

A Deep Hybrid Learning Based Story Point Estimation for Agile Project



MCS

Author

Asma Saleem
00000327245

Supervisor

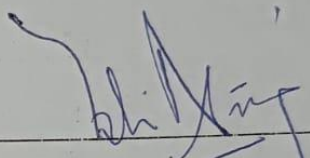
Dr. Fahim Arif

A thesis submitted to the faculty of Department of Computer Software Engineering, Military College of Signals, National University of Sciences and Technology (NUST), Rawalpindi, in partial fulfillment of the requirement for the degree of MS in Software Engineering.

September 2023

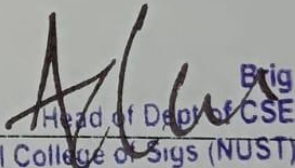
THESIS ACCEPTANCE CERTIFICATE

Certified that final copy of MS Thesis written by **Ms. Asma Saleem**, Registration No. **00000327245**, of **Military College of Signals** has been vetted by undersigned, found complete in all respects as per NUST Statutes/Regulations/MS Policy, is free of plagiarism, errors, and mistakes and is accepted as partial fulfillment for award of MS degree. It is further certified that necessary amendments as pointed out by GEC members and local evaluators of the scholar have also been incorporated in the said thesis.

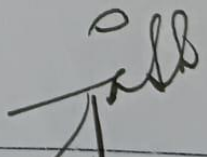
Signature:  _____

Name of Supervisor: Assoc. Prof Dr. Fahim Arif

Date: _____

Signature (HOD):  _____
Head of Dept of CSE
Mil College of Sigs (NUST)

Date: _____

Signature (Dean/Principal)  _____

Date: 15/11/23

Brig
Dean, MCS (NUST)
(Asif Masood, Phd)

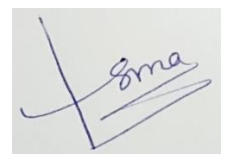
Declaration

I, Asma Saleem declare that this thesis “A Deep Hybrid Learning Based Story Point Estimation for Agile Project” and the work presented in it are my own and have been generated by me as a result of my original research.

I confirm that:

- 1) This work was done wholly or mainly while in candidature for a Master of Science degree at NUST.
- 2) Where any part of this thesis has previously been submitted for a degree or any other qualification at NUST or any other institution, this has been clearly stated.
- 3) Where I have consulted the published work of others, this is always clearly attributed.
- 4) Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my work.
- 5) I have acknowledged all main sources of help.
- 6) Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Name of Student: Asma Saleem

A handwritten signature in blue ink, appearing to read 'Asma', is written over a horizontal line. The signature is stylized and includes a large, sweeping flourish on the left side.

Signature of Student

Dedication

“In the name of Allah, the most Beneficent, the most Merciful”

Glory be to Allah (S.W.A), the Creator, the Sustainer of the Universe. Who only has the power to honor whom He please, and to abase whom He please. Verily no one can do anything without His will. From the day, I came to NUST till the day of my departure, He was the only one Who blessed me and opened ways for me and showed me the path of success. There is nothing which can payback for His bounties throughout my research period to complete it successfully. I dedicate this thesis to myself, my parents, brothers, friends and teachers who supported me in achieving my goals.

Abstract

The Agile technique is a widely recognized and used strategy within the field of software engineering. Numerous organizations that prioritize adaptability, customer-centric methodologies, and team appreciation use agile practices. The use of agile approach in software development is undeniably associated with the capacity to adapt to changing environments and foster personal growth among employees. The Agile methodology allows for the active participation of all team members, fostering creativity and facilitating the management of large-scale projects within expansive teams. The proficiency in inter-team and cross-team communication for the purpose of deliberating project resolutions and effectively implementing effective solutions to real-world challenges.

In the realm of agile methodology, a story point refers to a challenging job that has the potential for adverse consequences on team collaboration and overall project performance, often resulting in budgetary overruns. This relatively small but crucial area need meticulous deliberation throughout the estimating process, when a project manager with substantial expertise is tasked with making pivotal decisions for the project. If the manager is deficient in certain expertise, it might have catastrophic consequences for the whole business. Story points refer to the user requirements that are carefully considered by a business analyst, with potential involvement from a software developer to clarify the expectations. Projects that have clear and well-defined timetables are more likely to be completed on time, resulting in a higher level of satisfaction in terms of team performance.

Artificial Intelligence (AI) has emerged as a rapidly expanding field in the current decade, with several activities being performed utilizing this advanced technology. Deep Learning, Federated Learning, ML, Reinforcement learning, and several other techniques are integral components of artificial intelligence (AI). The practice of artificial intelligence (AI) in addressing practical challenges partakes provided researchers with an opportunity to apply their repertoire of knowledge and skills towards resolving environmental issues. This is a significant demonstration of software engineering, whereby automation is used to enhance human productivity and save their precious time. This phenomenon is seen in the context of narrative point estimate with artificial intelligence (AI), whereby AI is used to provide precise estimations in order to enhance project planning.

This study introduces a unique Deep Hybrid Learning Model, namely GPT2-CNN, which is used as an Agile Project Estimation approach through story point. The Deep Hybrid Learning Model, GPT2-CNN, utilizes a GPT2 etymological model that has prior experience, combined with deep neural-based architecture known as CNN. This integration enables our models to effectively comprehend the

interconnections among words, taking into account the contextual information surrounding a specific word and its placement within the sequence. The findings of our study indicate that our proposed Hybrid Learning Model, namely GPT2-CNN, has a median Mean Absolute Error (MAE) of 1.96. This performance surpasses that of the current baseline technique used for estimating within-project estimates. The ablation research also demonstrates the efficacy of our deep hybrid learning architecture in augmenting the estimation process of agile user stories, henceforth rank the momentous progressions of AI in Agile story point estimation. Our findings are substantiated by the implementation of five distinct hybrid models, namely GPT2-CNN (with default settings), GPT2-LSTM, GPT2-LSTM-CNN (a fusion of three deep learning models), and RoBERTa-RoBERTa. The whole of the experimental configuration confirms the strong performance of GPT2-CNN(AdamW). The task of estimating story points is regarded as a complex endeavor, as noted by Agile practitioners. The use of big data and powerful computers is expected to significantly improve the quality of the solution offered.

Acknowledgments

All praises to Allah for the strengths and His blessing in completing this thesis. I would like to convey my gratitude to my supervisor, Dr. Fahim Arif, PhD, for his supervision and constant support. His priceless help of constructive comments and suggestions throughout the experimental and thesis work are major contributions to the success of this research. Also, I would thank my committee members; Prof. Dr. Hammad Afzal, and Brig. Adnan Ahmed Khan, PhD for their support and knowledge regarding this topic. Lastly, I am highly thankful to my parents, my siblings and my friends for their constant support. I would like to thank them for their patience, cooperation and motivation in times of stress and hard work.

Table of Contents

Introduction	1
1.1 Overview	1
1.2 Problem Statement	4
1.3 Objectives of Research	5
1.4 Research Contribution	5
1.5 Thesis Organization	6
1.6 Summary	6
Related Work	7
2.1 Overview	7
2.2 Conventional Approaches to Agile Story Point Estimation	7
2.3 Machine Learning Approaches to Agile Story Point Estimation	10
2.4 Deep Learning Approaches to Agile Story Point Estimation	16
2.5 Limitations in Estimation of Story Point Using DL Approaches	20
2.6 Summary	21
Methodology and Framework	22
3.1 Overview	22
3.2 Overview of Transformers (GPT2) Generative Pretrained Transformers	22
3.3 Proposed GPT2-CNN Hybrid Learning Model for Story Point Estimation in Agile Projects	24
3.3.1 Data Pre-processing and Acquisition	25
3.3.2 Sub-word Tokenization and Byte Pair Encoding (BPE)	27
3.3.3 Convolutional Neural Networks	29
3.4 Ablation Experiments	31
3.4.1 GPT2-CNN Model	31
3.4.2 GPT2- LSTM Model	31
3.4.3 GPT2-LSTM-CNN Model	33
3.4.4 RoBERTa(Tokenizer)- RoBERTa(Classifier)	34
3.5 Experimental Analysis of Proposed Framework	35
3.5.1 Implement Details	35
3.5.2 Hyper parameters Details	35
3.5.3 Loss function	36
3.6 Ablation Study – Experimental Analysis	37
3.7 Summary	37
Results and Discussion	39

4.1 Overview	39
4.2 Regression Evaluation Metrics	40
4.2.1 Mean Absolute Error	40
4.2.2 Median Absolute Error Evaluation	41
4.3 Experimental Analysis of Proposed Hybrid Models	41
4.3.1 Graphs	43
4.3.2 Heat Map	52
4.4 Evaluation on GPT2-CNN	53
4.5 Evaluation on GPT2-LSTM	54
4.6 Evaluation of GTP-2 – LSTM – CNN	55
4.7 Evaluation of Roberta-Roberta-Adam	57
4.8 Comparative Analysis of Various Methods	58
4.9 Summary	59
<i>Conclusion and Future Work</i>	60
5.1 Conclusion	60
5.2 Limitations	60
5.3 Future Work	60
6. References	62

List of Figures

- Figure 1.1 A visualized sample of Jira story from Appcelerator Studio4
- Figure 3.1 GPT2 sub-word tokenization is shown where applying Byte Pair Encoding (BPE) is used for the task of making tokens29
- Figure 3.2 CNN Architecture Displaying all layers and transition of data between layers is shown30
- Figure 3.3 A detailed Architecture Diagram, explaining the working of GPT2-CNN Hybrid Learning Model for estimation of agile user stories in projects38
- Figure 4.1 Graph of Trained and Validated Appcelerator Studio44
- Figure 4.2 Training and Validation of Aptana Studio44
- Figure 4.3 Bamboo Studio graph explaining the training and validation of dataset45
- Figure 4.4 Clover data file training and validation MAE46
- Figure 4.5 Data Management File Training and Validation MAE46
- Figure 4.6 Dura Cloud Training and Validation MAE47
- Figure 4.7 Jira Software Training and Validation MAE47
- Figure 4.8 Mesos Training and Validation MAE48
- Figure 4.9 Moodle Training and Validation MAE48
- Figure 4.10 Mule Training and Validation MAE49
- Figure 4.11 Mule Studio Training and Validation MAE49
- Figure 4.12 Springxd Training and validation MAE50
- Figure 4.13 Talend Data Quality Training and Validation MAE50
- Figure 4.14 Talendesb Training and Validation MAE51
- Figure 4.15 Titanium Training and Validation MAE51
- Figure 4.16 Usergrid Training and Validation MAE52
- Figure 4.17 Heatmap of Training and Validation MAE of all project files53

List of Tables

Table 3.1 Detailed description of dataset utilized for GPT2 CNN Hybrid Learning Model.....	26
Table 4.1 Results of GPT2-CNN (Adam W optimizer) Hybrid Learning Model	42
Table 4.2 GPT2 as tokenizer -CNN as classifier where CNN is not used with any optimizer.....	54
Table 4.3 GPT2- LSTM where gpt2 as tokenizer and LSTM as classifier.....	55
Table 4.4 GPT2-LSTM-CNN a tri model variation evaluation	56
Table 4.5 Roberta as tokenizer and Roberta as classifier with Adam optimizer	57
Table 4.6 Comparison between all models and state-of-the-art model	58

List of Abbreviations

<i>CNN</i>	<i>Convolutional Neural Network.</i>
<i>LSTM</i>	<i>Long Short Term Memory.</i>
<i>NLP</i>	<i>Natural language processing.</i>
<i>GPT</i>	<i>Generative Pre-Trained Transformer.</i>
<i>AI</i>	<i>Artificial Intelligence.</i>
<i>ML</i>	<i>Machine Learning.</i>
<i>DL</i>	<i>Deep Learning.</i>
<i>MAE</i>	<i>Mean Absolute Error.</i>
<i>SPE</i>	<i>Story Point Estimation.</i>

Chapter 1

Introduction

1.1 Overview

Agile methodologies entail a collection of pivotal activities that collectively define the operational landscape of dynamic project development. These activities, calibrated for collaborative engagement, communication, and continuous enhancement, align with agile principles. "Sprint Planning" involves selecting tasks from the prioritized product backlog for an upcoming sprint, reflecting agility's responsiveness to evolving requirements. The "Daily Stand-up" meeting is a brief, daily forum for team members to share progress, plans, and obstacles, promoting communication and synergy. "Sprint Review" entails showcasing completed work to stakeholders, eliciting feedback for iterative refinement. The "Sprint Retrospective" fosters introspection, identifying strengths, areas for improvement, and actionable steps. "Product Backlog Refinement" involves meticulous grooming of backlog items to ensure clarity and readiness for integration. Incremental development partitions projects into sprint-sized segments, delivering value iteratively and enabling course corrections. "Pair Programming" fosters knowledge exchange as two developers collaborate at one workstation, elevating code quality. "Test-Driven Development (TDD)" mandates writing tests before code to ensure functionality aligns with expectations. "Continuous Integration (CI)" automates code merging and testing, enhancing codebase stability. "Continuous Delivery/Deployment (CD)" automates code delivery, enabling rapid and dependable releases. "Backlog Prioritization" orders items based on feedback, market shifts, and strategic goals, prioritizing value. "Cross-Functional Collaboration" blends roles, fostering comprehensive perspectives for holistic development. These activities collectively embody agile principles, promoting adaptability, user-centricity, and iterative progress.

Less research is carried on estimating all methods of agile project, particularly the struggle needed to finish user stories or issues, despite the fact that there has been a lot of work done on software engineering for approximation effort for conventional software projects. The story point is the most often used unit of measurement for estimating the amount of struggle required to complete a user experience or resolve an issue on schedule and within budget. As a starting point for developing project planning, iteration or release timelines, planning, and estimate, various stakeholders may use effort estimates. To estimate a proportionate number in terms of man hours or man months, agile makes use of story-based estimations. The goal of estimating story points from agile user stories is related to how much work, how much time, how difficult it is, the risks involved, and how unpredictable it is. Before

a project is started, the scope is established when a team is given a project. User stories are used in Agile teams to break down work into manageable, achievable chunks. Typically, story point estimating occurs during the product backlog refinement session. The proposed research will carry out the utilization of deep learning models in combination with machine learning models.

Estimation of effort is considered challenging task in agile practices. The estimated amount of work required to entirely accomplish artefact is expressed in story points. Story points are units of measurement. According to team consensus, the team often estimates the story points by utilizing a variety of approaches, including Planning Poker, analogies, and expert judgment, while taking into consideration the labor required, the complexity of the situation, the risk involved, and the degree of uncertainty. However, it should be understood that a subjective analysis based on domain experts may introduce bias [5].

Many projects failed due to over estimation and under estimation of project schedule. Story point estimation comes in the initial stages of software development. Analyst after gathering requirement delivers the user stories to team, where product owner and project managers are responsible for making project plans.

As a result, many machine learning (ML) and artificial intelligence (AI) algorithms have been used to evaluate story points. For example, Porru[4] employed a combination of Bag-of-Words (BoW) features and other approaches to learning, such as support vector algorithms. The process was time-consuming due to the manual creation of Bag-of-Words (BoW) pieces. The Deep-SE venture was initiated by providing training to a language model in order to generate vector interpretations. Deep-SE incorporates the utilization of LSTM (Long Short-Term Memory) plus Recurrent Highway Network (RHWN) to automatically make estimations of story points also acquire the scattered depiction of phrases. Nevertheless, the Deep-SE technique is susceptible to certain restrictions. Throughout the setting of a single task, Deep-SE initially constructs an already-trained language model by leveraging its exclusive story point databases.

Project-explicit prior training requires a significant amount of time, characteristically ranging from 2 to 7 hours for Deep-SE [23] to generate a pretrained model for each project. This approach restricts the language model's comprehension to familiar vocabularies specific to the trained project and lacks the ability to be applied to different projects, as evidenced by Deep-SE's unreliable estimations when used across projects. Additionally, the embedding vectors produced by the pre-trained models are acquired through the utilization of Long Short-Term Memory (LSTM) by Deep-SE. The chronic order

of LSTM bounds processing words in one direction. Hence, the potency of these networks is constrained in terms of thoroughly capturing the complex lexical variations and interrelationships among phrases within the context of a topic and its corresponding story points. This limitation results from their incapacity to determine wider word dependencies that extend across the present words through the preceding words during a given sequence. A sample story point from apccelerator studio is show in **figure 1.1**

The purpose of story point estimation will save the development team time, all members are expected to put their input in estimating story point, now with help of machine learning based techniques can assist in this important task. In general, the use of algorithms based on machine learning for the assessment of story points is a subject of ongoing scholarly investigation, with several distinct methodologies recently examined until now. One advantage of machine learning algorithms is their ability to enhance accuracy by effectively analyzing extensive datasets and detecting patterns that may elude human perception, hence resulting in more precise estimations.

- 1) **Consistency:** Machine learning models can provide consistent estimates, reducing variability in estimates between team members.
- 2) **Speed:** Machine learning algorithms can process data and generate estimates much faster than humans, which can be especially useful for large projects with many stories to estimate.
- 3) **Adaptability:** Machine learning models can be trained on new data and adjust their estimates accordingly, allowing them to adopt to changes in the project or team's estimation process.
- 4) **Reduced Bias:** Machine learning algorithms can reduce bias in estimates by removing human subjectively from the estimation process.

Area of Application of study:

This ablation study will be used to assist project managers, product owners and other sub fields of project management as mentioned below:

- 1) **Agile software development:** Machine learning can be used to help teams estimate the size and complexity of stories in agile development process.
- 2) **Project management:** Machine learning can be used to help project managers accurately estimate the duration and cost of projects, leading to more accurate budget and resource planning.
- 3) **Resources Allocation:** Machine learning can be used to help teams allocate resources more effectively by accurately estimating the efforts required for each task.

- 4) **Risk Management:** Machine learning can help teams identify and mitigate risks by analyzing data on past projects and identifying patterns that may indicate potential issues.
- 5) **Quality Assurance:** Machine learning can be used to help teams identify and prioritize defects or issues that need to be addressed, leading to improved product quality.

