software for software categorization to help search comparable software programs in the software archive.	Build-up upon results of prior work in relationship determination between modules of a 41 x 220,488 software framework as an input in a matrix.
Barrios, Sonia, et al., 2019 [11]	This work, practical the goal of this thesis was to look at the possibilities of deep neural networks in software automatic categorization.	Low-level code tokens and high-level shaping concepts differentiate critical features such as calls to the API and identifiers to help categorize software.
LeClair, Eberhart, & McMillan, 2018 [12]	This article has indicated numerous modifications to the off-the-shelf neural-based text categorization in the software field.	Two datasets were used for evaluation purposes: c and c++ libraries' datasets, and the Debian end-user datasets employed expert annotating programmers.
Nafi et al., 2020 [4]	Software categorization applications for software dictionaries are a research field.	Experiments effectively detect related software in cross-language applications that outperform all current methods (mean, the average accuracy rate is 0.65, the confidence rate is 3.6, and effective queries are 75% highly rated).
Zhang et al., 2017 [13]	Propose a method in this paper to detect similar repositories found on GitHub effectively.	The experiment setup refers to RepoPal on a dataset of 1,000 projects from GitHub and compares it to CLAN's state-of-the-art approach.
Gu, Xiaodong al et, 2016 [14]	The idea of automatic detection is closely related to CLANs (Applications), allowing certain Java program users to find associated applications.	CLAN was generated by the dataset and performed and experimented with 33 respondents to assess and equate the CLAN with the CLAN MUDABlue, the nearest competitive solution, and a scheme combining CLAN of MUDABlue.
Samtani, Sagar, et al., 2017 [15]	Illustrate automatic machine learning approaches for source code categorization in eleven	For topical classifying, concentrate on Ibiblio and the C and C++ programs. Archives from Source forge.
	implementation subjects and ten programming languages.	
Catal et al. [16]	This review uses a classical method ensemble to overcome the mechanical software categorization issue where the source code is no more presented.	Make use of three collections of data and 745 closed-source Java applications.
Alreshedy, Dharmaretnamet	This study explains how to categorize source code using a	It can distinguish between code samples from related programming

al. 2018	classifier that can distinguish between the programming languages of code snippets published in 21 distinct programming languages.	languages, such as C, C++, and C#, and categories of programming language variants, such as C# 3.0, C# 4.0, and C# 5.0.
Reyes et al. [17]	The topic of classification of the source code is necessarily based on the text appearance sequence (syntax).	With minimal transformations, three datasets were included, both with their directory hierarchies.

GitHub is a browser-based code organization and collaboration tool that enables programmers to collaborate and succeed on various software development projects. GitHub is free and open source, with multiple features such as B. Contact management, virus monitoring, feature requests, job management, and wiki pages. These capabilities make it easier for developers to communicate and use version control systems like Git and Subversion (SVN) to control code changes. Git is the most widely used version control system for projects hosted on GitHub. Many software developers utilize GitHub, the most popular code management and collaboration service in the software industry. GitHub is used as an educational platform for software engineering classes in science. [18]. Because of the popular GitHub, the trainer uses GitHub in a corporate company to share code samples and publish programming assignments. This survey describes how the academy uses GitHub and uses GitHub in programming classes. Developers can construct many repositories and branches with Git, a distributed version control system. Subversion is a centralized version control system like Git. The primary difference is that it's a centralized version control system with only one central repository capable of keeping track of all code changes.

Developers who use GitHub can select these two version control systems based on their project's requirements for open-source software Wesseletal [19]. We have analyzed 351 famous GitHub repositories and recognized using bots in 93 (26%). They classified those bots, interviewed task maintainers, and accrued a few metrics to evaluate the reputation of the tasks earlier than and after the bots had been introduced. Fourteen bots were recognized for code or pull request reviews; however, no bots did the automated refactoring. In addition, they were using bots no longer, resulting in a statistically vast development inside the accrued metrics. The maximum pronounced troubles in bots are insufficient choice guide mechanisms and actively considering using refactoring instead of complete feedback. Some clinical courses additionally describe unique bots in best and productiveness development.

M Wyrick et al. [20] Presents a bot's imagination and prescient to manipulate the refactoring of wiki articles. The bot detects the best degradation, initiates a network balloting method to deal with the recognized issue, and implements the answer of preference for the majority, except for the specific balloting method, which is very like the supply code intended. Balachandran introduces VMware's Review [21] and its Fixit extension. The bot uses Check style, PMD, and Find Bugs to investigate Java repository pull requests. It integrates with VMware's code evaluation tool (evaluate board) to indicate refactoring of modifications and advise human reviewers based on the affected code segments. Extensions enhance this workflow with the aid of mechanically doing minor refactoring. A large quantity of inner information was used to assess the Review Bot, and the answer was defined in detail. However, the technique appears to be VMware-unique, and the code is not always open supply. In [22], proposed a unified model for face recognition which used the local binary pattern for face feature to texture conversion.

