Journal of Information Technology Management

Print ISSN: 2008-5893

Online ISSN: 2423-5059

A Decentralized Polling System Using Ethereum Technology

Samarth Shakya*

*Corresponding Author, MSc in Information Security, Department of Information Technology, Institute of Engineering Technology Devi Ahilya University, Indore 452017, India. E-mail: samarthshakya@gmail.com

Vivek Kapoor

Assistant Professor, Department of Information Technology, Institute of Engineering Technology Devi Ahilya University, Indore 452017, India. E-mail: vkapoor@ietdavv.edu.in

Abstract

Polling system is not trusted everywhere around the world it is very important in this modern world to replace the traditional polling system with the new technology. Some countries like United States, Japan, and India suffer from corrupted polling system. Major issues are faced by current polling systems like system hacking, vote rigging, vote manipulation, distributed denial of service attack, and online polling booth capturing. This paper will lead to the problems faced by the traditional polling system and how the new technology will provide the solution to that problem. Also, our purpose is to check the feasibility of the system by recording the transaction fees and evaluate the right way to spend the amount of gas in the transaction. This will highlight blockchain frameworks including blockchain as a service and polling system which is on blockchain that addresses all constraint introducing ethereum which is a blockchain-based distributed computing platform. Ethereum is open source, and publicly available with a system featuring smart contracts. It provides the cryptocurrency wallets that let you make cheap, instant payments with gas in the form of ethers. The ethereum community is the most active and largest blockchain community in the world. There is no centralized organization that controls ethereum.

Keywords: Blockchain; Ethereum; Decentralization; Gas; Distributed System; Metamask

DOI: https://10.22059/jitm.2022.85645 Manuscript Type: Research Paper University of Tehran, Faculty of Management

Received: January 12, 2021 Accepted: March 25, 2020

Introduction

Blockchain is relatively new technology, as we see the earlier work in Blockchain some of research is extended from side to side in the last ten years. The area of investigation are security and privacy check on blockchain and usage of blockchain in various ways with digital ledger techniques, their challenges and application. Many countries are seeking the opportunity and have taken some initiative to improve their existing voting system by making it a decentralized voting system with peer-to-peer network. The first country in the world to use blockchain technology is Sierra Leone to verify votes in an election in March, 2018. Blockchain is the best technology for polling systems because it provides the failure to modify or remove information from blocks makes the polling system immutable. Blockchain technology consists of a large number of interconnected nodes is supported by a distributed network. Nodes have their own copy of distributed ledger which have the record of all the transactions processed by the network. There will be no single authority that controls the network. This network allows users to vote anonymously. It is making e-voting acceptable as our modern democracies are built up on voting system. The Increase of voter lack of enthusiasm in recent years, especially among the younger generation which is more into computers and technology the e-voting is a potential solution to attract voters specially the young one. A robust polling scheme requires distribution of authority which is provided in blockchain. A great decentralized application utilizing blockchain technology allows you perform the same actions without a third party. Since a blockchain is a permanent record of transactions(votes) that are distributed on every node making votes immutable. Introducing a secured and customizable voting application made for everyday use built on the Ethereum blockchain, a 100% decentralized platform customizable and simple, so that you can remove your focus from security and focus on what matters.

Rationale

Polling system focuses on voting techniques which can be applied concomitantly to business purposes. The main objective of this is to bring together the seemingly disparate fields, such as Decentralized applications, Ethereum, Smart contracts, Blockchain and Knowledge discovery.

The problem with existing e-voting system is:

1. Centralized Architecture
2. Hacking of the centralized voting systems.
3. Election Manipulation.
4. Vote casting: Votes should be anonymous to everyone including the administrators.
5. Security problems: The DDoS attacks are well known attack in voting systems.
6. No transparency and trust: People usually do not trust when everything is online.

All these are the reasons which seeks and motivates to develop a decentralized system.

There are some of the objectives to which we have focus while developing a decentralized ethereum polling system. To make any transaction in ethereum based system we have to pay some transaction fee that is also known as gas. This gas is variable with the transaction and the words used by user in the transaction.

This research focuses on the feasibility of the polling system as we are going to use a large number of transactions. It also defines the proportion in which the transaction being processed. The agenda behind developing a polling system is to give people a right to vote on particular scenarios rather than electing a particular candidate to do so. People can also give the topics to get the votes from the people anonymously.