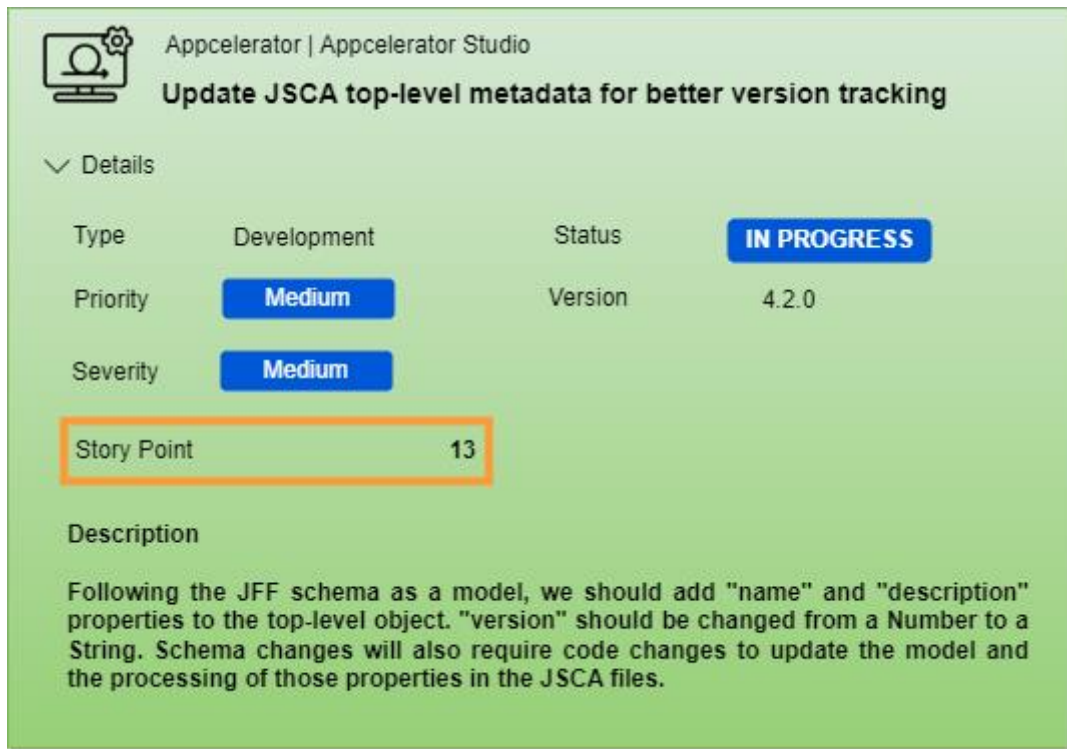


Figure 1.1 A visualized sample of Jira story from Appcelerator Studio

1.2 Problem Statement

Agile development is the most practiced in software house around globe. Beside it has many variations in the form Scrum, Kanban, etc. The gathering of user stories is always a tedious task. After selecting the right requirements for the project user stories are evaluated by the experienced team members. The practice of story-point estimate allows teams working on software to enhance their ability to define the scope of projects, organize requirements, distribute assets, and then evaluate progress, besides other benefits. Project managers who give story points estimations should have prior experience and knowledge. Most of the time when a single project manager is working on multiple projects needs some assistance in executing the managerial tasks. To ease this problem many researchers have carried out research in the field of story point estimation using user stories. Since user stories are stored in the product backlog, the product owner has to maintain that product backlog. The maintenance of product backlog items involves the estimation of story points and prioritizing user stories according to pre-planned schedules and available resources. The

predicted story points will help the product owner to make decisions beforehand so that development team time can be saved. An accurate estimation by the model is beneficial to the product owner. As our predicted story points allowed them to decide in a better way and not go through the extensive task of reading the user stories completely.

1.3 Objectives of Research

The present study centers on the following research aims:

- To develop a novel approach where deep learning models and conventional machine learning models are used for making story point estimation interpretable by product owners and project managers.
- A comparative analysis on dataset, provided by cutting-edge approaches.
- Evaluate the efficiency and efficacy of the suggested model.

1.4 Research Contribution

To the finest of our knowledge, the methodology and framework presented in this thesis have not previously been used in the process of estimating story point. The primary points of this thesis are:

- Dataset pre-processing, since the provided dataset by baseline approach require a cleansing of data to improve the model results.
- We present unique hybrid learning model to solve the desired problem. A transformer-based model and machine-learning-based model, which we denoted as Hybrid Learning Model.
- To the best of our knowledge, this study conducts a comprehensive evaluation of 23,313 Jira issues using a prevalent reference point approach, including Deep-SE, to analyze assessment situations inside a project. Ablation research was conducted to assess the individual contributions of five components (GPT2-CNNAdamW, GPT2-CNN, GPT2-LSTM, GPT2-LSTM-CNN, Roberta-Roberta) employed in our hybrid learning model.
- Furthermore, we propose the deep hybrid learning model is lightweight model as compared to Deep-Se.

1.5 Thesis Organization

The present thesis has been structured in the following manner:

- Chapter 2 provides a comprehensive summary concerning related work in the domain of Story Point Estimation (SPE). The section 2.1 thoroughly discusses the Traditional techniques used in SPE. Section 2.2 gives briefing about Machine Learning (ML) techniques. Section 2.3 provides the through literature from Deep Learning (DL) approaches 2.4 provides the limitation of all approaches 2.5 provides summary of SPE. The chapter also gives systematic review of the models used in the current study.
- Chapter 3 discusses materials and methodology used for conducting the analysis. It gives an overview regarding dataset collection, preprocessing, feature extraction techniques, baseline models and discusses the proposed framework.
- Chapter 4 is results and discussion which presents results of the best baseline models applied and their limitations. It also provides an insight of the results improved by applying proposed framework for the particular dataset. The comparative analysis of previous studies is also presented which shows how classification has improved with application of deep learning models.
- Chapter 5 serves as the concluding section of this study, wherein the research effort is summarized and future directions are outlined, also presents the limitations of the study and the proposed framework with respect to the SPE, it also suggests future direction in the corresponding domain.

1.6 Summary

This chapter 1 discusses detail background and problem statement. The agile method is one of the famous around the world for project execution method. This research area has a lot more to cover and require detail data gathering. The chapter also highlights the advantages, areas of application and the contribution of this research in the field of software engineering.

Chapter 2

Related Work

2.1 Overview

The literature is thoroughly studied in this chapter, with a focus on various techniques of agile story point estimation. The existing story point estimation techniques are characterized into three classes. The first class is expert-opinion based approaches, consist of delphi method, planning poker and wideband delphi method. The second category is model based approach, it involves the **Functional Points Analysis (FPA)** and the **Constructive Cost Model (COCOMO)**. The third category is machine learning based approaches, which includes regression models, classification model and neural networks. Story point assessment is the primary job to begin a project, which implies that correctly done can be beneficial to project budget and overall team performance[1].

The other sections will describe the limitations of existing approaches of story point estimation in agile. The further discussions will focus on the use of deep learning approach in agile story point estimation. The evaluation and performance metrics for story point estimation. This discussion categorizes and organizes the different technologies and methods used for story point estimation in software engineering and making it relevant with Agile projects.

2.2 Conventional Approaches to Agile Story Point Estimation

The existing approaches involves the expert opinion based, the metrics based and machine learning based algorithms[4]. Expert opinion-based approaches to story point estimation depend on the collective judgement and experience of a team of software development experts or stakeholders. These approaches seek to capitalize on the information and expertise of the acquaintance people with the project requirements and development context. As mentioned earlier the expert-based approaches [5] to agile story point estimation are divided in three categories.

i) **Delphi Method**

The Delphi method is a structured communication technique to gather and refine expert opinions. In this technique a panel of experts provides anonymous and iterative estimates for story points. Multiple rounds of estimation and feedback are involved in the process. After each round, a facilitator collects, summarizes, and distributes the estimates to the experts, encouraging them to revise their estimates based on the response received. The goal is for the experts to collaborate and discuss their estimates in order to arrive at a

consensus estimate. To explain further, how delphi method is applied to story point estimation we consider following activities that will initiate delphi method.

- A) Preparation:** To prepare means to gather people from team or experts or relevant stake holders who have sound experience with Agile Story point estimation. The gathered group must have the ability to understand the backlog items in the form of story points and have knowledge about Agile practices.
- B) Anonymity:** Keep the identity of expert anonymous, which can be done via online survey. Or a facilitator might help in collecting the estimates and maintain anonymity.
- C) Initial Estimates:** Each member of the group make individual estimates to story points or backlog items of agile sprint. The members use fibonacci sequence starting from (1,2,3,5,..etc). The sequence is selected upon the task complexity or the relative effort required to execute story points.
- D) Consolidation:** The facilitator collects all estimates and then calculates mean or median of all estimates given by the participants. A consolidated estimate is then shared among group without disclosing individuals estimate values.
- E) Discussion and Feedback:** The participants view the consolidated estimates and compare it with their individual estimate. They share the rationale behind their estimates and analyze any difference appear between their estimated and consolidated estimate. They also discuss any insight that might help with better story point estimate.
- F) Iterative Process:** The facilitator streamlined the received feedback and share it with participants. The participants then revisit their estimates based on this feedback and insights provided by others. This process of feedback, consolidation and estimation revision is performed in iteration until a consensus is attained.
- G) Consensus:** Consensus is achieved when the estimates are converged and the differentiation between individuals estimates are decreased. This point of consensus allows choosing the right estimate which will help in planning and prioritization of story points.

The Delphi method assists in mitigating the biases that can arise in the estimation process due to group dynamics or dominant individuals. It promotes a more objective and informed estimation approach by providing anonymity and allowing iterative revisions based on feedback. It's important to note that the Delphi method is only one approach to estimating story points; different teams or organizations may prefer different techniques.

ii) Planning Poker:

Planning Poker is a widely recognized collaboratively developed estimation technique in Agile environments. It entails a group of experts, usually formed up of developers, testers, and other stakeholders. Each team member receives a set of story point cards representing various values (for illustration, the Fibonacci sequence starts with 1, 2, 3, 5, and 8). The team discusses the requirements of a user story, and each member chooses a card representing their estimate for the story points in private. The cards are revealed at the same time, and any significant differences are discussed. The process is repeated until an agreement on an estimate is reached.

iii) Wide Band Delphi Method:

Wideband Delphi is a Delphi method manifestation that incorporates additional discussion and feedback from experts. Experts in this approach meet in groups or workshops to exchange ideas and revise their estimates based on feedback from others. Wideband Delphi's iterative nature allows experts to refine their judgments through collaborative discussions, resulting in more accurate estimates.

The expert-opinion based approaches are helpful to build collaboration, communication and substantial relation among team. It enhances trust between the team and allows each member to express their individual experience. The expert opinions and feedback ameliorate members learning from sessions. Alternatively, the downside to approach is often more time consumption as per expected, not suitable where large team or project is under discussion. The tendentiousness of individual can delay the process and often lead to deficient estimates [2].

Wilson Rosa et.al discussed the nature of elements considered during the estimation of a project. Most of the agile software development require particular information at the beginning of project to make concrete estimations. The functional points in traditional approach were used and story points were used in non-conventional way more. The selection of either story points and function point is truly based on the nature, behavior, scope and business goals of the project [4].

2.3 Machine Learning Approaches to Agile Story Point Estimation

The model based approach is considered when team has past projects data. The data plays a vital role in learning the patterns and relationships between various factors. The past completed user stories in agile sprint gives a ground to estimate new user stories with some mathematical inference.

This research paper by Federica Sarro et al[9] published in the IEEE Transactions on Software Engineering. It focuses on the subject of effort assessment in the development of software projects. The paper presents an empirical study that investigates the predictability of human expert misestimations in terms of type, severity, and magnitude. The study uses machine learning techniques to predict these misestimations and evaluates their effectiveness in improving future effort estimates. The paper discusses an introduction that provides background information on effort estimation and the motivation for the study. It then presents the research questions that the study aims to answer, which include predicting the type and severity of misestimations, predicting the magnitude of misestimations, and enhancing effort estimates using machine learning. The methodology of this paper describes the datasets used in the study, which consist of real-world industrial projects from different application domains. It also explains the machine learning techniques and evaluation criteria used in the study. The results section presents the findings of the empirical study. It shows that human expert misestimations can be predicted to a certain extent, with average classification accuracies of 71% for type and 70% for severity. The predicted magnitude of misestimations is also close to the true amount of misestimation. In summary, this research paper explores the predictability of human expert misestimations in effort estimation for software development projects. It presents an empirical study using machine learning algorithms such as Linear Regression, RF, Naïve Bayes, KNN and LP that provides insights into the potential for improving effort estimates. The findings of the study contribute to the field of software engineering and can be useful for practitioners and researchers involved in effort estimation.

Vali Tawosi et.al. under taken the state-of-the-art model namely DeepSE, where they studied the efficacy of results by this model. They have taken another data set of TAWOS with 31,960 more issues. They used the algorithm of TF/IDF- SVM, and others that were used by the Choetkiertikul et.al. The results appeared to prove effectiveness of benchmark approach of DeepSE. The authors suggest that further studies were required to estimate agile software development. [2]

Ritu et.al. recapitulated a comparative analysis of machine learning algorithms. They analyzed *Naïve Bayes Algorithm, Regression Algorithm, SG Boost Technique, and Neural Networks*. The Naïve Bayes technique performed quite well with an accuracy metric of 80%, but its complex in nature. The most promising results were obtained by Story Point technique. [3]

Yuvna Ramchureetoo et.al.[7] worked on estimating agile user stories with multiple classifiers of machine learning. The Naïve Bayes method, Support Vector Machine algorithm, and Decision Tree algorithm were employed. These three algorithms were applied to user stories data and effort was estimated and measured by accuracy, precision, error rate, recall and F-measure. The comparison of these machine learning algorithms was taken into study and it was initiate that SVM algorithm has got the highest precision of 100%. The study was undertaken by WEKA tool. The preprocessing of data was done initially and then the processed data was used in WEKA. Accuracy of Naïve Bayes was 64% and Decision Tree got accuracy of 80%.

The paper by Muaz et al [8] focuses on developing an objective and exact effort approximation model for the projects of software using the Scrum methodology. The ultimate focus of the research is to evaluation the effort involved in software development projects using story points as a element of evaluation. The proposed model divides projects into phases and iterations, and estimates effort for every problem. The whole effort is then estimated using aggregation functions for iterations, phases, and the entire project. This approach provides elasticity to making decision in case of deviations from the plan of project. This study examines the execution of the suggested model and assesses its performance through the utilization of regression-based machine learning algorithms, including gradient boosting, SV Regression, RF Regression, and MLP. The results indicate that the method for estimating story points based on the story point-based estimation model performs significantly more than other models in terms of error rate. Overall, this research paper provides a framework for accurately estimating effort in software projects using story points and machine learning techniques. It offers a valuable approach for decision-makers and project managers in planning and managing software development projects.

The research conducted by Pendharkar et al. [9] focuses Bayes networks for estimation of software development effort. The paper examines the application of the Bayesian model in prior research and emphasizes the distinctive contributions made by this particular study. The paper additionally addresses the constraints of the study and proposes potential avenues for its enhancement. This paper presents a comprehensive examination of the COCOMO and

COCOMO II models, which are commonly used in cost estimation. The SLIM method, which is utilized for estimating software development effort and schedule, is also referenced. This document elucidates the utilization of Bayesian networks and Artificial Neural Network algorithms in the acquisition of joint probability distribution and the belief updating procedure for the purpose of modeling uncertainty in managerial decision-making. This study makes a valuable contribution to the existing body of literature by providing a benchmark for evaluating the effectiveness of Bayesian point software development forecasts. Additionally, it demonstrates the potential for integrating subjective managerial estimates into the Bayes-based model. In general, the document offers valuable insights regarding the estimation of software development effort and the utilization of Bayesian networks within this particular domain.

This study examines the utilization of developer features as a means to estimate story points in software projects, as investigated by Ezequiel Scott et al [10]. This study conducts a comparative analysis of various prediction models that employ a fusion of developer-related and textual elements in order to assess the story points assigned to subject reports. The paper elucidates various characteristics employed in the study, including developer repute, present developer workload, overall work capacity, number of developer comments, and textual attributes derived from issue descriptions. The article additionally addresses the utilization of Support Vector Machines (SVMs) in constructing predictive models. It also examines the assessment parameters employed to evaluate the algorithms' performance, including Accuracy, Mean Absolute Error (MAE), and Standardized Accuracy (SA). The objective of this study is to address research questions pertaining to the appropriateness of developer features in estimating story points, the accuracy of forecasting story points using developer features compared to textual characteristics, and the performance of developer features in terms of SA (Spearman's rank correlation coefficient) and MAE (Mean Absolute Error). The findings indicate that the models using developer features generally exhibit superior performance compared to the models incorporating textual features. However, it is important to note that the performance of these models differs across various projects. The study continues by recommending additional inquiry into the factors contributing to the diverse performance outcomes and the possible enhancements that might be implemented. This study offers valuable insights into the utilization of developer attributes for the estimation of story points in software projects, emphasizing the potential advantages of integrating these features into predictive models.

The study conducted by Ardiansyah et al. [11] examines the utilization of the analogy-based method for software project effort estimation. The objective of this study is to examine the

precision of estimating the level of effort necessary for software development projects. The author elucidates the significance of precise effort estimation in order to successfully accomplish projects within the designated timeframe and allocated resources. The text examines a range of methodologies and approaches employed in the estimation of software projects. These include expert judgment, Planning Poker, Function Point, COCOMO, Use Case Point, regression analysis, and Bayesian Belief Network. The study centers on the analogy-based estimation approach, which entails the comparison of the project to be assessed with historical data derived from like projects. The study outlines the several processes involved in the process of analogy-based estimation, encompassing the assessment of similarity, adaptation of analogies, computation of estimates, and evaluation of the model. Additionally, the paper examines the many parameters employed in the estimation process, including Euclidean distance, Manhattan distance, and Minkowski distance. The findings of the research indicate that the most effective approach for estimating software project work through analogy approaches involves employing the Manhattan distance metric. This method demonstrates a level of accuracy with a 50% Mean Magnitude of Relative Error (MMRE), a 28% Median Magnitude of Relative Error (MdmRE), and a 48% prediction within 25% of the actual effort. This implies that the model described in the research can be effectively employed for accurate estimation of software project effort.

The present research study conducted by Tawosi et al. [12] examines the correlation between story points and development effort within the context of Agile software projects. The objective of this study is to examine the relationship between estimated narrative points and the actual effort needed for the completion of a software development activity. The researchers employ proxies as a means of approximating the duration of development and evaluating their efficacy in accurately determining the true level of effort required. The document additionally addresses the methods employed in the study, encompassing the dataset, statistical analyses, and research inquiries. This study offers valuable insights into the degree of consistency in assigning story points throughout a project and delves into the difficulties associated with estimating effort in the context of Agile software development. The document provides valuable insights for anyone involved in software development, project management, and research who have an interest in Agile estimate methodologies and the correlation between narrative points and development work.