3. Mining GitHub Repositories

Mining GitHub repositories has been a long-standing topic in software engineering and social computing communities. Analytical research looks at how user behavior (collaboration, tracking, observation, and so on) influences development methods. README files and repository information are examined in algorithm research to conduct data mining tasks, including similarity search and clustering. Previous research, to our knowledge, has not considered automatic / lightly monitored subject classification in GitHub projects. Several analyses require heavily human work to annotate each repository. We hope the GitClass framework will pique your interest in automatically tagging repositories, as shown in Figure 1.

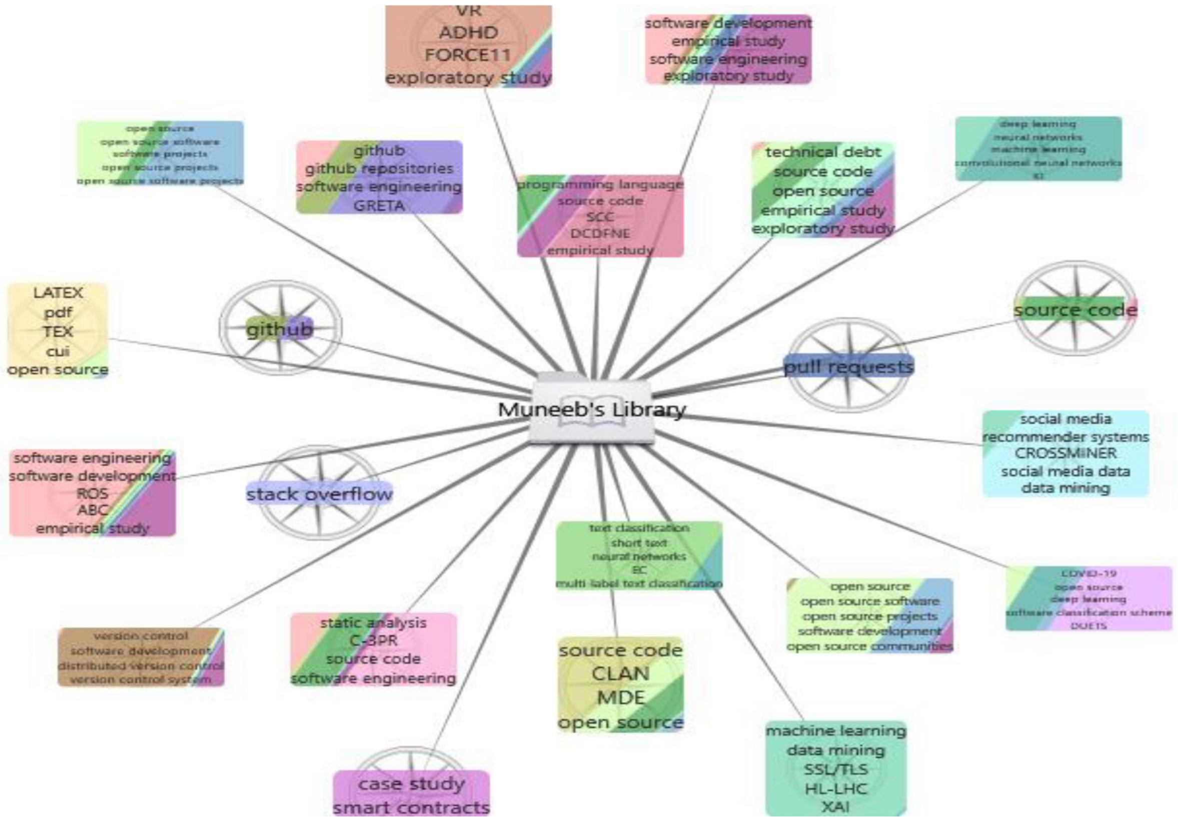

Figure 1: Source code classification-related work themes

4. HIN Mining

Much research has focused on using meta paths to perform similarity searches [18, 19] and node embeddings on HINs. Some studies have applied HIN node embedding to downstream classification tasks from an application perspective. Malware detection [20] and medical diagnosis [21]. In contrast to the fully monitored settings of [20, 21], the archive classification task relies on a small set of prompts. In addition, most of the information used in [20, 21] is structured. In contrast, the GitHub README signal is buried in heavy text noise. GitClass overcomes these unique challenges by introducing new technologies.

5. Using GitHub in a Variety of Areas

GitHub was primarily cast-off by instructors and tutors in computer science and interrelated disciplines. Programs are stored in most repositories. Like social sciences and law, teachers in other fields can also pass on GitHub. The cooperation feature is helpful for student development projects, and you can practice different types of tracking to make project documents for separate student or group projects. The GitHub repository is included US Federal laws. The repository holder used GitHub to save the document and made the federal code changes publicly available to contributors for viewing by all followers of the repository. Federal code changes can also be seen through version tracking. GitHub will become well known in many areas, so it's not just used in computer science. Davis has published a new study on librarians' use of GitHub and librarian education.

6. GitHub Classification Issues

The GitHub classification problem based on the InformatiCup 2017 task is a classification challenge. The goal is to appropriately identify the GitHub-hosted repository into one of the content categories below:
DEV: Projects involving software development, for example. HW: Homework and exercise solutions EDU: Educational projects, for example. DOCS: Documents that aren't intended to be educational, for example. WEB stands for the (individual) website. Data is a collection of information. Other: Repositories that do not fall into one of the categories above. Two aspects make the GitHub classification problem a daunting task.

- GitHub has various repositories. You can find a repository with thousands of contributors (such as the Linux kernel) from a repository with a project run by one person.
- Repository classifications can be ambiguous, and many projects can fall into several categories. This white paper makes some interesting observations about GitHub issues.

7. Methodological Framework

This section discusses the GitHub dataset, and we also discuss implementation details. To embark, we prepared our dataset; we downloaded code repositories from GitHub, including 11,380 folders. These folders have three types of programming languages (java, python, and CPP). We save these codes into separate folders according to their file type (.java, .py, .cpp) and label each folder. The folder name is depended on the purpose of writing the code, as shown in Figure 2

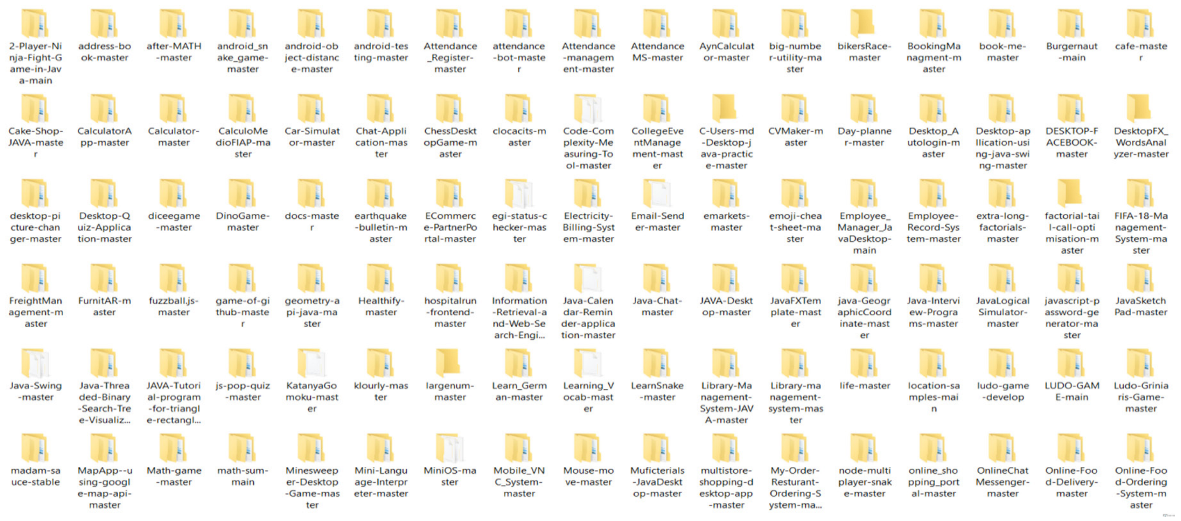
2-Player-Ninja-Fight-Game-in-Java-main
address-book-master
after-MATH-master
android_snake_game-master
android-object-distance-master
android-testing-master
Attendance_Register-master
attendance-bot-master
Attendance-management-master
AttendanceMS-master
AynCalculator-master
big-number-utility-master
bikersRace-master
BookingManagment-master
book-me-master
Burgernaut-main
cafe-master
Cake-Shop-JAVA-master
CalculatorApp-master
Calculator-master
CalculoMedioFIAP-master
Car-Simulator-master
Chat-Application-master
ChessDesktopGame-master
clocacits-master
Code-Complexity-Measuring-Tool-master
CollegeEventManagement-master
C-Users-md-Desktop-java-practice-master
CVMaker-master
Day-planner-master
Desktop_Autologin-master
Desktop-application-using-java-swing-master
DESKTOP-FACEBOOK-master
DesktopFX_WordsAnalyzer-master
desktop-picture-changer-master
Desktop-Quiz-Application-master
diceegame-master
DinoGame-master
docs-master
earthquake-bulletin-master
ECommerce-PartnerPortal-master
egi-status-checker-master
Electricity-Billing-System-master
Email-Sender-master
emarkets-master
emoji-cheat-sheet-master
Employee_Manager_JavaDesktop-main
Employee-Record-System-master
extra-long-factorials-master
factorial-tail-call-optimisation-master
FIFA-18-Management-System-master
FreightManagement-master
FurnitAR-master
fuzzball.js-master
game-of-github-master
geometry-api-java-master
Healthify-master
hospitalrun-frontend-master
Information-Retrieval-and-Web-Search-Engi...
Java-Calendar-Reminder-application-master
Java-Chat-master
JAVA-Desktop-master
JavaFXTemplate-master
java-GeographicCoordinate-master
Java-Interview-Programs-master
JavaLogicalSimulator-master
javascript-password-generator-master
JavaSketchPad-master
Java-Swing-master
Java-Threaded-Binary-Search-Tree-Visualiz...
JAVA-Tutorial-program-for-triangle-rectangl...
js-pop-quiz-master
KatanyaGomoku-master
klourly-master
largenum-master
Learn_German-master
Learning_Vocab-master
LearnSnake-master
Library-Management-System-JAVA-master
Library-management-system-master
life-master
location-samples-main
ludo-game-develop
LUDO-GAME-main
Ludo-Grinris-Game-master
madam-sauce-stable
MapApp--using-google-map-api-master
Math-game-master
math-sum-main
Minesweeper-Desktop-Game-master
Mini-Language-Interpreter-master
MiniOS-master
Mobile_VNC_System-master
Mouse-move-master
Muficterials-JavaDesktop-master
multistore-shopping-desktop-app-master
My-Order-Resturant-Ordering-System-ma...
node-multiplayer-snake-master
online_shopping_portal-master
OnlineChatMessenger-master
Online-Food-Delivery-master
Online-Food-Ordering-System-master