The study conducted by Abdelali Zakrani et al. [13] examines the process of effort estimation in the context of agile software development. The article explores the utilization of user stories

and narrative points as a means of estimating the level of effort necessary for software development. The paper also examines various estimation methodologies, including expert opinion, planning poker, and use case points. The study emphasizes the subjective character of these methodologies and the possibility of errors and discrepancies in estimations. The subsequent section of the paper presents the application of machine learning methods, notably support vector regression (SVR) that has been tuned using the grid search strategy, in order to enhance the precision of effort estimation. This paper introduces a theoretical model and optimization approach, accompanied by an empirical assessment conducted on past agile software projects. The evaluation of the model's accuracy is conducted using various metrics. This study offers valuable insights into the difficulties and strategies involved in estimating effort in the context of agile software development. Specifically, it explores the utilization of machine learning techniques and optimization methods for this purpose.

Yasir Mehmood and colleagues [14] discuss the topic of software effort estimation within the realm of software development. The significance of precise effort estimation and the difficulties linked to it were emphasized. The study offers a comprehensive examination of several estimation methodologies, encompassing algorithmic, non-algorithmic, and machine learning methodologies. Additionally, this paper provides a comprehensive analysis of existing literature pertaining to software effort estimate, with a particular emphasis on the utilization of use case points and expert-based estimating techniques. The estimation of effort plays a critical role in ensuring the success of software development initiatives. Various methodologies exist for estimating software effort, encompassing algorithmic, non-algorithmic, and machine learning techniques. The primary focus of this study pertains to the utilization of use case points and expert-based estimation techniques in the estimation of software work. This paper presents a comprehensive assessment of pertinent studies that have been undertaken to assess the enhancement in accuracy achieved through the utilization of ensemble and solo machine learning approaches. A checklist for quality assessment is employed to evaluate the chosen studies in accordance with their research objectives, study design, estimation methodologies, data collection techniques, data analysis procedures, and statistical approaches. In general, this study offers a thorough examination of software effort estimating methods and includes a methodical analysis of pertinent research articles. This study is beneficial for scholars and practitioners in the domain of software development who possess an interest in enhancing the precision of effort estimation.

In their study, Shashank et al. [15] conducted an empirical evaluation of machine learning models in order to estimate the work required for agile software development, specifically focusing on the utilization of story points. This paper examines the significance of effort estimation in the context of agile software development and the corresponding issues that arise. This paper elucidates the story point methodology, a widely adopted technique for estimating the effort required in software development projects that employ agile approaches. Additionally, this study introduces various machine learning methodologies, including decision tree, stochastic gradient boosting, and random forest, which have the potential to enhance the precision of effort estimation predictions. This study offers a comparative examination of the aforementioned strategies and provides a comprehensive discussion on their respective performance. The research findings underscore the necessity for additional inquiry in this domain and indicate future expansions to the proposed methodology. This study provides valuable insights for software development firms and scholars who are interested in enhancing the precision of effort estimation in agile software development projects.

The present study pertains to the domain of agile software development, with a specific emphasis on the estimation of efforts required for different activities within agile projects. In her work, Lan Cao [16] examines the complexities and methodologies associated with the estimation of effort for tasks related to feature development, issue repair, and refactoring. This study also investigates several estimation methodologies employed in agile software development, including expert-based techniques such as planning poker and story points, alongside emerging techniques like machine learning. The paper presents a comprehensive account of the research design and technique employed for the purpose of gathering and examining data pertaining to the precision of estimations. This emphasizes the significance of documenting and evaluating comprehensive data pertaining to the estimation process, a task that is commonly perceived as burdensome inside many organizations. This study provides a significant contribution to the comprehension of the difficulties and optimal approaches involved in estimating effort for diverse tasks within the context of agile software development. This study offers valuable insights into the estimation process, the methodologies employed, and the significance of data collecting in enhancing the accuracy of estimations.

The study work by Yanming Yan et al. [17] centers around the application of predictive models within the field of software engineering. This paper offers a detailed and systematic evaluation of several predictive modeling approaches and their application in a range of software engineering jobs. The objective of this study is to provide a comprehensive overview,

categorize, examine, and suggest potential avenues for further research, all based on empirical data. The author examines the many difficulties and potential advantages associated with the utilization of predictive models within the field of software engineering. This statement underscores the significance of utilizing high-quality datasets and acknowledges the influence of publication bias on the formation of conclusions. The research approach employed in this study encompasses three distinct research inquiries: the examination of publication distribution pertaining to predictive models, the categorization and distribution of the predictive models utilized, and the evaluation of the application of predictive models in certain software engineering jobs. The findings of the study are provided, encompassing the quantity of pertinent research articles found and the dispersion of these articles throughout various publication platforms. Additionally, the paper presents a comprehensive overview of the predictive modeling approaches employed in the aforementioned investigations and examines the distribution of these techniques within the research.

2.4 Deep Learning Approaches to Agile Story Point Estimation

The study conducted by P. Suresh Kumar and colleagues [18] examines the topic of software effort estimate, with a specific emphasis on the application of machine learning techniques, notably neural networks, for the purpose of predicting software effort. The author highlights the need of precise software estimating in the effective management of intricate and extensive software projects. This paper provides an overview of the several stages encompassed in the execution of a systematic literature review (SLR) pertaining to software effort estimation. These stages encompass the formulation of research inquiries, development of a search strategy, selection of relevant studies, synthesis of data, and the subsequent reporting of the review findings. Additionally, this underscores the necessity for future research to prioritize the development of more precise assessment methods that mitigate the risk of exceeding budgetary limits and compromising quality standards. The review article presents a collection of fundamental inquiries that the systematic literature review (SLR) endeavors to address. These inquiries encompass commonly employed datasets, methodologies, accuracy metrics, and amalgamation approaches utilized in the prediction of software effort. The paper also examines the prominent publications within the discipline, the years characterized by significant research activity, the specific categories of software effort that are examined, and the effectiveness of intelligent approaches across diverse datasets. The document finishes by providing a comparative analysis of existing literature pertaining to software effort estimation and proposes potential avenues for further research.

Micheal Fu [19] presented a "LineVul: A Transformer-based Line-Level Vulnerability Prediction". The paper discourses the use of DL models, specifically the BERT architecture, for predicting vulnerabilities at the line-level in source code. It addresses the limitations of existing vulnerability prediction approaches that focus on file or function-level granularity, and proposes a novel approach that leverages the mechanism of attention inside the BERT model architecture. The study explains the concept of line-level vulnerability prediction and highlights the challenges associated with it. It discusses the importance of explain ability in AI models and how the attention mechanism can be used to provide meaningful explanations for predictions. The study provides a summary of the research paper, including key points such as the use of BERT architecture, the attention mechanism, and the focus on line-level vulnerability prediction. It emphasizes the novelty of the approach and its potential usefulness in improving the accuracy and explain ability of vulnerability prediction models. In conclusion, this study presents a research paper that introduces a new approach, LineVul, for line-level vulnerability prediction using deep learning models.

The research conducted by Zhang et al. [20] focuses on enhancing the efficacy of function point-based software size estimate. This study presents a novel deep learning-based approach for automating the manual function point analysis process. The effectiveness and efficiency of the proposed model are evaluated to assess its potential for improving the overall efficiency of function point analysis. The author provides an analysis of the research inquiries, the methodology employed, the preparation of data, and the outcomes of the tests carried out to authenticate the suggested approach. The research inquiries center on evaluating the appropriateness, precision, and enhancement in efficiency of the offered model in comparison to the manual approach of function point analysis. The process encompasses the development of a Named Entity Recognition (NER) model based on deep learning principles, specifically utilizing a Bidirectional Long Short-Term Memory (BiLSTM) architecture with a Conditional Random Field (CRF) layer. The process of data preprocessing encompasses several tasks, such as cleansing and annotating requirement papers, segmenting words, and labeling function point elements. The experimental results demonstrate that the model provided in this study attains elevated levels of precision and F1 scores in the domain of function point analysis. Furthermore, it exhibits a notable enhancement in the efficiency of software size estimation. The study introduces an innovative methodology for enhancing the effectiveness of function point-based software size estimate through the utilization of deep learning techniques. The model under

consideration has favorable outcomes in terms of both accuracy and efficiency, hence offering potential advantages for software project management.

Hung Phan et al [23] paper presents a model called HeteroSP, which uses graph neural networks to estimate story points in software issues. The study explains the motivation behind the research, the dataset used, the approach taken, and the research questions addressed. The paper addresses the problem of estimating story points in software issues, which is important for effort estimation in Agile methodology. The HeteroSP model uses graph neural networks to capture the structure and content of software issues. The model constructs a heterogeneous graph representation of the issues. The graph is built by preprocessing the textual information of the issues and creating nodes for words, sentences, and code parts. The model then uses a series of graph transformer layers to propagate information and make predictions. The paper presents experiments and results for different research questions, including within-project repository and cross-project repository estimation, the impact of preprocessing, and the use of parse trees. The HeteroSP model shows promising performance in estimating story points and outperforms existing approaches. The model is designed to handle the combination of natural language and code parts in software issues and shows good performance in estimating story points.

Jirat et al [22] conducted study on software development projects to understand the factors that influence the changes in story points (SP) of work items. The study focuses on six dimensions: activity, collaboration, completeness, experience, readability, and text. It explores various metrics within these dimensions and their impact on SP changes. The document explains the method of collection and analyzation of the data, including the use of JIRA work items and different classification algorithms. The study aims to provide insights into the factors that can help predict SP changes in software development projects. The key points of the document include the identification of influential metrics, such as stable story points and word-based content, and the reflection that certain types of information changes, such as changing scope and task adding, are more common in work items with SP changes. The study also confers the suggestions of future research directions.

In their study, Panda et al. [23] discuss the use of the story point technique for agile software work estimate. This paper elucidates the use of agile methodologies in the realm of software development, while also emphasizing the significance of accurately predicting the magnitude and intricacy of the products to be constructed. The notion of narrative points and their use in estimating work within the context of agile software development is introduced by the author. The research also examines the use of several neural network methodologies, including General

Regression Neural Network, Probabilistic Neural Network, Group Method Data Handling networks, and cascaded networks, in order to enhance the optimization of the estimate procedure. The study presents a comprehensive examination of the suggested methodology and the sequential procedures included in forecasting effort via the utilization of diverse neural network models. Additionally, the document discussed the assessment criteria used for assessing the performance of the models.

In their study, Fu et al. [24] provide a novel methodology known as GPT2SP for the purpose of Agile story point estimation. This strategy leverages a pre-trained language model that utilizes a Transformer-based architecture, hence enhancing its ability to comprehend the interconnections between words and ultimately leading to improved estimate accuracy. The study assesses the suggested methodology using a dataset consisting of 23,313 problem instances derived from 16 open-source software projects. The evaluation includes the comparison of the proposed technique to 10 established baseline methodologies, including both within-project and cross-project situations. The findings indicate that the GPT2SP methodology attains a median Mean Absolute Error (MAE) of 1.16, surpassing the precision of current baseline methodologies for both within-project and cross-project predictions. The research furthermore does an ablation analysis to demonstrate the considerable enhancement of Agile story point estimation via the utilization of the GPT-2 architecture in the proposed technique, resulting in improvements ranging from 6% to 47%. This underscores the noteworthy progress of artificial intelligence in the context of Agile story point estimation. In conclusion, this research article presents the creation of a proof-of-concept tool aimed at enhancing practitioners' comprehension of story point estimates. Additionally, a survey study is conducted with Agile practitioners to assess the effectiveness and reliability of AI-based story point estimation accompanied by explanations.

The significance of effort estimating in software project management, namely in the areas of project scheduling and evaluation, is discussed by Morakot Choetkiertikul et al [25]. This statement emphasizes the prevalent occurrence of overruns in both costs and schedules in software projects and the negative consequences that inaccurate estimations have on project results. This article centers on the need of predicting the work required to complete individual user stories in agile development environments, characterized by iterative cycles and incremental software development. This research presents a novel prediction model that utilizes a hybrid approach, combining two deep learning architectures, namely long short-term memory (LSTM) and recurrent highway network (RHN), to estimate narrative points. Additionally, it

offers a complete dataset for estimating tale points based on a points-based estimate technique. Empirical examination of the suggested strategy reveals its superiority over conventional baselines and alternative methods in terms of Mean Absolute Error, Median Absolute Error, and Standardized Accuracy. This work presents a novel prediction model for the estimation of story points in agile projects, therefore making significant contributions to the field. The prediction model utilizes a hybrid approach by integrating two deep learning architectures, namely long short-term memory (LSTM) and recurrent highway network (RHN). This study presents a comprehensive dataset of 23,313 problems extracted from 16 open source projects, with a focus on narrative points-based estimate. This study aims to empirically evaluate the suggested methodology and compare its performance against three commonly used baselines and six alternative methods in terms of Mean Absolute Error, Median Absolute Error, and Standardized Accuracy. The results consistently demonstrate that the proposed approach outperforms all the compared methods across these evaluation metrics. This study aims to address the existing research vacuum in the field of effort estimate in agile projects, with a specific focus on estimating the work necessary for the completion of user stories or problems. This system provides a comprehensive trainable prediction framework that encompasses the whole process, from initial data to forecast results, eliminating the need for human feature engineering.

2.5 Limitations in Estimation of Story Point Using DL Approaches

Hoa Khanh Dam et al [26] discusses the concept of explainable software analytics and its importance in the field of software engineering. It argues that while accurate predictions are crucial, it is equally important for software practitioners to understand the rationale behind those predictions. The study outlines a research roadmap for achieving explainability in software analytics models, drawing on social science, explainable artificial intelligence, and software engineering. Software analytics has gained attention but has not been widely adopted due to a lack of trust in the predictions without understanding the reasoning behind them. Explainability should be a key measure for evaluating software analytics models. There are different types of explanations in software engineering, and software engineers have different preferences for the types of explanations they are willing to accept. Transparency at different levels of the model, such as the entire model, individual components, and the learning algorithm, can contribute to explainability. Designing architectures that self-explain decision making, using attention mechanisms and natural language processing, can enhance explainability. Using external explainers, such as simpler models or interpretable structured knowledge, can help mimic or explain the behaviors of complex models. Evaluating explainability can involve measures such

as the size of the model, conducting experiments with practitioners, and formalizing evaluation methods using evidence-based approaches. Lastly, this study emphasizes the importance of explainability in software analytics and provides a roadmap for achieving it. It highlights different approaches and evaluation methods that can be used to make software analytics models more understandable and trustworthy for software practitioners.

2.6 Summary

In this chapter we discussed the related literature studied during the carrying out our research. The related work suggests that use of hybrid model is not widely practiced by many other researchers to solve this problem. Since our proposed model is innovative and novel in terms of applying GPT2-CNN a hybrid model to solve the problem of story point estimation in agile projects.

Chapter 3

Methodology and Framework

3.1 Overview

Many statistical and software-based solutions have been proposed and improved over the years to address the main problems with story point estimation in agile projects. But even systems already in place are still susceptible to the accurate story point estimation problem in software projects, as well as lack the adaptability to deal with the diversity of real-world settings. [1]

This chapter introduces an enhanced agile story point estimation method that outperforms the existing Deep-SE method. To begin, data provided by Deep-SE is taken to study, and then pre-processing the data to make it clean for model. Next step involves tokenization to create token embeddings of each story point title and description. The model actively learns token-wise dependencies for word to word rather than making enormous size of word vocabulary. This architecture allows model to predict with efficiency. It can also give answers that are more semantically relevant than RHWN and LSTM due to its great efficiency and ability to implicitly understand the semantic structure. Therefore, a number of ablation investigations were carried out to determine the most effective hybrid model.

3.2 Overview of Transformers (GPT2) Generative Pretrained Transformers

Transformers are a specific kind of neural network structures that find use in the domain of natural language processing (NLP) endeavors. The models are founded upon the concept of self-attention, enabling them to acquire knowledge of distant relationships among words within a sequence. Furthermore, they use word order embedding to consider the positional information. The first introduction of Transformers occurred in the scholarly work authored by Vaswani et al. [27] The study demonstrated that transformers have the capability to attain cutting-edge performance on a range of natural language processing (NLP) activities, including translation by machines, summarization of text, and query responding. Subsequently, transformers emerged as a highly favored neural network framework for Natural Language Processing (NLP). They are being used to get cutting-edge outcomes in a diverse array of jobs, encompassing:

- **Automated translation:** Transformers have been used to improve the accuracy of machine translation systems by a significant margin.

- **Text summarization:** Transformers have been used to generate summaries of text documents that are both accurate and concise.
- **Question answering:** Transformers have been used to develop question answering systems that can answer questions posed in natural language.
- **Natural language inference:** Transformers have been used to develop systems that can determine whether a sentence implies another sentence.
- **Text generation:** Transformers have been used to generate text, such as poems, code, and scripts.

The transformer architecture is composed of two main components: an encoder and a decoder. The encoder function processes the input sequence and transforms it into a series of hidden states. The encoder function processes the input string in a sequence and transforms it into a series of hidden states. The encoder is made up of stack with self-attention layers. Each self-attention layer takes the previous hidden states and produces a new set of hidden states. The hidden states from the last self-attention layer are then passed to the decoder.

The decoder utilizes the hidden states obtained from the encoder in order to construct the output sequence. The decoder is composed of a series of self-attention layers. In each self-attention layer, the preceding hidden states and the output tokens from the preceding phase are used to generate an entirely new set of hidden states. The subsequent resulting token is generated using the hidden states derived from the preceding self-attention layer. Following that, the decoder utilizes the secret states obtained from the encoder to produce the resultant sequence. The encoder and decoder consist of a stack with self-attention layers. The mechanism of self-attention enables the transformer model to capture and understand the relationships and interdependence among words in a given sequence, even when the words are distant from one another.

$$Attention(Q, K, V) = Softmax\left(\frac{(QK)^T}{\sqrt{d_k}}\right)V \quad (1)$$

Where,

Q is the query vector, which represents the attention weights for each word
 \in sequence of input string

K is the key vector, which represents the significance of each word \in the input sequence

Value vector, which represents the information about each word \in the input sequence

d_k is dimension of the key \wedge value vectors

Softmax function which normalizes the attention weights so that they \sum^1

The attention mechanism works by first calculating the dot product between the query vector and each of the key vectors. This produces a score for each word in the input sequence, indicating how relevant it is to the query. The scores are then normalized using the softmax function, and the resulting attention weights are used to weighted sum the value vectors. The weighted sum is then the output of the attention mechanism. The attention mechanism allows the transformer to learn long-range dependencies between words in the input sequence. This is because the attention weights are not limited to the immediate context of the query word. Instead, they can be influenced by words that are far away in the sequence. This allows the transformer to learn the relationships between words that are not explicitly stated in the sequence. The transformer is a powerful neural network architecture that can be used for a variety of NLP tasks. It has been used to achieve state-of-the-art results on a wide range of tasks. It is still being used to develop new and innovative NLP applications.