Figure 2: GitHub repositories' dataset in the form of source code folders

We downloaded 450 source code repositories or data clones from GitHub in three languages (java, python, and C++). Each of these three languages has 150 source code repositories. After that, we perform preprocessing on the data. Data preparation is a crucial stage in developing a deep learning model. When working on a deep learning project, you do not always come across tidy, well-formatted data. As a result, while working with data, you must clean it up and prepare it. Because machines can only interpret numbers and not text, this is the situation for deep learning and machine learning algorithms. Hot coding is crucial in transforming categorical data variables for machines and deep learning algorithms. It enhances model prediction and classification accuracy. One hot encoding is a typical method of preprocessing a machine learning model's category information. This form of coding creates a new binary feature for each potential category and assigns a value of 1 to each sample feature that matches the original category. One hot encoding is an integral aspect of the feature engineering process in learning technology training. Colour, for example, was a variable, and the labels were "red," "green," and "blue." Each of these labels can be represented as a binary vector consisting of three red elements: [1, 0, 0], green elements: [0, 1, 0], and blue elements: [0, 0, 1]. During processing, category data must be translated into numeric format. The dataset is now ready to be fed into the deep learning model. To classify GitHub repositories, use the CNN model. In this model, you can enter different epoch values and get varied accuracy results.

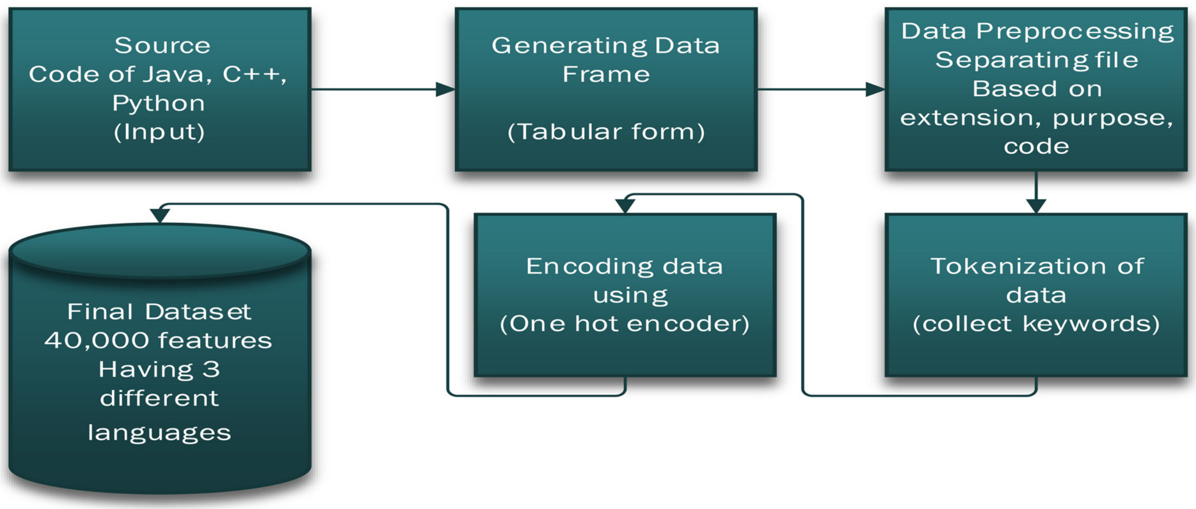

Figure 3: Methodology for GitHub Repositories' Classification

8. Results and Discussion

In this section of the research paper, we discuss the results of our research. We run our model on different epoch values to observe the model behavior on other inputs—our model trains from 150 epoch values to 250 epoch values. The overall framework of our approach is shown in Figure 3. It takes a GitHub source code as input and outputs a set of classifications and the purpose of writing code. By dividing the dataset into 70 and 30 ratios for mutual validation using neural networks, we achieved 97% overall classification accuracy. The model was trained in 11,280 source code folders and tested with 5,814 code scraps to classify the three languages, i.e., CPP, Java, and Python. The overall accuracy result is 97%, as shown in Figure 4.

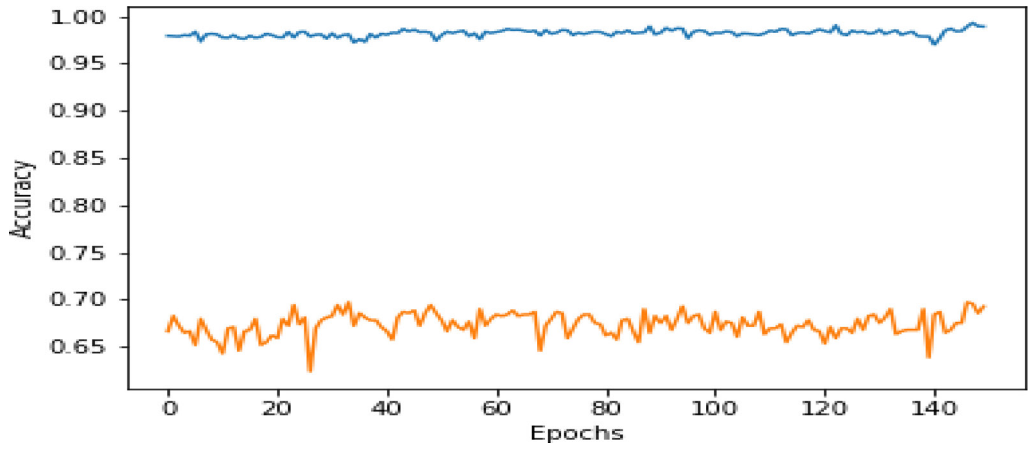

Figure 4: Repositories Classification Training and Validation Accuracies

This paper presented Classify GitHub repositories and algorithms for handling GitHub classification problems with high fidelity (97%). It is achieved through the usage of CNN. In addition, we provide a dataset of 450 GitHub repositories. It contains 11,280 folders and 5,814 files that can be used for further research.

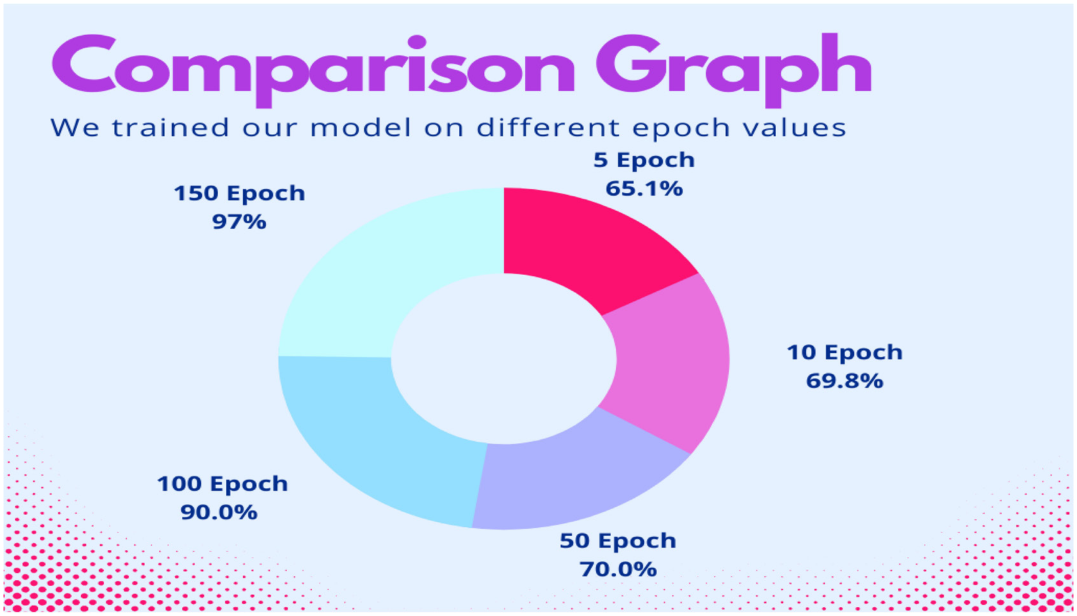

Figure 5: Classification training and validation accuracies

Figure 5 shows the accuracy comparison at training time at different epoch values. This figure shows 65.1 %, 69.8%, 70%, 90% and 97% accuracy at 5 epoch, 10 epochs, 50 epochs, 100epochs and 150 epochs respectively as shown in Figure 5.