3.3 Proposed GPT2-CNN Hybrid Learning Model for Story Point Estimation in Agile Projects

The GPT2-CNN hybrid model was created which focused on story point estimation tasks and can operate without stacking convolution to extract structural data more effectively. Similar to word embedding in NLP, title and description are tokenized and turned into token embedding in our model. GPT2 actively understands token-wise dependencies for input text rather than directly computing the semantic of each word in a sentence. GPT2 enhances the given issue semantic information in a project with excellent efficiency. Along with being highly effective, GPT2-CNN can intuitively learn the semantic information and hence produce results that are more accurately predicted than Deep-SE. Nevertheless, obtaining comparable performance with Deep-SE often requires a large amount of training time or extra supervision else cannot perform as expected due to the lack of inductive biases.

Suggested model's overall layout is shown in *figure 2.2*. To start, we preprocess the data into a clean data removing unnecessary elements like html tags, repeated rows and unintended data. The Multi-head Self-Attention based Transformer module will receive the data of issues descriptions as input and produce word embedding vectors. The embedded vectors will be utilized to give CNN input which will then perform the remaining task of story point estimation.

3.3.1 Data Pre-processing and Acquisition

The data-preprocessing and data acquisition are two-fold process. The data acquisition is the process of collecting data from online resources. The process is time taking and confidential at times. The author of base-line approach took months to acquire data and provide it to other for open research. The second step is data pre-processing, where it requires the molding of data into our model required shape. So that model perform well on tuned data. Both of the two-fold process are described in detail.

Data Acquisition

The data set acquired for this research is being made by Choetkiertikul[27], they have gathered data from 16 different repositories and 9 different projects using JIRA. The overall dataset size is 16.6MB. The dataset has four main features, which are **issueskey**, **title**, **description**, **story points** and **split_mark**. The details of each feature are mentioned below in **table 3.1**:

- **Issueskey:** The issue(s) key is a a unique identifier, which is used to distinguish one issue from another issue. For more information, issues are also represented as sprints and backlog items in agile software development. Issue key is normally taken as integer value.
- **Title:** The title represents the main context of issue identification in natural language in brief form. The title often gives hints to the reader about what is going the happen in the description of story point.
- **Description:** The description of story points tells the reader who might be a senior manager, project owner, team lead, senior developer, senior quality assurance engineer, junior developer, junior quality assurance analyst or a business analyst or some other stake holders of the project that have direct link with the execution of a project. The description involves the explanation of task, workable item ny the team. Descriptions should be written or gathered in a manner that is understandable by every stake holder.
- **Story points:** The story points are numbers which embodies the effort required to complete a task. The effort is calculated by senior team lead or project manager has experience in executing same kind of projects. This number also denoted the intuition of project manager.
- **Split_mark:** The split mark is use to denote the preprocessing done on data, while each mark classify data in train, test and validation part.

While training and testing a project file, we have seen the size of each file determines the time required to train and test. Those project files which has less number of issues shown more accuracy. The files which contain more number of issues require few more seconds to train. This also impacts the overall accuracy of the proposed model.

Data Pre-Processing

The process of pre processing is carried out to ensure that data is clean and ready to use for further processing. In this process we analyzed each data file, figure out the anomaly in data. The data is extracted from web pages of Jira, so it contains html tags and other web page elements not required. The data has some missing values which also affects the model performance so we clean all such things and convert it into a useable format.

Table 2.1 Detailed description of dataset utilized for GPT2 CNN Hybrid Learning Model

Sr No.	Repository	Project Name	Size in Memory	No. of total Issues in a file
1	Appcelerator	Appcelerator Studio	2509kb	2920
2	Appcelerator	Aptana Studio	770kb	830
3	Atlassian	Bamboo	488kb	522
4	Atlassian	Clover	335kb	385
5	Lsstcorp	Data Management	1774kb	4668
6	Dura Space	DuraCloud	282kb	667
7	Atlassian	JIRA Software	193kb	353
8	Apache	Mesos	2219kb	1681
9	Moodle	Moodle	640kb	1167
10	Mulesoft	Mule	529kb	890
11	Mulesoft	Mule Studio	389kb	733
12	Spring	Spring XD	2172kb	3527

13	Talendforge	Talend Data Quality	988kb	1382
14	Talendforge	Talend ESB	794kb	869
15	Appcelerator	Titanium SDK/CLI	2630kb	2252
16	Apache	Usergrid	357kb	483

3.3.2 Sub-word Tokenization and Byte Pair Encoding (BPE)

Tokenization is the process of making token from the given input text. Since text is hard for most of machine learning tasks. The issues are written in natural language which is not understandable by machine. Tokenization can sometimes be more complex than just splitting by spaces. It needs to handle punctuation, contractions, special characters, and other linguistic nuances to ensure that the resulting tokens accurately represent the meaning of the text. This is particularly important for languages with complex grammar and syntax. To resolve, natural language processing techniques are devised to make tokens which will represent a number against each word. The previous methods applied by Deep-SE utilizes the bag-of-words approach, the creation of a vocabulary encompassing all distinct words present within the corpus. Subsequently, each document within the corpus is represented as a vector in the high-dimensional space defined by this vocabulary. The vector's dimensions correspond to the words in the vocabulary, and its values reflect the frequency or presence of each word within the document. This transformation facilitates the conversion of textual data into a numerical format amenable to mathematical manipulation and analysis. This approach has limitations which inherent disregard for word order and grammar precludes it from capturing nuanced semantic relationships present in natural language. The algorithm's sensitivity to word frequency can also be a double-edged sword, as it may inflate the significance of common but semantically uninformative terms, such as stop-words.

Micheal Fu[26], uses Byte Pair Encoding (BPE), a subword tokenization technique that has gained prominence in the domain of NLP for its efficacy in segmenting text into meaningful units. This technique stems from the inherent need to handle morphologically rich languages and address the challenges posed by rare or out-of-vocabulary words in various NLP tasks. The fundamental premise of BPE resides in its data-driven approach to tokenization. It operates by iteratively merging the most frequent pairs of characters or subword units in a corpus, thereby forming new compound units. The process is guided by a heuristic that aims to maximize the compression rate, effectively leading to the formation of a shared vocabulary of subword units. By encoding text into

subword units, BPE effectively navigates the trade-off between granularity and vocabulary size. This granularity permits the representation of complex words and phrases as compositions of subword units, rendering the technique particularly well-suited for story point estimation.

The BPE algorithm, we used it to perform two operations subsequently, 1 and 2 as shown in the **figure 3.3**. The **1** represents the merge operations of which means that how a word should be split or divided on the basis of its location in the sentence. A merge operation entails the fusion of the most frequently co-occurring pair of characters or subword units within the training corpus. The frequency criterion is pivotal, as it guides the algorithm's prioritization of merging operations to maximize the compression of the data. As iterations of these merge operations persist, a hierarchy of subword units is progressively established, with higher-frequency pairs being amalgamated prior to those occurring less frequently. The significance of merge operations is twofold. First, it enable the construction of a shared vocabulary of subword units, thereby mitigating vocabulary discrepancies that often beset languages marked by agglutinative or inflectional tendencies. Second, it facilitate the representation of complex words as compositions of subword units, accommodating a granularity capable of capturing both morphological variations and semantic nuances.

With the subsequent process, we put the tokenization pipeline into action to deliver efficient token ids. As shown in **figure 3.3**, we first pre-process the data. After data cleaning we perform the action of splitting dataset in 3 ratios, one for training, second one is for testing and third one is for validation. Then, we input the sequence of trained dataset comprising of 16 projects to pretrained GTP2. The GPT2 model will perform the BPE tokenization, where a simple sentence is break in common words being coming together. For example, the story has description explanation as shown in figure 1 where expressing is used as a middle word in the description context, “expressing” will break into subword of ‘express’, ‘ing’; this allows the model to understand the actions a word is performing in that particular sentence. After the subword tokens are formed they are merged together by merge operation so that it does not lose any detail from story description as shown in **figure 3.3**. The vocabulary size has reduced from 185,625 to 50,257 vocab.

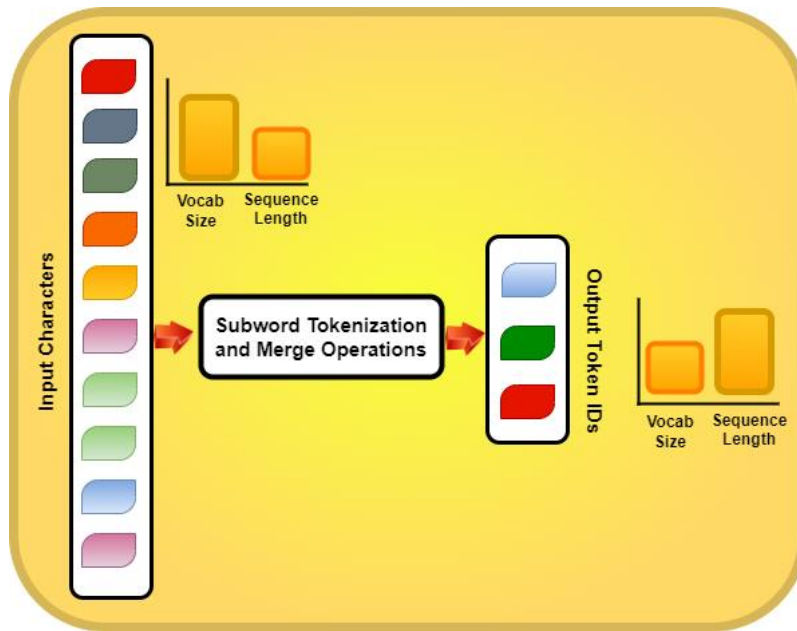


Figure 3.1 GPT2 sub-word tokenization is shown where applying Byte Pair Encoding (BPE) is used for the task of making tokens

3.3.3 Convolutional Neural Networks

We employed a Convolutional Neural Network model on the input tokens received from GPT2. The reason behind choosing this combination of hybrid deep learning model is to check whether running the same dataset will improve the results in contrast to Deep-SE model. We employ a standard module, 1D convolutional neural network (CNN) which is used to process 1D data, such as text or audio. 1D CNNs work by applying a series of filters to the input data. These filters are designed to detect specific features in the data, such as word. The output of the filters is then passed through a series of layers that learn to recognize these features.

The input to the CNN is a sequence of tokenized IDs. The first layer of the CNN is a convolutional layer. This layer applies a series of filters to the input data. The filters are designed to detect specific features in the data, such as word order or the use of certain words. The output of the convolutional layer is a sequence of feature maps. The next layer is a pooling layer. This layer reduces the size of the feature maps by taking the maximum value in each window. This helps to reduce the number of parameters in the model and makes it more efficient. The final layer is a fully connected layer. This layer takes the output of the pooling layer and learns to classify the input text. A detailed architecture is shown in **figure 3.2**

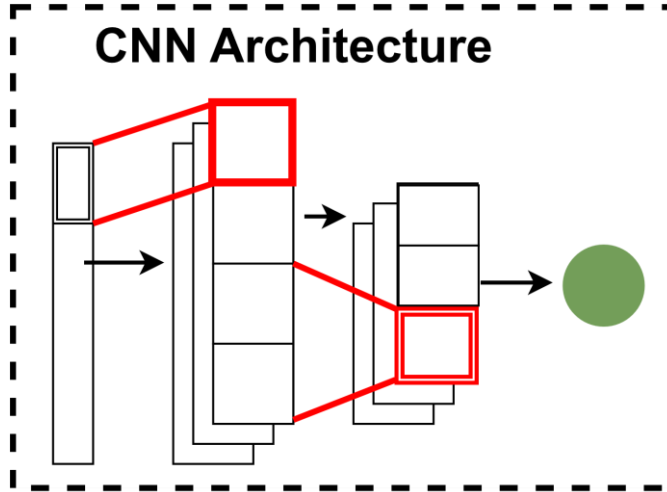


Figure 2.2 CNN Architecture Displaying all layers and transition of data between layers is shown

3.3.3.1 Embedding Layer of CNN

Given an input sequence of integer-encoded tokens, denoted as a matrix $X \in \mathbb{R}^{L \times N}$, where L is the sequence length and N is the vocabulary size, the embedding layer transforms each token into a continuous-valued vector in a learned embedding space of dimension D , resulting in an output tensor $E \in \mathbb{R}^{L \times D}$.

The embedding process is described as:

$$E_{ij} = \text{Embedding}(X_{ij}), \forall i \in \{1, 2, 3, \dots, L\}, j \in \{1, 2, 3, \dots, N\} \quad (2)$$

3.3.3.2 1D Convolutional Layer of CNN

In order to keep position information intact we also add a 1D learning position. Let E , be the output tensor from the embedding layer, with $E \in \mathbb{R}^{L \times D}$. The 1D convolutional layer applies a set of F learnable filters, each with a kernel of size K and C channels. The operation results in C feature maps, which are then passed through a bias term b and an activation function σ . The feature map tensor $C \in \mathbb{R}^{(L-K+1) \times C}$ is obtained as follows:

$$C_{ij} = \sigma\left(\sum_{k=1}^K \sum_{j=1}^D E_{i+k-1,j} \cdot W_{kj}^c + b_c\right), \forall i \in \{1, 2, 3, \dots, L - K + 1\}, c \in \{1, 2, \dots\} \quad (3)$$

Where, W_{kj}^c denotes the weight of the k_{th} element of the kernel of the c_{th} filter in relation to the j_{th} dimension of the embedding space.

3.3.3.3 Global Max Pooling Layer CNN

The global max pooling layer operates on each feature map from the previous convolutional layer independently. Given the feature map $C \in R^{(L-K+1) \times C}$, the global max pooling operation extracts the maximum value from each channel, resulting in a pooled tensor $P \in R^C$ Transformer inputs.

$$P_c = \max_i C_{ic}, \forall c \{1, 2, \dots, C\} \quad (4)$$

3.3.3.4 Dense Layer CNN

The dense layer performs a linear transformation on the input vector P with weights $W \in R^{C \times O}$, where O is the output dimension. The transformation is followed by an element-wise application of an activation function σ . The output tensor $Y \in R^O$ is computed as:

$$Y = \sigma(P.W + b) \quad (5)$$

Where “.” represents matrix-vector multiplication, b is bias vector, and σ is activation function.

3.4 Ablation Experiments

There are 4 other experiments carried out during the study:

3.4.1 GPT2-CNN Model

In this experiment we perform the analysis on GPT2-CNN where we didn't apply any changes to default CNN. We do apply tokenization process through GPT2 which is pre-trained on large corpora by the creators. The reason behind using unchanged parameters is to check the working and response of model on our problem of agile projects story point estimation. The results are discussed in the next section.

3.4.2 GPT2- LSTM Model

In the realm of Agile development of software, the precise assessment of work effort has significance in facilitating efficient planning for projects and allocation of resources. In order to address this difficulty, sophisticated methods using artificial neural networks, namely the Long Short-Term Memory (LSTM) architecture, are used in conjunction with pre-trained models like

GPT-2. This approach shows great potential in automating and improving the process of estimating story points.

The process begins with the utilization of GPT-2, a state-of-the-art language model which is pretrained on a diverse corpus of textual data. GPT-2 demonstrates exceptional proficiency in understanding natural language and generating coherent text. In story point estimation, GPT-2 is leveraged to tokenize the textual description of user stories, effectively segmenting the input into discrete units representing sub-word components. This tokenization process is essential to create a structured input representation that can be seamlessly integrated into subsequent stages of the LSTM-based estimation model.

Following tokenization, the resulting tokens are provided as input to the LSTM architecture. LSTMs are a class of recurrent neural networks (RNNs) explicitly designed to capture temporal dependencies and patterns within sequential data. In Agile story point estimation, LSTMs can effectively exploit the sequential nature of the tokenized user story descriptions, discerning intricate linguistic nuances and contextual cues that may impact the estimation process.

The LSTM model's structure incorporates memory cells, augmented with gating mechanisms that disseminate information throughout the network. The gates, consisting of forget, input, and output components, enable the LSTM to selectively retain relevant information over extended sequences, while alleviating the vanishing gradient problem that hampers conventional RNNs. Consequently, the LSTM-equipped model can inherently grasp long-range dependencies present in user story descriptions, which are often crucial for accurate story point estimation.

By employing a pretrained GPT-2 model for tokenization and subsequently feeding these tokens into an LSTM architecture, the story point estimation process benefits from a hybrid approach that amalgamates advanced natural language understanding capabilities with the ability to capture intricate contextual relationships. This amalgamation empowers the model to interpret and dissect the textual descriptions effectively, potentially leading to more accurate and consistent story point estimates.

In conclusion, the integration of pretrained GPT-2 and LSTM architecture within the realm of Agile story point estimation showcases a progressive convergence of sophisticated language processing and sequential modeling techniques. This hybrid methodology addresses the

subjectivity and complexity associated with manual story point estimation by automating the process through comprehensive linguistic comprehension and contextual analysis, ultimately improving the accuracy and efficacy of Agile project planning and execution.

3.4.3 GPT2-LSTM-CNN Model

The amalgamation of advanced neural network architectures—GPT-2, LSTM, and CNN—can potentially enhance the accuracy and efficiency of this estimation process. GPT-2, a renowned pre-trained language model, demonstrates exceptional proficiency in understanding and generating human-like text. Its application in word tokenization significantly enhances the representation of textual input pertinent to agile story descriptions. The tokenization process involves segmenting the text into discrete units, enabling a more structured input for subsequent neural network layers.

Leveraging LSTM (Long Short-Term Memory) following GPT-2's tokenization serves a dual purpose. Firstly, LSTMs possess the inherent capability to model sequential dependencies within data, a trait crucial for capturing the nuanced relationships often present in agile story narratives. These networks excel at handling temporal sequences and preserving contextual information over extended spans. Secondly, LSTMs aid in dimensionality reduction, a critical endeavor to manage the computational complexity associated with subsequent layers.

Subsequently, the integration of CNN (Convolutional Neural Network) exhibits its utility in enhancing feature extraction and dimensionality reduction within the context of agile story point estimation. By employing CNNs, we capitalize on their proficiency in capturing hierarchical patterns present within the segmented text tokens. Convolutions performed across these tokens enable the identification of localized features, effectively reducing the dimensionality while retaining relevant information. Pooling operations further enhance the network's ability to extract salient features, ensuring that the subsequent classification task is executed on a refined feature set.

In this integrated framework, the tokenized input sequence, enhanced by GPT-2's linguistic understanding and structured by LSTM's sequential modeling, is then channeled through a CNN architecture. The synergistic application of these neural components facilitates the extraction of high-level abstract features, thereby encapsulating the intrinsic semantics of agile stories for story point estimation.

This comprehensive approach amalgamates GPT-2's natural language prowess, LSTM's temporal modeling capabilities, and CNN's hierarchical feature extraction prowess. Such integration

augments the system's ability to discern intricate relationships within agile story narratives, ultimately enhancing the precision and robustness of story point estimation. As empirical evidence substantiates the efficacy of neural network ensembles, this framework presents a promising avenue for advancing agile software development practices through sophisticated estimation methodologies.

3.3.4 RoBERTa(Tokenizer)- RoBERTa(Classifier)

In the context of software development using agile methodology, story-point estimation plays a pivotal role in assessing the complexity and effort required to complete a particular user story. Accurate estimation of story points aids project planning and resource allocation. Recent advancements in Natural Language Processing (NLP) techniques, such as **RoBERTa** (A Robustly Optimized BERT Pretraining Approach), have shown promise in automating this process through the analysis of textual data associated with user stories, such as issue descriptions and corresponding identifiers.

The application of RoBERTa for story point estimation follows a systematic process that involves tokenization, model training, and prediction. Tokenization, the initial step, converts textual inputs into a sequence of subword tokens, thereby enabling the model to process them effectively. The tokens encompass the issue descriptions and issue identifiers. Each token is represented by a numerical embedding, which captures its semantic meaning in a continuous vector space.

Subsequently, the tokenized data is employed to train the RoBERTa model. During this phase, the model learns to comprehend the intricate contextual relationships within the text. It captures the semantic nuances associated with issue descriptions and the underlying complexities of the tasks denoted by the respective story points. The model adjusts its internal parameters through an iterative optimization process, minimizing a predefined loss function that quantifies the disparity between predicted and actual story point values.

After training, the model moves to the prediction stage, wherein it applies its acquired knowledge to make estimations on unseen data, known as the test dataset. The RoBERTa model encodes the textual information of issue descriptions and identifiers into meaningful representations. These representations are then used as input to a subsequent layer that

predicts the corresponding story point values. The model's predictions are evaluated against the actual story point labels present in the test dataset, and performance metrics such as Mean Absolute Error (MAE) computed to gauge the accuracy of the predictions.

3.5 Experimental Analysis of Proposed Framework

3.5.1 Implement Details

Tensorflow is used to implement the suggested GPT2-CNN along with an NVIDIA NVIDIA-SMI 460.32.03 GPU and CUDA Version 11.2 without pre-trained networks. The Adam [W] optimizer is used, processing 80 epochs with a batch size of 32, with a preset learning rate of $1e^{-4}$. The default setting was set to use Transformer depth 1 as concluded with multiple experiments that incurring the depth of the transformers did not increase any artifacts of the suggested pipeline.

3.5.2 Hyper parameters Details

Learning Rate and Weight Decay

The optimization process of the CNN model employs an adaptive learning rate algorithm known as AdamW. The learning rate (α) is set to 0.0001, controlling the step size during gradient descent. This value influences the speed and stability of convergence. Additionally, weight decay (λ) is also set to 0.0001, introducing a regularization term to the optimization objective. Weight decay discourages large parameter values, aiding in preventing overfitting and promoting generalization.

Batch Size

During both the training and validation phases, the data processing is done in batches. All of batches contains 32 instances. This parameter influences the granularity of weight updates and affects memory consumption. Smaller batch sizes can lead to more frequent updates and might allow the model to escape local minima, while larger batch sizes can lead to more stable updates but require more memory.

Epochs

The training process iterates over the dataset multiple times, with 80 epochs defined in this code. An epoch represents a complete pass through the training data. Training for multiple epochs allows the model to refine its parameters in response to the training data, improving its

performance over time. Nevertheless, the process of determining the optimal number of epochs involves striking a balance between the effectiveness of the model and the computing efficiency it requires.

Filter Size and Number of Filters

The convolutional layer employs 64 filters with a kernel size of 55. These filters slide across the embedded input data, capturing local patterns. The filter size influences the size of the receptive field, determining the spatial scope of patterns that the filters can recognize. The number of filters affects the CNN capacity to capture different features; a higher number can capture more diverse features but may also lead to overfitting if not balanced with other hyperparameters.

Embedding Dimension

The embedding layer maps input tokens to continuous-valued vectors in a learned embedding space. The embedding dimension is set to 128. This hyperparameter determines the dimensionality of the embedding space, influencing the CNN's ability to capture semantic relationships between words in the input sequence. Larger dimensions can capture more intricate relationships but might require more data.

Activation Functions

Rectified Linear Units (ReLU) serve as activation functions within the CNN model. ReLU activations introduce non-linearity which assist the GPT2-CNN Hybrid Learning model to acquire complex patterns. Specifically, they turn 0 for negative values while leaving positive values unchanged, thereby aiding in capturing and propagating relevant features while suppressing noise.

[3.5.3 Loss function](#)

In order to conduct an objective assessment of the model's performance, the mean absolute error (MAE) loss function used during training is utilised. The Mean Absolute Error (MAE) calculates the average absolute discrepancy between the expected and actual labels of story points. The loss metric presented above offers a straightforward assessment of the model's precision in forecasting story points, irrespective of the errors' direction.

Tokenizer and Padding

The GPT-2 tokenizer is utilized to preprocess the text data. It converts input text into token IDs suitable for model input. The maximum sequence length for tokenized input is set to 128, with padding applied to sequences that are shorter.

3.6 Ablation Study – Experimental Analysis

The pre-trained GPT2-CNN model with Adam(W) optimizer performs well in all of the other experimentation. A detailed diagram of whole architecture is presented to make proper understanding of whole model. The issues text and story points were taken as core input parameters. This textual input then given to GPT2 for tokenization and creation of input ids for CNN. The CNN is then utilized to comprehend those tokens into classified story points notated as predicted story points against each story. The evaluation comparison is explained in the next chapter.

3.7 Summary

This chapter courses proposed model GTP2-CNN Hybrid Learning model, which is based on combination of two models put together. The research also shed light on the combination of diverse models to validate our model performance. Since such combinations are not tried before, with this study future researchers can benefit from it.

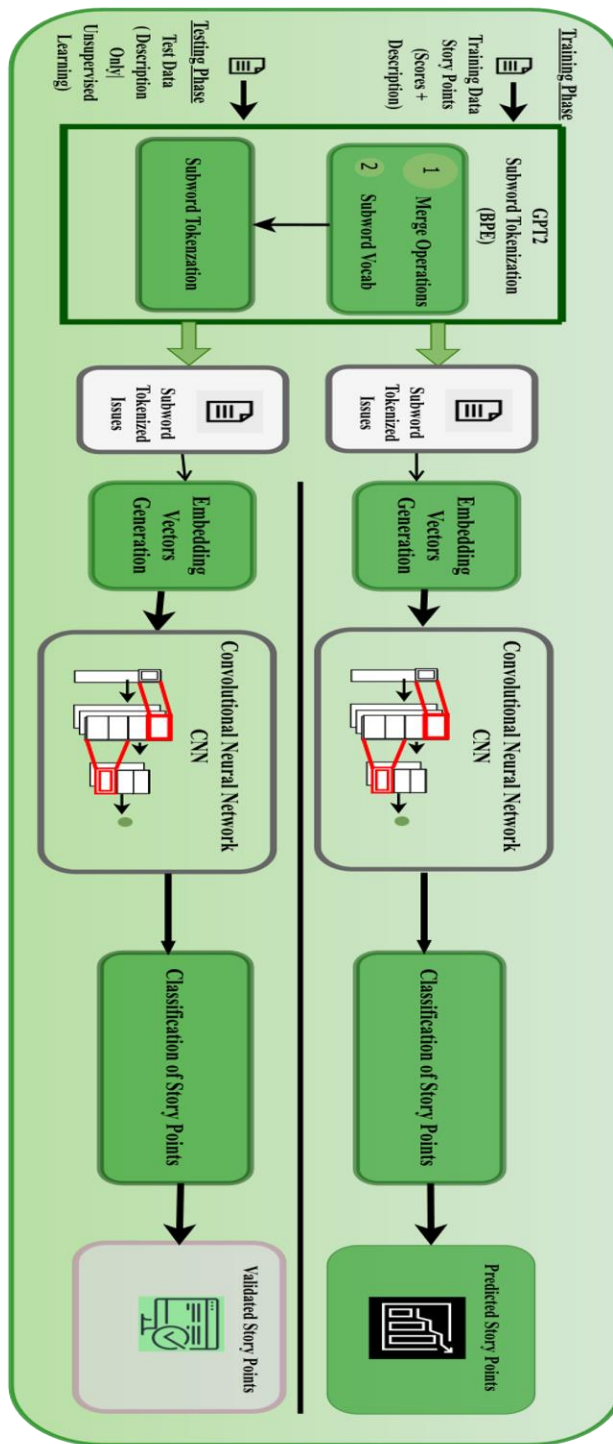


Figure3.3 A detailed Architecture Diagram, explaining the working of GPT2-CNN Hybrid Learning Model for estimation of agile user stories in projects

Chapter 4

Results and Discussion

4.1 Overview

The results segment presents the outcomes of our exploration into the application of a deep learning model for predicting story points in the context of software development. This endeavor involved the development of a sophisticated model that leverages tokenization, convolutional neural networks (CNNs), and embedding layers to tackle the challenge of estimating story points associated with software development tasks. In this section, we offer a comprehensive analysis of the model's performance across a diverse range of software development dataset. The dataset was meticulously selected to encompass various software projects, each characterized by unique attributes and complexities. Our investigation sought to evaluate the model's efficacy in providing accurate estimations of story points, thereby aiding software development teams in enhancing their project planning and resource allocation strategies.

The foundation of our approach lies in the amalgamation of cutting-edge techniques, including state-of-the-art tokenization with GPT-2 and the integration of CNNs for local feature extraction. The architectural configuration of the model is designed to extract meaningful patterns from textual descriptions of software development tasks, enabling the translation of these patterns into reliable estimates of story points. Additionally, we employed embedding layers to generate vector representations for the text, which further enriched the model's ability to grasp semantic nuances and correlations within the data.

Throughout this section, we delve into the specifics of our experimentation, detailing the dataset preparation, model architecture, training process, and subsequent evaluation. Our primary objective is to not only present quantitative metrics of the model's performance but also to provide a qualitative assessment of its effectiveness in real-world software development scenarios. The ensuing subsections elucidate the datasets under scrutiny, lay out the architecture of the deep learning model, delineate the training and evaluation procedures, and culminate in an in-depth analysis of the performance achieved across the diverse software projects. A comprehensive experiment is conducted in order to attain favorable outcomes. A comprehensive examination of existing scholarly works is necessary to establish the foundation for the experimental setup. In order to substantiate our conclusions, we thoroughly examined a variety of research papers and scientific articles. The gpt2-cnn hybrid model demonstrated superior performance compared to the other models in the experimental study. The field of software engineering has seen extensive research in the past and present. Our work specifically

focuses on the development and evaluation of story point estimation techniques within a project. The subsequent sections will provide a comprehensive explanation of the outcomes attained through various hybrid models.

4.2 Regression Evaluation Metrics

Regression evaluation metrics are used to evaluate the performance of regression models, which are used for predicting continuous numerical values. In regression, the goal is to minimize the discrepancy between the predicted values and the actual target values. MAE and MedAE are two ways to quantify this discrepancy and measure how well the model's predictions align with the true values. The implemented scenario has regression task, the evaluation of predicted story points involves the assessment of two models at a time. The evaluation carried out on the MAE taken from each epoch and then performing the MedAE.

The authors of DEEP-SE used the metrics of standard accuracy, media absolute error and mean absolute error. Also, it chooses the three (3) baseline benchmark evaluation metrics of Random guessing, median methods and mean. Our evaluation metric is implemented on the basis of considering the type of input which is human understandable text: story points and the predictions are made in the form of numeric numbers. We select the MAE and MedAE as prior metrics.

4.2.1 Mean Absolute Error

We carry out the assessment using Mean Absolute Error (MAE) to measure the accuracy of a predictive model's predictions. It's particularly useful when dealing with continuous numerical data. MAE measures the average absolute difference between the predicted values and the actual target values. Mathematically, for a set of story point predictions y_i and corresponding actual story point values taken from dataset as x_i .

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - x_i| \quad (6)$$

Where x_i = actual story points from dataset,

y_i = predicted story points by the GPT2-CNN Hybrid model

The MAE metrics is considered an unbiased metrics to evaluate the model performance. MAE gives you an idea of how far off, on average, the predictions are from the actual values. It provides a measure of the model's accuracy without considering the direction of the errors. The higher the value is poor the performance of model. The lower the value, good performance of model is achieved.

4.2.2 Median Absolute Error Evaluation

Median Absolute Error is another metric used to evaluate the performance of predictive models, especially when dealing with outliers or skewed data. The dataset we utilized has numerous numbers of lines containing text as explained earlier. MedAE, unlike MAE, considers the median of the absolute differences between predicted story points and actual story points. The median is less sensitive to outliers compared to the mean, making MedAE more robust in the presence of extreme values.

Mathematically, for a set of story point predictions y_i and corresponding actual story points x_i , the MedAE is calculated as:

$$MedAE = median(|y_1 - x_1|, |y_2 - x_2|, |y_3 - x_3|, \dots, |y_n - x_n|) \quad (7)$$

$n = \text{thenumberofdatapoints}$, in our case the number of data points are 16.

(median)functionisusedcalculatethemedianoftheabsolutedatapoints.

MedAE provides a measure of central tendency of the errors, which can be a more reliable indicator of model performance when dealing with skewed or noisy data. The data we curtailed has some missing values and repeated values which makes it a noisy data, it justifies our choice of evaluation metric.

4.3 Experimental Analysis of Proposed Hybrid Models

On dataset of story points, our experiments show variety of results. To give a small recap to what have been shown earlier, this section will demonstrate the results of all the models. The first model is gpt2-cnn version 2.0, our foremost important model whose performance out performs the baseline paper and out other experiments too. The gpt2 pretrained model is used here for the purpose of word vector embeddings. These embeddings are then fed to CNN which classifies the vectors into story point predictions. The model uses the learning rate (lr) of 0.0001, with a decay of 0.0001 and used AdamW, as optimizer. The experiment has been in state of running till 80 epochs. We use the early stopping technique to stop the training of model at a point where the results are undesirable. The following table shows the results of gpt2-cnn with AdamW as optimizer.

Table 4.1 Results of GPT2-CNN (Adam W optimizer) Hybrid Learning Model

S.No	Repository	Project Name	MAE Epoch 10	MAE Epoch 20	MAE Epoch 30	MAE Epoch 50	MAE Epoch 80	Median
1	Appcelerator	Appcelerator Studio	1.39	1.45	1.59	1.61	2.05	1.39
2	Appcelerator	Aptana Studio	5.41	3.53	3.54	3.53	3.54	3.53
3	Atlassian	Bamboo	0.94	0.79	0.76	0.76	0.86	0.76
4	Atlassian	Clover	3.97	3.72	3.74	3.62	3.65	3.62
5	Lsstcorp	Data Management	6.18	5.97	5.99	6.52	7.15	5.97
6	Dura Space	DuraCloud	0.80	0.80	0.80	0.77	0.78	0.77
7	Atlassian	JIRA Software	3.64	1.91	1.91	1.84	1.77	1.77
8	Apache	Mesos	1.21	1.19	1.13	1.18	1.10	1.10
9	Moodle	Moodle	5.38	6.52	6.75	6.82	7.73	5.38
10	Mulesoft	Mule	3.01	2.55	2.52	2.48	2.55	2.48

11	Mulesoft	Mule Studio	5.94	3.83	3.80	3.78	3.84	3.78
12	Spring	Spring XD	1.68	1.68	1.74	1.87	1.81	1.68
13	Talendforge	Talend Data Quality	3.36	3.87	3.88	3.47	3.79	3.36
14	Talendforge	Talend ESB	0.90	0.84	0.82	0.84	0.86	0.82
15	Appcelerator	Titanium SDK/CLI	2.16	2.20	2.23	2.44	2.87	2.16
16	Apache	Usergrid	1.94	1.47	1.11	1.17	1.17	1.11
MedAE = 1.96								

The **table 4.1** contains extensive trials data of each epoch till 80, where gpt2 used as tokenizer and cnn used as classifier to produce the desired outcome. Best values of MAE from each epoch used to calculate the median absolute error, which represents the significance of our research.

4.3.1 Graphs

The following graphs of our model are also shown for better understanding the merits and demerits of our model.

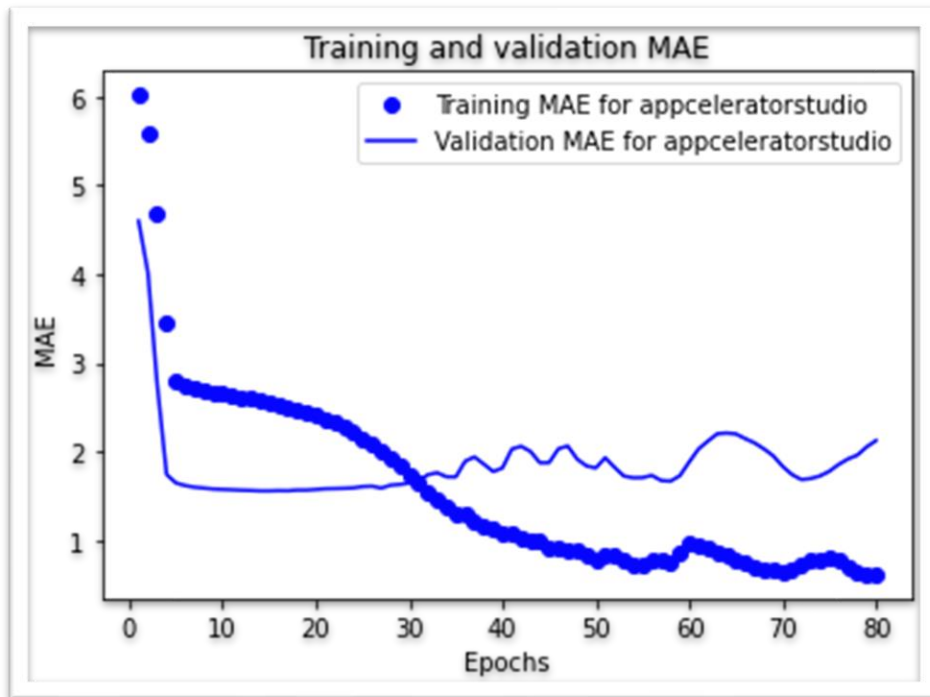


Figure 4.1 Graph of Trained and Validated Appcelerator Studio

The appcelerator studio **figure 4.1** shows the training process is having a straight line convergence that generalize the results after 80th epoch. The validation is quiet good as it replicates the behavior of training.