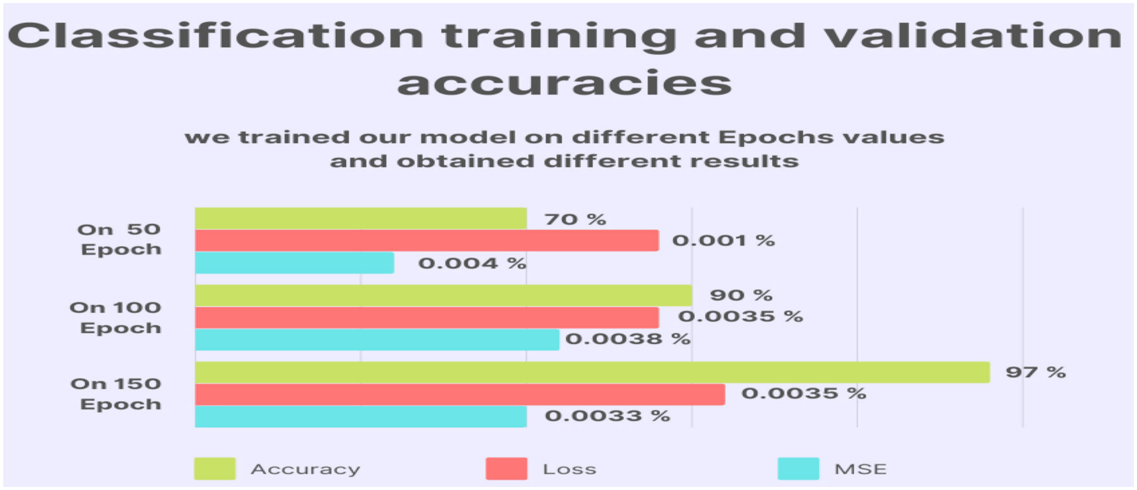

Figure 6: Accuracy, loss, and MSE for the training and validation of the model using CNN

This bar chart compares the Accuracy, Loss, and MSE graph at different epochs inputs. The green line shows the accuracy percentage. The Red line shows loss values, and the blue line indicates the mean squared error, as shown in Figure 6. In the bar chart, we readily observe that when the epoch value increases, accuracy increases, and loss and mean square error decrease. When we train our mode at a high epoch value, it increases accuracy. On 150 epochs, the training model takes 3 to 5 hours, and testing takes 2 hours.

Training 🟥

Testing 🟦

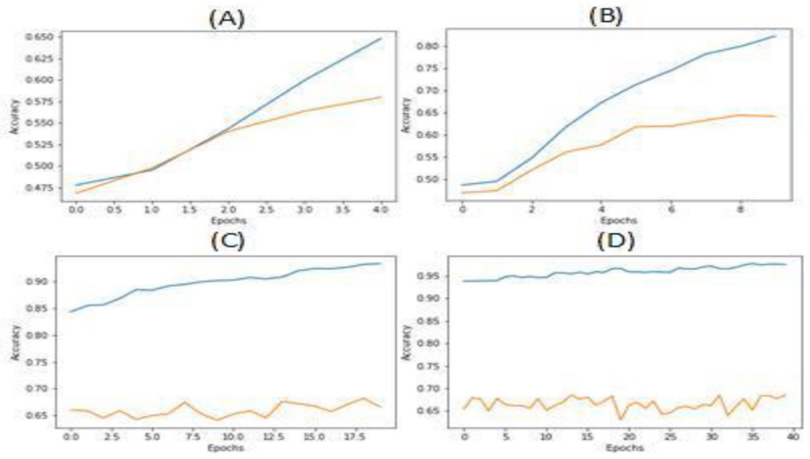

Figure 7: Classification and validation accuracies at 5, 10, 20, and 40 Epochs values Using CNN

Training 🟥

Testing 🟦

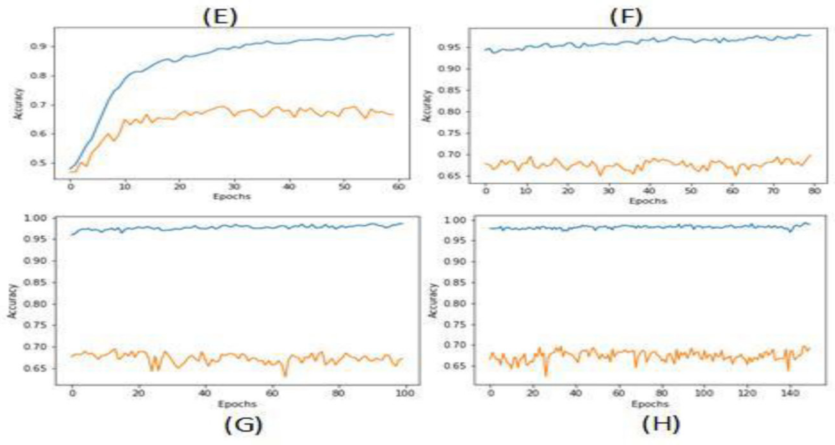

Figure 8. Classification accuracies and validation at 60,80,100 and 150 Epochs values Using CNN

The figures show different accuracies at different input values. Graph (A) shows accuracy at five epochs during training and testing. Same as graph (B), (C), (D), (E), (F), (G) and (H) shows accuracies on 10, 20, 40, 60, 80, 100, and 150 epochs values respectively. We trained our model using this approach and achieved high accuracy, as shown in Figures 7 and 8.

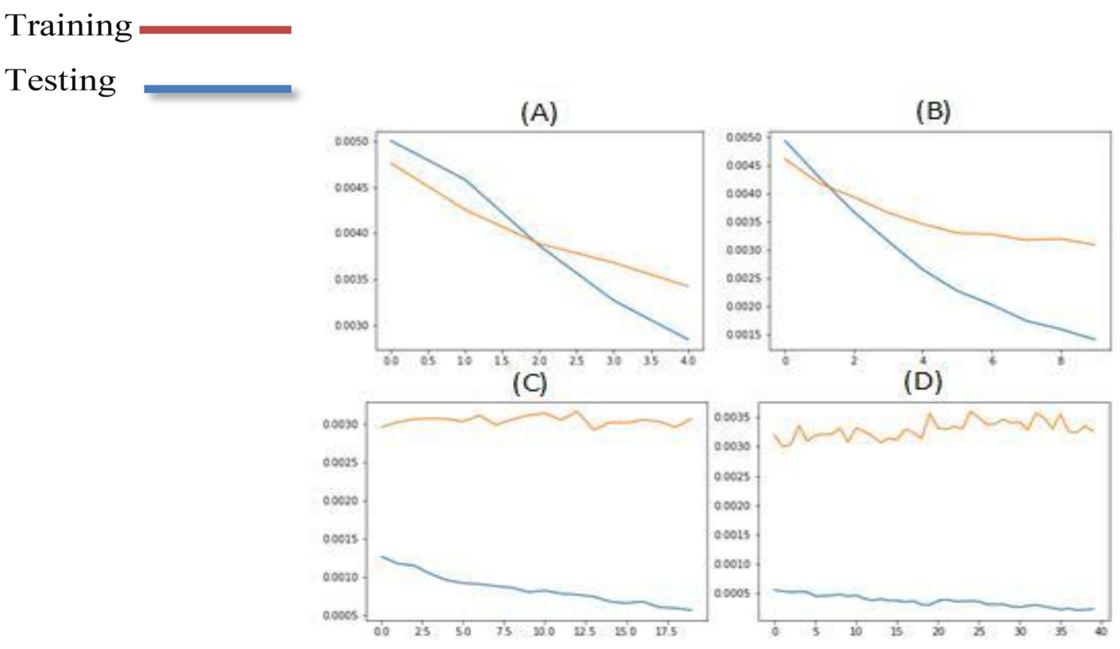

Figure 9: Loss values at 5, 10,20, and 40 epochs using CNN

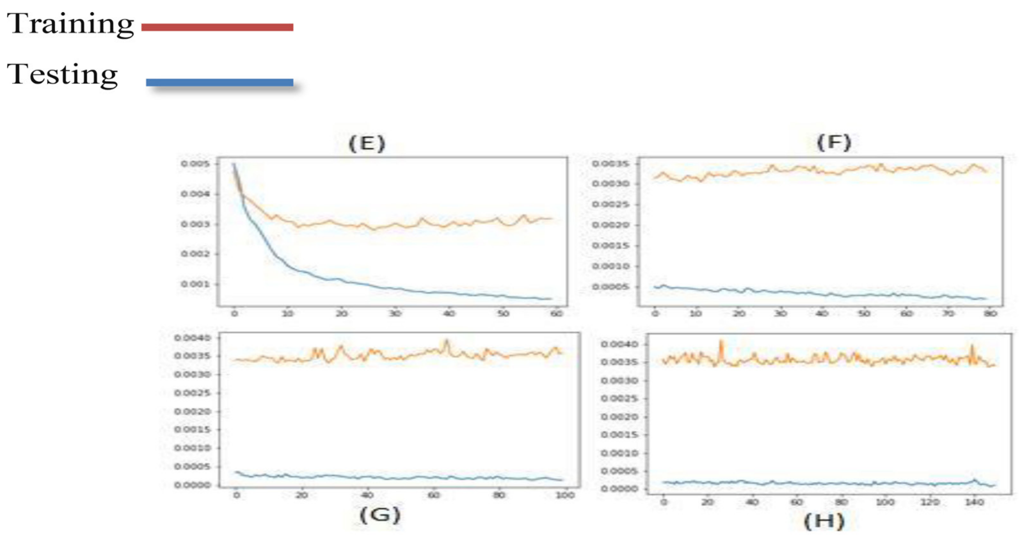

Figure 10: Loss values at 60,80,100 and 150 Epochs Using CNN

The figures show a loss at different input values. Graph (A) shows loss at five epochs during training and testing. Same as graph (B), (C), (D), (E), (F), (G) and (H) shows loss on 10, 20, 40, 60, 80, 100, and 150 epochs values respectively. We trained our model with less loss using this approach, as shown in figure 9 and figure 10.