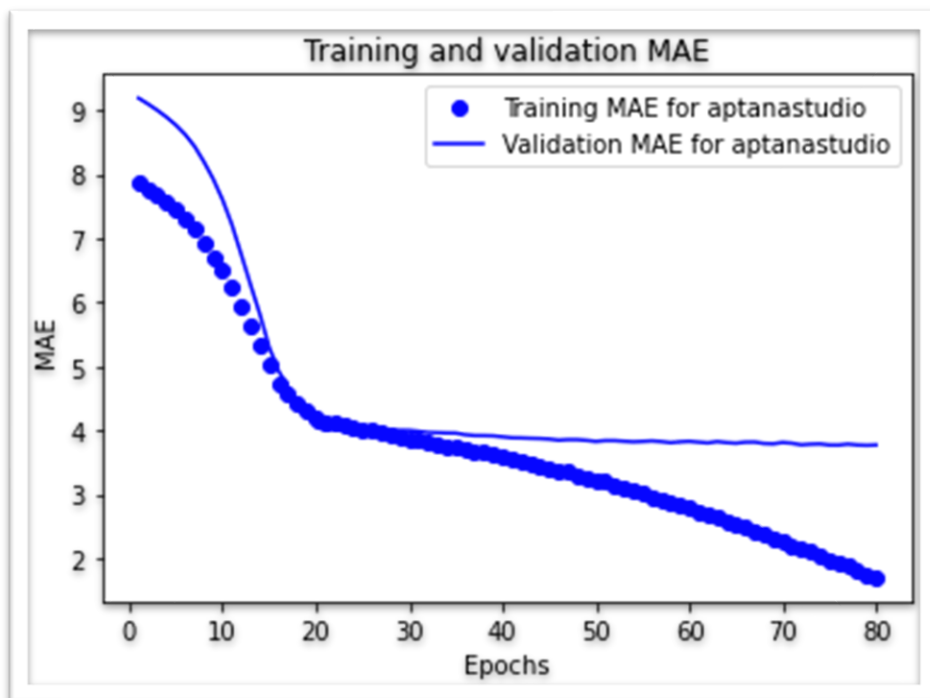


Figure 4.2 Training and Validation of Aptana Studio

Aptana Studio file **figure 4.2** performs well on GPT2-CNN hybrid learning model. All project files have some different size of stories contained in data files. This variance of data has allowed our GPT2-CNN hybrid learning model to train and validate on different data which changes its performance on each data file. Like wise **figure 4.3** for bamboo studio, **figure 4.4** for clover, **figure 4.5** for data management, **figure 4.6** for dura cloud, **figure 4.7** for jira software, **figure 4.8** for mesos, **figure 4.9** for moodle, **figure 4.10** for mule, **figure 4.11** mule studio, **figure 4.12** spring XD, **figure 4.13** for talend dataquality, **figure 4.14** for talend ESB, **figure 4.15** for titanium sdk/cli, **figure 4.16** for usergrid respectively.

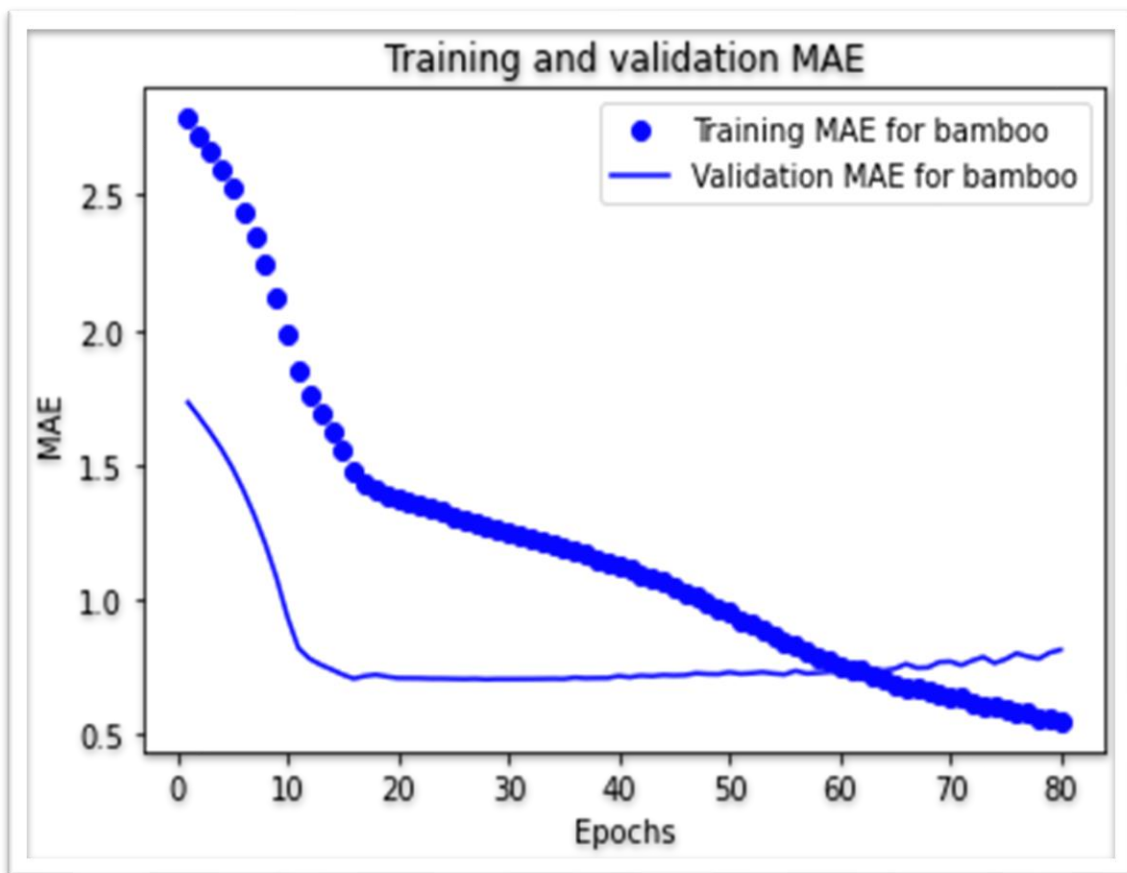


Figure 4.3 Bamboo Studio graph explaining the training and validation of dataset

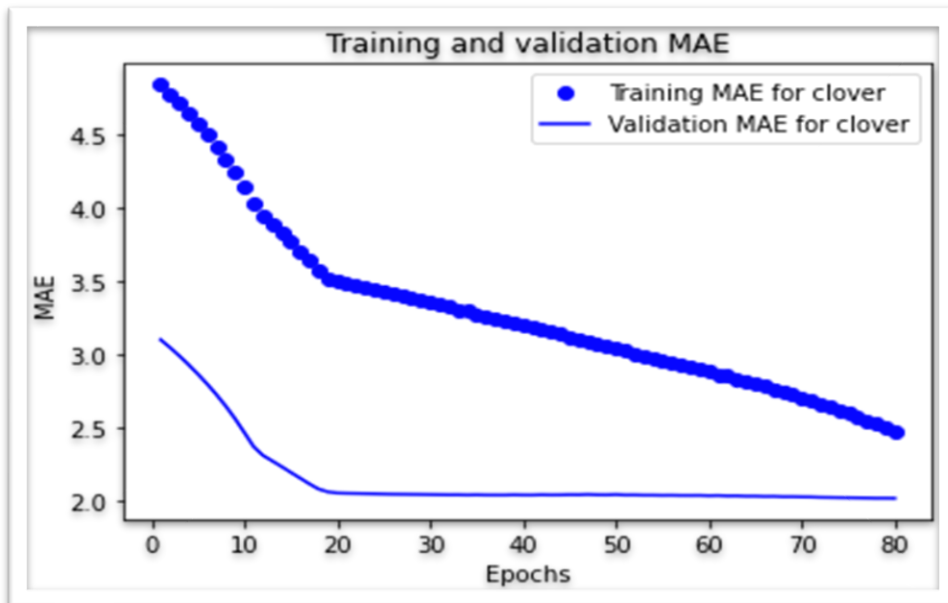


Figure 4.4 Clover data file training and validation MAE

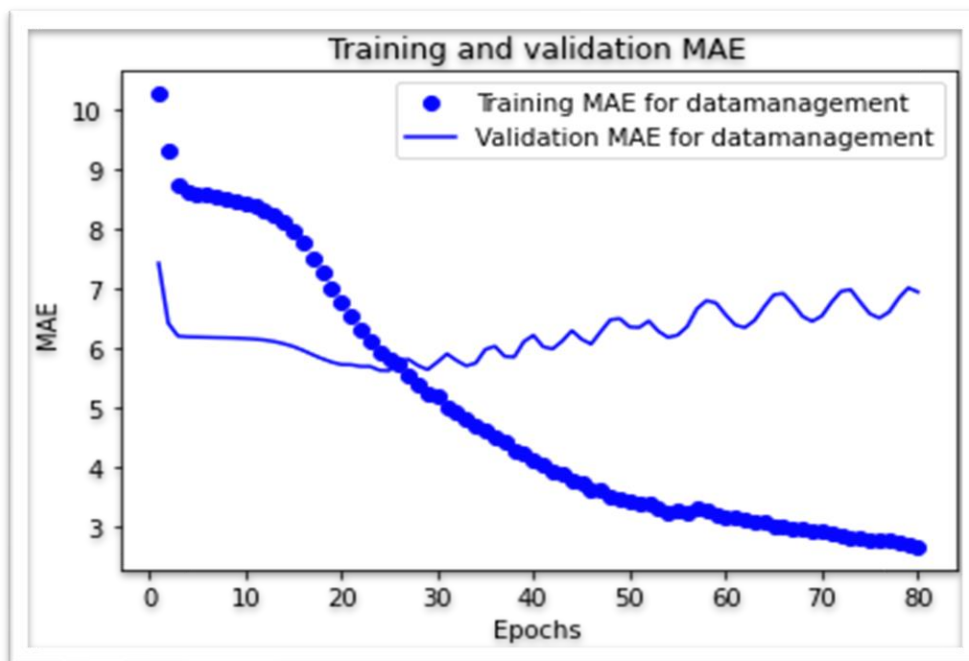


Figure 4.5 Data Management File Training and Validation MAE

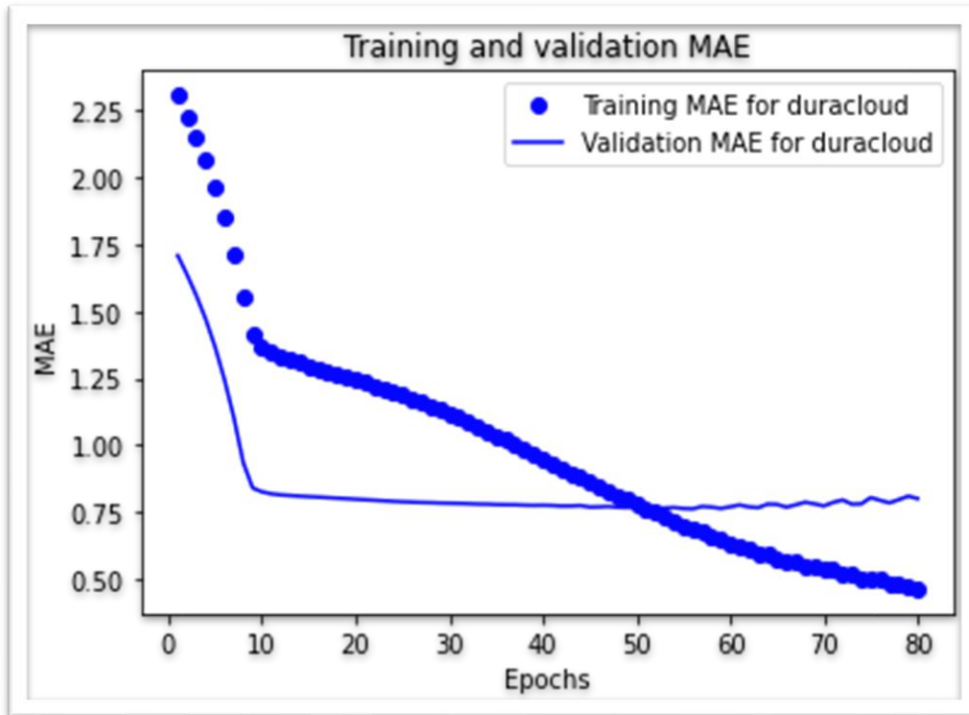


Figure 4.6 Dura Cloud Training and Validation MAE

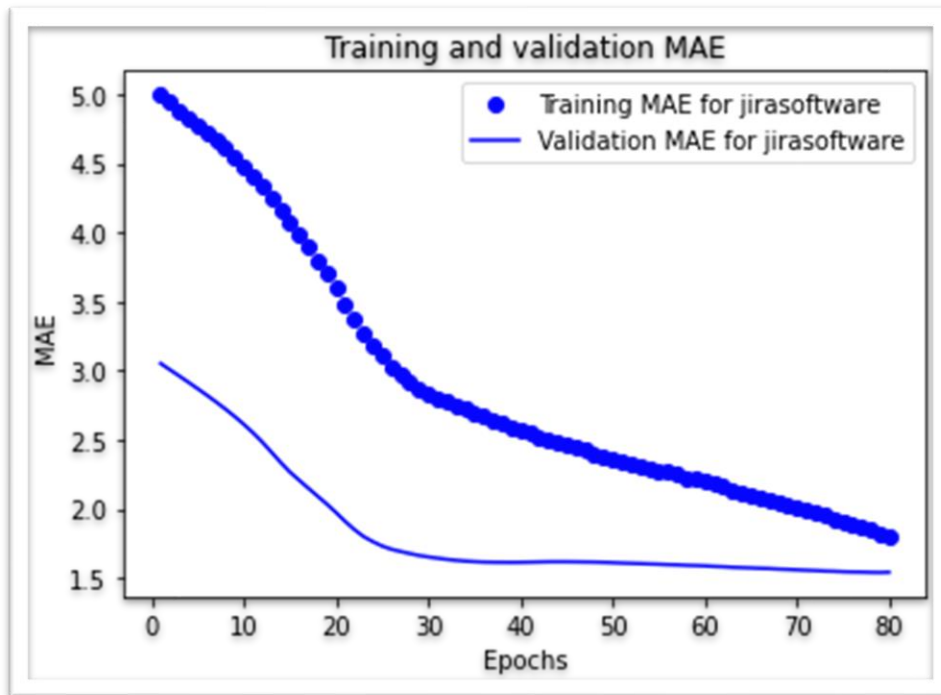


Figure 4.7 Jira Software Training and Validation MAE

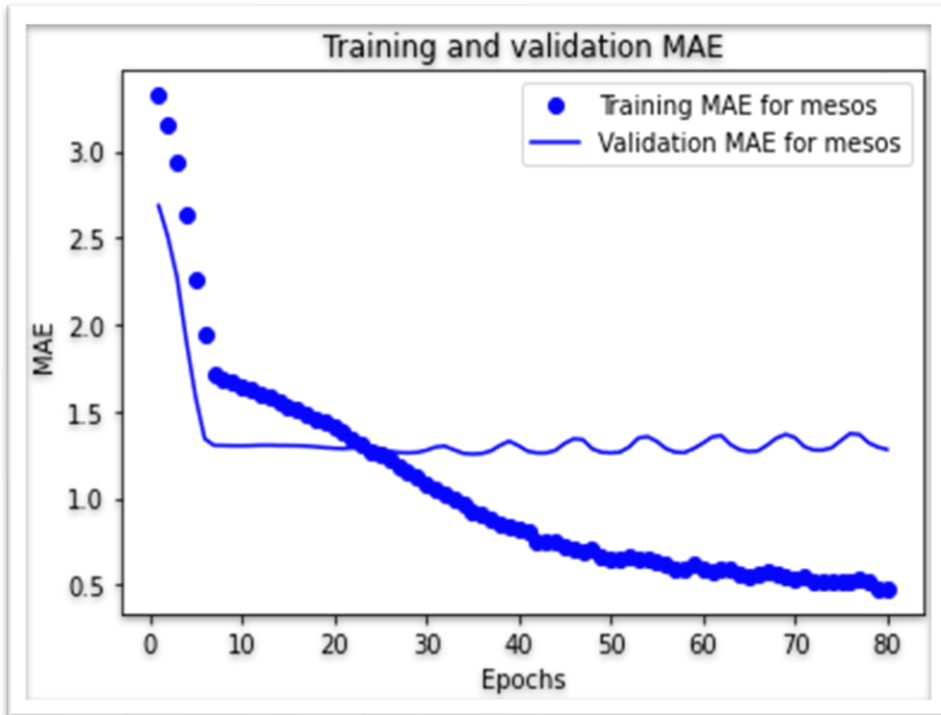


Figure 4.8 Mesos Training and Validation MAE

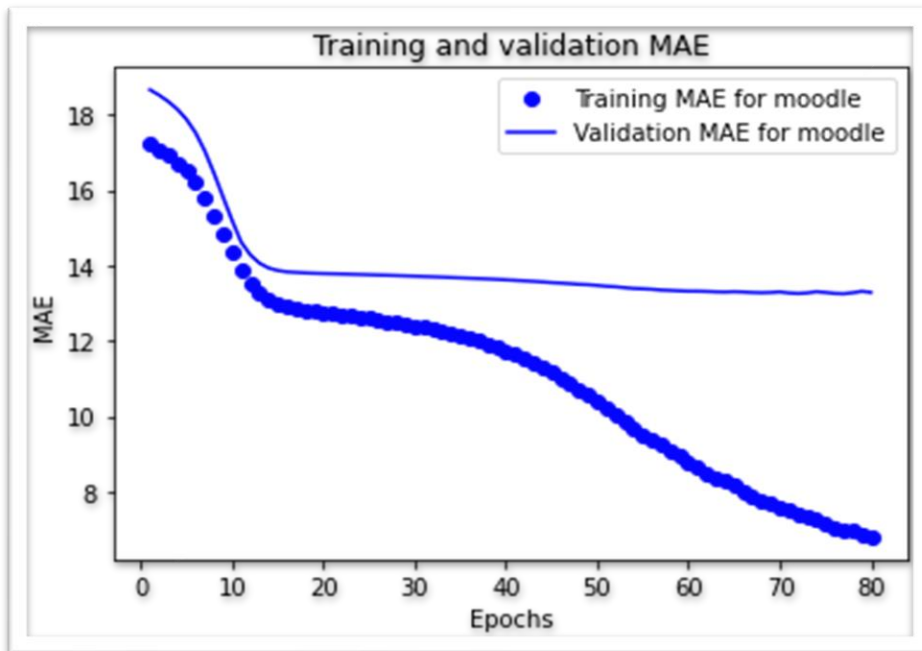


Figure 4.9 Moodle Training and Validation MAE

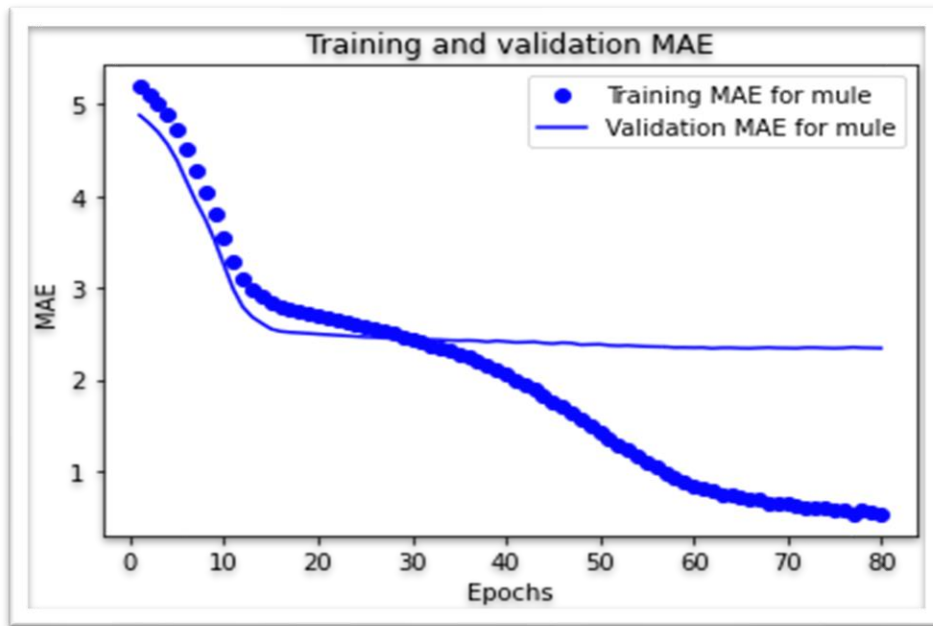


Figure 3.10 Mule Training and Validation MAE

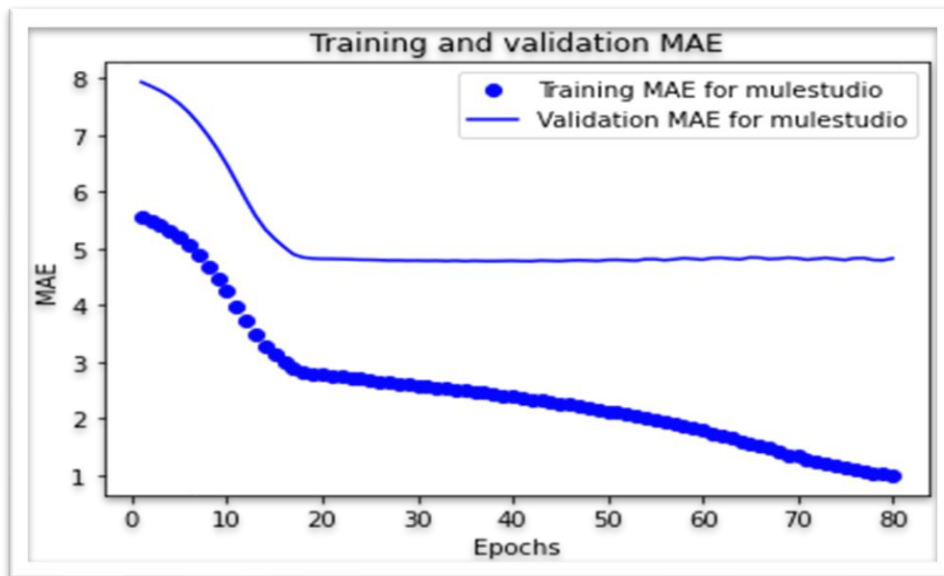


Figure 4.11 Mule Studio Training and Validation MAE

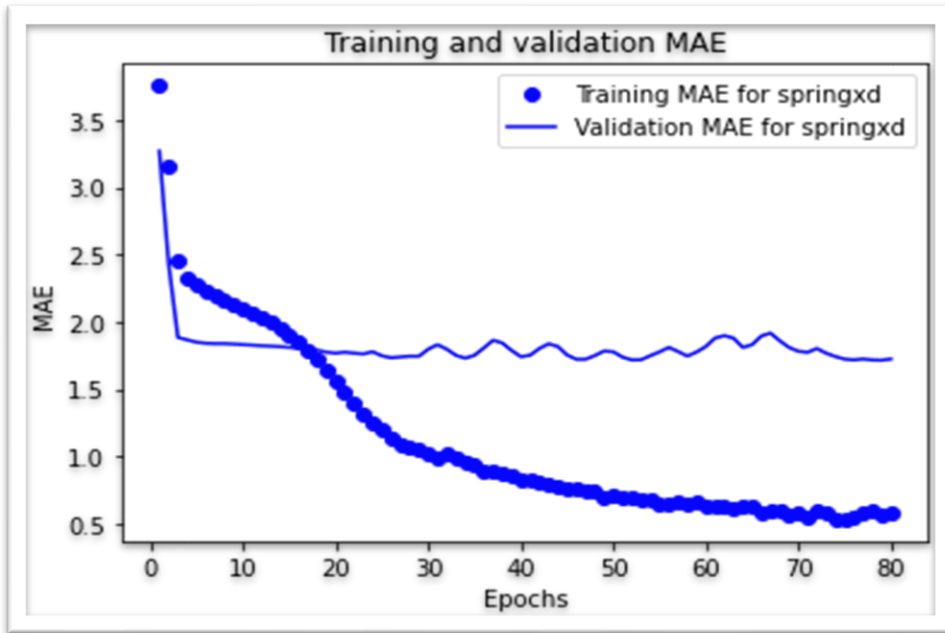


Figure 4.12 Springxd Training and validation MAE

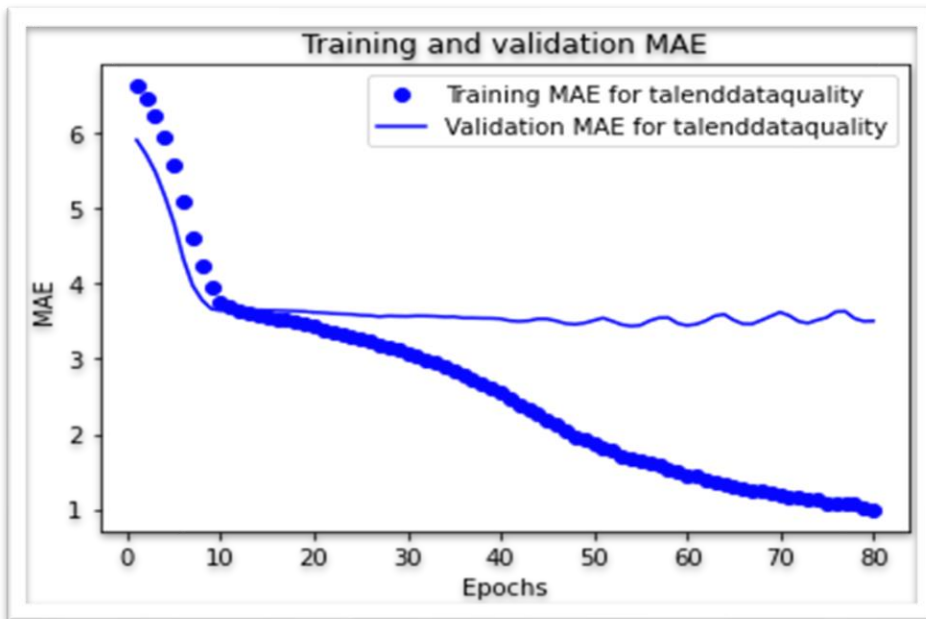


Figure 4.13 Talend Data Quality Training and Validation MAE

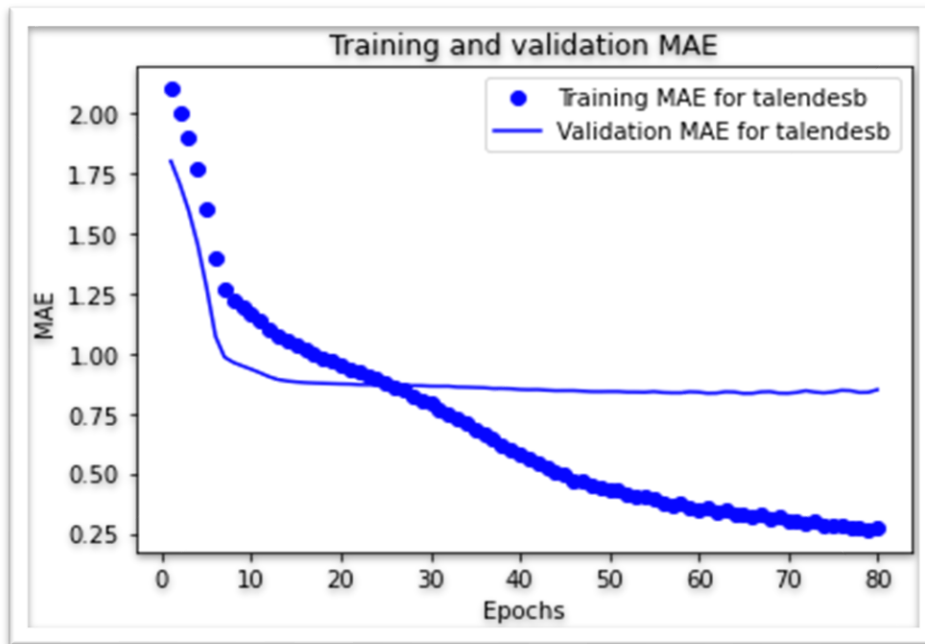


Figure 4.14 Talendesb Training and Validation MAE

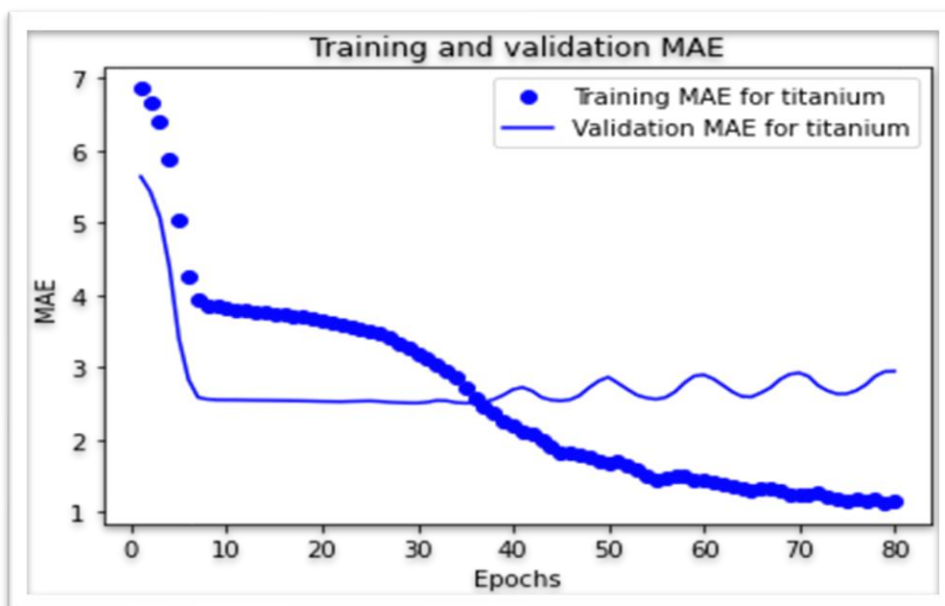


Figure 4.15 Titanium Training and Validation MAE

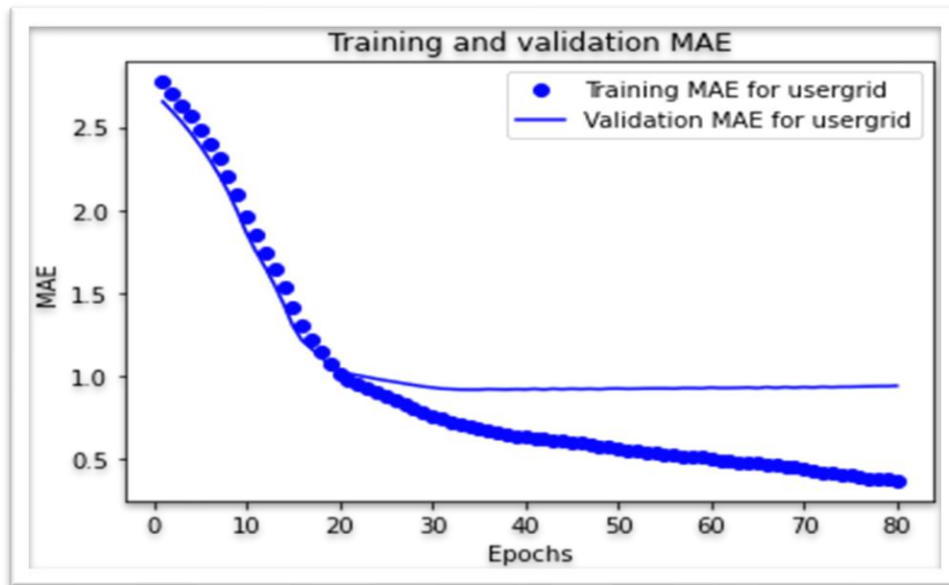


Figure 4.16 Usergrid Training and Validation MAE

4.3.2 Heat Map

The heatmap is a visual representation of data under study. The heatmap represents the variance between the projects on x-axis and projects on y-axis. The use of heatmap allows us to visually understand the difference between data files. We observe that the clean data and unclean data has some impact on the performance of GPT2-CNN hybrid learning model. See **Figure 4.17**

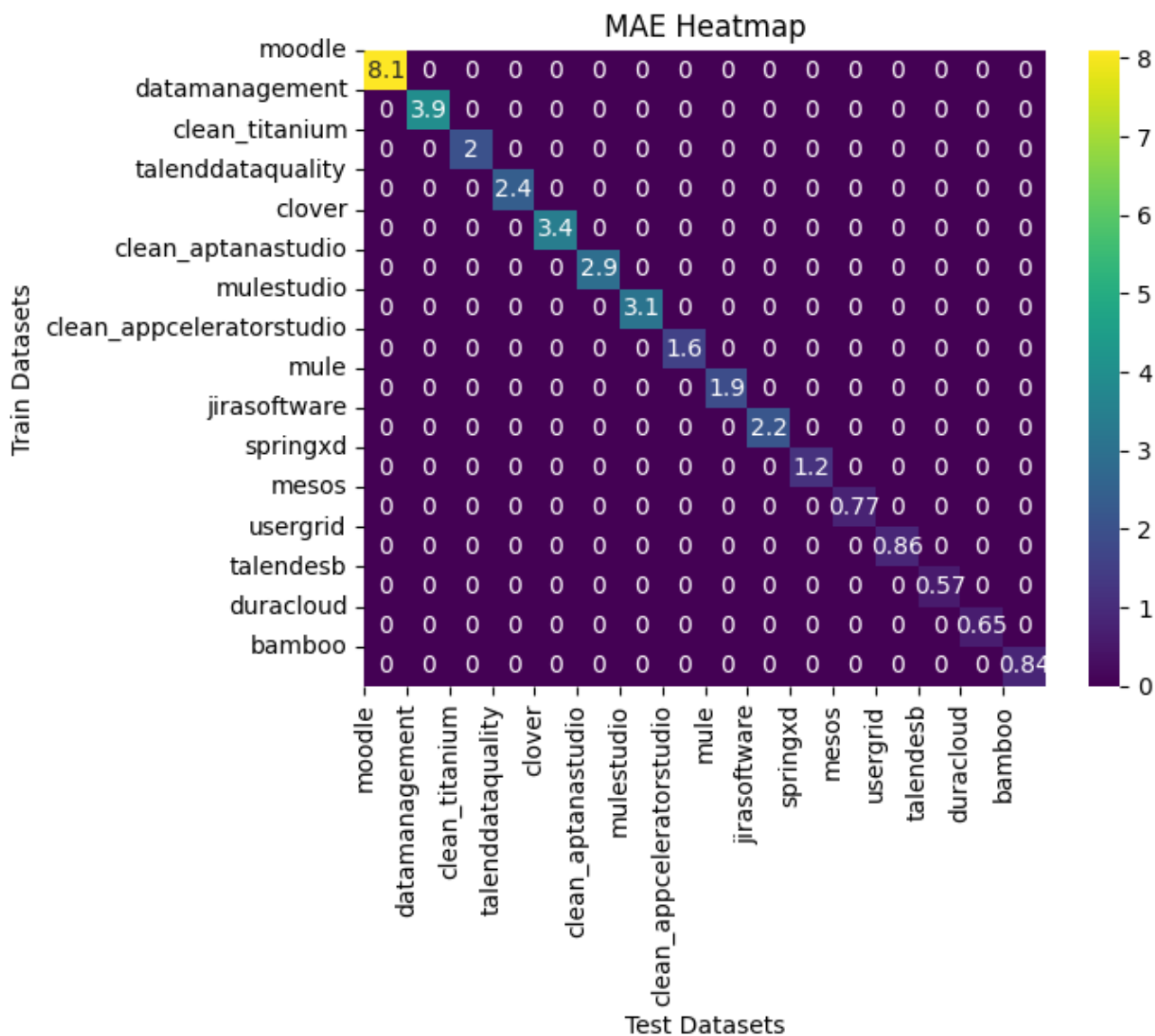


Figure 4.17 Heatmap of Training and Validation MAE of all project files

4.4 Evaluation on GPT2-CNN

The evaluation carried out on gpt2-cnn hybrid model, we use the pre-defined hyperparameters. The results achieved are equivalent to those of base line paper. The difference is about computational time it took. As mentioned in the baseline paper, it requires several hours to train the model but our model only require 1.5 hours to train on this enormous dataset of story points. The numbers of epochs are less as compared to the previous one because we use early stopping and learned that further training will overfit the model for this particular task. The following **table 4.2** shows the results.