9. Conclusion

The outcomes mentioned above appear to be quite encouraging. This method shows that the source code type can be completed using the earlier criteria. The software's great precision and quick response time make it ideal for the most realistic functions. The software had a 97 percent accuracy when identifying three programming languages. The computerized application does not rely on expanding the programming language record or other report facts. It does not necessitate knowledge of how the programming language identifies it, and identification takes approximately 100 microseconds.

Despite the classifier's outstanding overall performance, there is room for improvement. A character-stage model, for example, could be tried, which learns directly from the characters without needing a word embedding layer. By obtaining versioned information for each programming language, it is also possible to select the exact version of the code sample. In the future, we want to address such concerns.

10. Limitation

- I have trained the model using three programming languages such that Java, C++, and python. Currently, in these languages, our model predicts more accurately.
- We have collected the source code from the GitHub portal. But there are a lot of software repository portals available on the internet.

11. Future Work

In the future, we will prepare a model that will predict the source code language and category of code based on programming language syntax. So, we will make our model more efficient in the future. And in the future, we will convert the code file into byte files and images. After that, we classify the programming languages on image base classification.

Reference

[1] Altarawy, D., et al., *Lascad: Language-agnostic software categorization and similar application detection.* Journal of Systems and Software, 2018. **142**: p. 21-34.

[2] Jelodar, H., et al., *Latent Dirichlet allocation (LDA) and topic modeling: models, applications, a survey.* Multimedia Tools and Applications, 2019. **78**(11): p. 15169-15211.

[3] Auch, M., et al., *Similarity-based analyses on software applications: A systematic literature review.* Journal of Systems and Software, 2020. **168**: p. 110669.

[4] Nafi, K.W., et al., *A universal cross-language software similarity detector for open source software categorization.* Journal of Systems and Software, 2020. **162**: p. 110491.

[5] Katz, D.S., et al. *Detecting execution anomalies as an oracle for autonomy software robustness*. in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020. IEEE.

[6] Bouziane, Y., M.K. Abdi, and S. Sadou, *Automatically Labelled Software Topic Model.* International Journal of Open Source Software and Processes (IJOSSP), 2020. **11**(1): p. 57-78.

[7] Zhang, J., et al. *A novel neural source code representation based on abstract syntax tree*. in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. 2019. IEEE.

[8] Chen, L., et al. *Taming behavioral backward incompatibilities via cross-project testing and analysis*. in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. 2020.

[9] Komamizu, T., et al. *Exploring Identical Users on GitHub and Stack Overflow*. in *SEKE*. 2017.

[10] Izadi, M., A. Heydarnoori, and G. Gousios, *Topic recommendation for software repositories using multi-label classification algorithms.* Empirical Software Engineering, 2021. **26**(5): p. 1-33.

[11] Barrios, S., et al., *Partial discharge classification using deep learning methods—Survey of recent progress.* Energies, 2019. **12**(13): p. 2485.

[12] LeClair, A., Z. Eberhart, and C. McMillan. *Adapting neural text classification for improved software categorization*. in *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 2018. IEEE.

[13] Zhang, Y., et al. *Detecting similar repositories on GitHub*. in *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. 2017. IEEE.

[14] Gu, X., et al. *Deep API learning*. in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. 2016.

[15] Samtani, S., et al., *Exploring emerging hacker assets and key hackers for proactive cyber threat intelligence.* Journal of Management Information Systems, 2017. **34**(4): p. 1023-1053.

[16] Catal, C., S. Tugul, and B. Akpinar, *Automatic software categorization using ensemble methods and bytecode analysis.* International Journal of Software Engineering and Knowledge Engineering, 2017. **27**(07): p. 1129-1144.

[17] Reyes, J., D. Ramírez, and J. Paciello. *Automatic classification of source code archives by programming language: A deep learning approach*. in *2016 international conference on computational science and computational intelligence (CSCI)*. 2016. IEEE.

[18] Yang, Z., et al. *Hierarchical attention networks for document classification*. in *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*. 2016.

[19] Li, C., et al. *Effective document labeling with very few seed words: A topic model approach*. in *Proceedings of the 25th ACM international on conference on information and knowledge management*. 2016.

[20] Meng, Y., et al. *Weakly-supervised neural text classification*. in *proceedings of the 27th ACM International Conference on information and knowledge management*. 2018.

[21] Angulo, M.A. and O. Aktunc. *Using GitHub as a teaching tool for programming courses*. in *2018 Gulf Southwest Section Conference*. 2019.

[22] Alim, Affan, et al. "The most discriminant subbands for face recognition: A novel information-theoretic framework." *International Journal of Wavelets, Multiresolution and Information Processing* 16.05 (2018): 1850040.