Table 4.2 GPT2 as tokenizer -CNN as classifier where CNN is not used with any optimizer

S.No	Repository	Project Name	MAE Epoch 10	MAE Epoch 20	MAE Epoch 30	Median
1	Appcelerator	Appcelerator Studio	1.41	1.45	1.78	1.41
2	Appcelerator	Aptana Studio	5.41	3.65	3.55	3.55
3	Atlassian	Bamboo	0.88	0.80	0.76	0.76
4	Atlassian	Clover	4.96	3.71	3.77	3.71
5	Lsstcorp	Data Management	6.18	5.98	6.01	5.98
6	Dura Space	DuraCloud	0.82	0.81	0.76	0.76
7	Atlassian	JIRA Software	3.11	2.67	1.94	1.94
8	Apache	Mesos	1.24	1.20	1.14	1.14
9	Moodle	Moodle	5.41	6.52	6.75	5.41
10	Mulesoft	Mule	3.02	2.57	2.54	2.54
11	Mulesoft	Mule Studio	5.94	3.81	3.77	3.77
12	Spring	Spring XD	1.68	1.72	1.89	1.68
13	Talendforge	Talend Data Quality	3.36	3.76	3.99	3.36
14	Talendforge	Talend ESB	0.98	0.84	0.84	0.84
15	Appcelerator	Titanium SDK/CLI	2.15	2.19	2.34	2.15
16	Apache	Usergrid	1.95	1.22	1.15	1.15
MedAE = 2.04						

4.5 Evaluation on GPT2-LSTM

Evaluation of gpt2-LSTM is carried out to testify our own model presented with multiple variations. The variation of gpt2-LSTM, we use the default parameters. The gpt2 pretrained model is used as tokenizer and LSTM is used as classifier and predictor. The notion of using this variation is to learn

the difference of performance on a language task. The results showed in following **table 4.3** assist the understanding of LSTM on a task like story point estimation.

Table 2.3 GPT2- LSTM where gpt2 as tokenizer and LSTM as classifier

S.No	Repository	Project Name	MAE Epoch 10	MAE Epoch 20	MAE Epoch 30	Median
1	Appcelerator	Appcelerator Studio	1.39	1.30	2.159	1.30
2	Appcelerator	Aptana Studio	3.61	3.61	3.61	3.61
3	Atlassian	Bamboo	0.77	0.75	0.77	0.75
4	Atlassian	Clover	3.75	3.65	3.52	3.52
5	Lsstcorp	Data Management	6.18	6.21	6.07	6.07
6	Dura Space	DuraCloud	0.84	0.72	0.70	0.70
7	Atlassian	JIRA Software	2.24	2.24	2.18	2.18
8	Apache	Mesos	1.26	1.50	1.27	1.27
9	Moodle	Moodle	6.48	6.48	6.48	6.48
10	Mulesoft	Mule	2.46	2.46	2.50	2.46
11	Mulesoft	Mule Studio	3.58	3.58	3.58	3.58
12	Spring	Spring XD	1.73	1.80	2.37	1.73
13	Talendforge	Talend Data Quality	3.29	3.29	3.47	3.29
14	Talendforge	Talend ESB	0.93	0.87	0.84	0.84
15	Appcelerator	Titanium SDK/CLI	2.42	2.14	2.89	2.14
16	Apache	Usergrid	1.16	1.16	1.16	1.16
MedAE = 2.16						

4.6 Evaluation of GTP-2 – LSTM – CNN

GPT-2 – LSTM-CNN model is a pure hybrid model, we implement this model to understand the working of tokenizer, a separate embedding vector mechanism and a classifier of CNN. The architecture aims to capture both sequential and local patterns in the data. The AdamW optimizer with weight decay helps in optimizing the model's parameters during training. The results are shown in **table 4.4**

Table 4.4 GPT2-LSTM-CNN a tri model variation evaluation

S.No	Repository	Project Name	MAE Epoch 10 lr=0.0001	MAE Epoch 20- lr=0.0001	Median
1	Appcelerator	Appcelerator Studio	1.30	1.30	1.30
2	Appcelerator	Aptana Studio	3.61	3.61	3.61
3	Atlassian	Bamboo	0.76	0.75	0.75
4	Atlassian	Clover	3.77	3.61	3.61
5	Lsstcorp	Data Management	6.33	6.88	6.33
6	Dura Space	DuraCloud	0.80	0.75	0.75
7	Atlassian	JIRA Software	2.23	2.09	2.09
8	Apache	Mesos	1.148	1.23	1.23
9	Moodle	Moodle	6.48	6.48	6.48
10	Mulesoft	Mule	2.46	2.51	2.46
11	Mulesoft	Mule Studio	3.57	3.58	3.57
12	Spring	Spring XD	1.71	1.72	1.71
13	Talendforge	Talend Data Quality	3.29	3.29	3.29
14	Talendforge	Talend ESB	0.86	0.78	0.78
15	Appcelerator	Titanium SDK/CLI	2.03	2.13	2.03
16	Apache	Usergrid	1.16	1.16	1.16
MedAE = 2.06					

4.7 Evaluation of Roberta-Roberta-Adam

Roberta- Roberta- Adam is blend of two models, one model implied to work as tokenizer and the other model is implied to predict story points on the basis on previous model output. The optimizer used here is ADAM optimizer, other hype parameters settings are set as default settings like other models under experiment. See **table 4.5**

Table 4.5 Roberta as tokenizer and Roberta as classifier with Adam optimizer

S.No	Repository	Project Name	Epoch 10	Epoch 20	Best
1	Appcelerator	Appcelerator Studio	1.75	2.21	1.75
2	Appcelerator	Aptana Studio	3.35	3.44	3.35
3	Atlassian	Bamboo	0.88	0.82	0.82
4	Atlassian	Clover	3.58	3.84	3.58
5	Lsstcorp	Data Management	6.35	6.64	6.35
6	Dura Space	DuraCloud	0.81	0.81	0.81
7	Atlassian	JIRA Software	1.91	1.58	1.58
8	Apache	Mesos	1.21	1.21	1.21
9	Moodle	Moodle	6.11	6.48	6.11
10	Mulesoft	Mule	2.62	2.54	2.54
11	Mulesoft	Mule Studio	4.46	4.46	4.46

12	Spring	Spring XD	1.77	1.77	1.77
13	Talendforge	Talend Data Quality	2.81	2.81	2.81
14	Talendforge	Talend ESB	1.04	1.04	1.04
15	Appcelerator	Titanium SDK/CLI	2.64	2.64	2.64
16	Apache	Usergrid	1.91	1.91	1.91
MedAE = 2.22					

4.8 Comparative Analysis of Various Methods

To demonstrate our performance superiority, we compare the GPT2-CNN Model with 5 approaches. It contains comparison of different models. The top results are bold. This analysis includes regression evaluation techniques. See **table 4.6**

Table 4.6 Comparison between all models and state-of-the-art model

S.No	Repository	Project Name	Deep-SE	GPT2-CNN(Adam)	GPT2-CNN	GPT2-LSTM	GPT2-LSTM_CNN	Roberta-Roberta
1	Appcelerator	Appcelerator Studio	5	1.39	1.41	1.30	1.30	1.75
2	Appcelerator	Aptana Studio	8	3.53	3.55	3.61	3.61	3.35
3	Atlassian	Bamboo	2	0.76	0.76	0.75	0.75	0.82
4	Atlassian	Clover	2	3.62	3.71	3.52	3.61	3.58
5	Lsstcorp	Data Management	4	5.97	5.98	6.07	6.33	6.35
6	Dura Space	DuraCloud	1	0.77	0.76	0.70	0.75	0.81
7	Atlassian	JIRA Software	3	1.77	1.94	2.18	2.09	1.58

8	Apache	Mesos	3	1.10	1.14	1.27	1.23	1.21
9	Moodle	Moodle	8	5.38	5.41	6.48	6.48	6.11
10	Mulesoft	Mule	5	2.48	2.54	2.46	2.46	2.54
11	Mulesoft	Mule Studio	5	3.78	3.77	3.58	3.57	4.46
12	Spring	Spring XD	3	1.68	1.68	1.73	1.71	1.77
13	Talendforge	Talend Data Quality	5	3.36	3.36	3.29	3.29	2.81
14	Talendforge	Talend ESB	2	0.82	0.84	0.84	0.78	1.04
15	Appcelerator	Titanium SDK/CLI	5	2.16	2.15	2.14	2.03	2.64
16	Apache	Usergrid	3	1.11	1.15	1.16	1.16	1.91
	MedAE		2.33	1.96	2.04	2.16	2.06	2.22

4.9 Summary

The experimentation carried out lead our research to show good results as compared to other approaches especially DEEP-SE. The median mean absolute error MedAE of DEEP-SE is 2.33 and our gpt2-CNN model achieves MedAE of **1.96** which shows that our model outperforms the previous model. Our research comprises of executing different machine learning model, and then learning the best model which has given good results as solution to problem of story point estimation. The experimentation involved the use of building five(5) models and comparing their performance on the story point dataset.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

This chapter, briefly summarizes the efforts, limitations, and recommendations for future studies. After explaining the conceptual approach, conducting experiments, and reviewing the findings the final observations and expositions are discussed. The goal of the project is to create a hybrid model that performs better than cutting-edge approaches. In previous studies deep learning models are utilized for estimation of agile projects using story point but no other study found with a novel hybrid model comprising of GPT2 – CNN; the transformer is used to perform word embeddings and neural network is used for the classification and predictions of story points. In this paper, we introduce GPT2-CNN, a unique and portable deep learning model for improving story points predictions. The suggested techniques provide quick inference with less memory consumption. To assess GPT2-CNN settings, we undertook numerous tests. The proposed GPT2-CNN is shown to be efficient and effective when compared to previous state-of-the-art work through quantitative and qualitative results.

5.2 Limitations

By learning many samples, the methodology based on deep neural algorithms can lessen the effect of the complicated story point estimation in agile story point estimation. However, the dataset is crucial, as the existing dataset's coverage is still constrained. Deep learning-based techniques, put more emphasis on improving full integration of the story point estimation model. Therefore, maximizing the features which can bear good generalization performance. Estimation of story point only relies on the user stories data is not sufficient therefore, gathering the data which has more meaningful relationship will help the model to generalize more. The deep learning models have a mechanism that is not interpretable by humans easily so defining a good metric is also limitation to this study.

5.3 Future Work

The deep learning and machine models can be trained using perception-related loss function and introduce factors that are consistent with human interpretation, which will make the network more effective across wider range of scenarios. The interpretability of model results is truly dependent on the observer who will generalize the estimations. If the results are not interpreted well then project will get affected. This area of explainable AI must be addressed for further research. Explaining how model

generalize its behavior in terms of calculations and refinement of hidden states will allow humans to fully visualize the output.

Furthermore, in light of the challenge of effectively handling the reduction of intricate data complexities while simultaneously prioritizing computing efficiency, researchers may enhance the use of the data employed for modeling purposes. Simultaneously, the use of gradual reinforcement learning approaches is becoming recognized as a feasible approach. This strategy framework focuses on enhancing real-time operational efficiency, hence driving progress in research related to the technical field of story point estimation. Domain experts can come forward to help researchers with the exploration of user requirement gathering and enable researchers to find ways to better estimations. Since the data collected is not refined and employing any powerful model will not yield any better results until data is large and robust.

6. References

- [1] Ramchurreetoo, Yuvna, and Visham Hurbungs. "A multiclass classification model to estimate Agile user stories." In *2022 3rd International Conference on Next Generation Computing Applications (NextComp)*, pp. 1-5. IEEE, 2022.
- [2] Tawosi, Vali, Rebecca Moussa, and Federica Sarro. "Agile Effort Estimation: Have We Solved the Problem Yet? Insights From A Replication Study." *IEEE Transactions on Software Engineering* (2022).
- [3] Garg, Yashika. "Comparative Analysis of Machine Learning Techniques in Effort Estimation." In *2022 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COM-IT-CON)*, vol. 1, pp. 401-405. IEEE, 2022.
- [4] Rosa, Wilson, Bradford K. Clark, Raymond Madachy, and Barry W. Boehm. "Empirical effort and schedule estimation models for agile processes in the US DoD." *IEEE Transactions on Software Engineering* 48, no. 8 (2021): 3117-3130.
- [5] Jørgensen, Magne. "A review of studies on expert estimation of software development effort." *Journal of Systems and Software* 70, no. 1-2 (2004): 37-60.
- [6] Hamid, Muhammad, Furkh Zeshan, and Adnan Ahmad. "Fuzzy Logic-based Expert System for Effort Estimation in Scrum Projects." In *2021 International Conference on Decision Aid Sciences and Application (DASA)*, pp. 761-765. IEEE, 2021.
- [7] Tawosi, Vali, Afnan Al-Subaihin, and Federica Sarro. "Investigating the Effectiveness of Clustering for Story Point Estimation." IEEE, 2022.
- [8] Porru, Simone, Alessandro Murgia, Serge Demeyer, Michele Marchesi, and Roberto Tonelli. "Estimating story points from issue reports." In *Proceedings of the The 12th International Conference on Predictive Models and Data Analytics in Software Engineering*, pp. 1-10. 2016.
- [9] Sarro, Federica, Rebecca Moussa, Alessio Petrozziello, and Mark Harman. "Learning from mistakes: Machine learning enhanced human expert effort estimates." *IEEE Transactions on Software Engineering* 48, no. 6 (2020): 1868-1882.
- [10] Gultekin, Muaz, and Oya Kalipsiz. "Story point-based effort estimation model with machine learning techniques." *International Journal of Software Engineering and Knowledge Engineering* 30, no. 01 (2020): 43-66.

- [11] Pendharkar, Parag C., Girish H. Subramanian, and James A. Rodger. "A probabilistic model for predicting software development effort." *IEEE Transactions on software engineering* 31, no. 7 (2005): 615-624.
- [12] Scott, Ezequiel, and Dietmar Pfahl. "Using developers' features to estimate story points." In *Proceedings of the 2018 International Conference on Software and System Process*, pp. 106-110. 2018.
- [13] Ardiansyah, Ardiansyah, Murein Miksa Mardhia, and Sri Handayaningsih. "Analogy-based model for software project effort estimation." *International Journal of Advances in Intelligent Informatics* 4, no. 3 (2018): 251-260.
- [14] Tawosi, Vali, Rebecca Moussa, and Federica Sarro. "On the relationship between story points and development effort in Agile open-source software." In *Proceedings of the 16th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pp. 183-194. 2022.
- [15] Zakrani, Abdelali, Assia Najm, and Abdelaziz Marzak. "Support vector regression based on grid-search method for agile software effort prediction." In *2018 IEEE 5th International Congress on Information Science and Technology (CiSt)*, pp. 1-6. IEEE, 2018.
- [16] Mahmood, Yasir, Nazri Kama, Azri Azmi, Ahmad Salman Khan, and Mazlan Ali. "Software effort estimation accuracy prediction of machine learning techniques: A systematic performance evaluation." *Software: Practice and experience* 52, no. 1 (2022): 39-65.
- [17] Satapathy, Shashank Mouli, and Santanu Kumar Rath. "Empirical assessment of machine learning models for agile software development effort estimation using story points." *Innovations in Systems and Software Engineering* 13, no. 2-3 (2017): 191-200.
- [18] Cao, Lan. "Estimating Efforts for Various Activities in Agile Software Development: An Empirical Study." *IEEE Access* 10 (2022): 83311-83321.
- [19] Yang, Yanming, Xin Xia, David Lo, Tingting Bi, John Grundy, and Xiaohu Yang. "Predictive models in software engineering: Challenges and opportunities." *ACM Transactions on Software Engineering and Methodology (TOSEM)* 31, no. 3 (2022): 1-72.
- [20] Kumar, P. Suresh, Himansu Sekhar Behera, Anisha Kumari, Janmenjoy Nayak, and Bighnaraj Naik. "Advancement from neural networks to deep learning in software effort estimation: Perspective of two decades." *Computer Science Review* 38 (2020): 100288.

- [21] Fu, Michael, and Chakkrit Tantithamthavorn. "Linevul: A transformer-based line-level vulnerability prediction." In *Proceedings of the 19th International Conference on Mining Software Repositories*, pp. 608-620. 2022. [25]
- [22] Zhang, Kui, Xu Wang, Jian Ren, and Chao Liu. "Efficiency improvement of function point-based software size estimation with deep learning model." *IEEE Access* 9 (2020): 107124-107136.
- [23] Phan, Hung, and Ali Jannesari. "Heterogeneous Graph Neural Networks for Software Effort Estimation." In *Proceedings of the 16th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pp. 103-113. 2022.
- [24] Pasuksmit, Jirat, Patanamon Thongtanunam, and Shanika Karunasekera. "Story points changes in agile iterative development: An empirical study and a prediction approach." *Empirical Software Engineering* 27, no. 6 (2022): 156.
- [25] Panda, Aditi, Shashank Mouli Satapathy, and Santanu Kumar Rath. "Empirical validation of neural network models for agile software effort estimation based on story points." *Procedia Computer Science* 57 (2015): 772-781.
- [26] Fu, Michael, and Chakkrit Tantithamthavorn. "GPT2SP: A transformer-based agile story point estimation approach." *IEEE Transactions on Software Engineering* 49, no. 2 (2022): 611-625.
- [27] Choetkiertikul, Morakot, Hoa Khanh Dam, Truyen Tran, Trang Pham, Aditya Ghose, and Tim Menzies. "A deep learning model for estimating story points." *IEEE Transactions on Software Engineering* 45, no. 7 (2018): 637-656.
- [28] Dam, Hoa Khanh, Truyen Tran, and Aditya Ghose. "Explainable software analytics." In *Proceedings of the 40th international conference on software engineering: New ideas and Emerging results*, pp. 53-56. 2018.