Background

The solution to the voting problem is to use ethereum technology. It serves the property of distribution, immutability, irreversibility, provides data-security and more importantly it is decentralized. The background working of ethereum blockchain technology can be classified in five main features:

a) Smart Contract: Developer creates a decentralized application while using the Ethereum technology with creation of a self-enforcing piece that managed by peer to peer network of computer according to the need. This smart contract can contain the business logic or agreement between the two or more people directly written in the lines of code. Smart contracts are created with a help of open source solidity programming platform like *remix.ethereum* and tested and deployed with the help of truffle framework.
b) Publishing: After the creation, that smart contract is published on ethereum blockchain and to publish it some amount of ether is deducted from the connected blockchain account. The publishing of smart contract can be done with truffle framework by a command *truffle migrate*.
c) User: To use the decentralized application user has to pay some amount of ether. To do so user/client must have an active ethereum account with a private key. The user account must contain some amount of gas to process the transaction. Transaction for each allowed feature can be processed only once. This feature gives the quality of uniqueness to the system. To pay ether user should have metamask extension in systems browser.
d) Miners: They verify or validate the transactions and add it to a new block these blocks together forms a ledger which is distributed among all the nodes. Based on a cryptographic hash algorithm they compete to solve a difficult mathematical problem and the solution is called the Proof-Of-Work. They are also rewarded with ether for each successful block.
e) Nodes: A infrastructure of blockchain is formed by nodes. They check a newly formed

block and add it to the blockchain. Any proposed “new block” to the ledger must reference the previous version of the ledger, which creates a chain possess immutability.

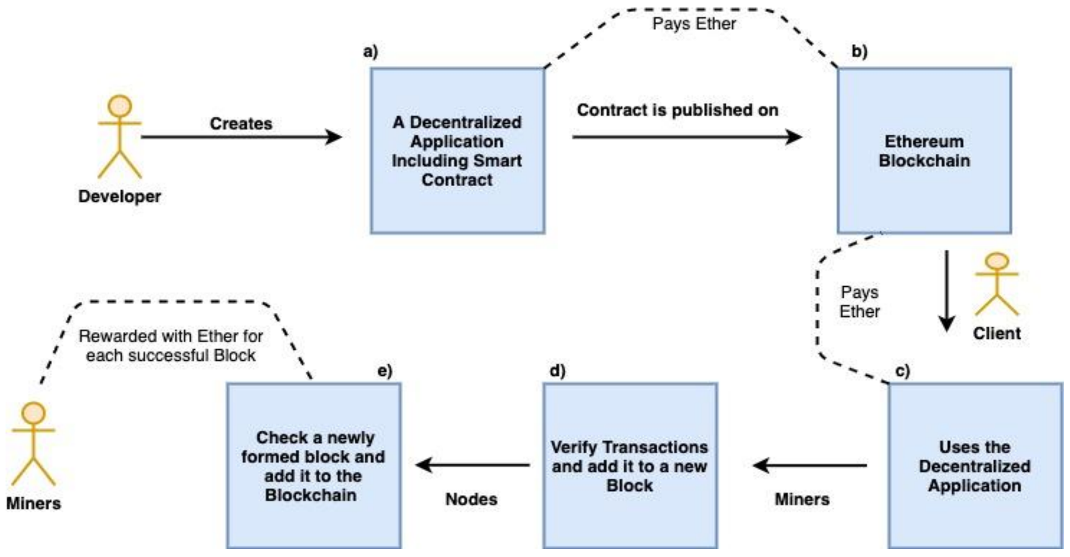

Pays Ether
a)
b)
Creates
Developer
A Decentralized Application Including Smart Contract
Contract is published on
Ethereum Blockchain
Pays Ether
Client
Rewarded with Ether for each successful Block
e)
d)
c)
Miners
Check a newly formed block and add it to the Blockchain
Nodes
Verify Transactions and add it to a new Block
Miners
Uses the Decentralized Application

Figure 1. Background flow of ethereum blockchain.

Proposed System

In this section we introduce our proposed polling system that aims at solving the existing barriers.

System Components

The proposed platform consists of the various following components:

1) Smart Contract: There is only one type of smart contract present in our system that is polling contract. This contract serves purpose to authenticate the voters and start the voting process. Polling contract increment the count of votes immediately when voted. It ensure the feature that only one vote is given with the private key. Appendices A list the code of our polling contract.

2) Ganache: It acts as a public blockchain dependency. Ganache is a personal blockchain deployed locally. We are going to use the ganache for the deployment of our polling contract and running tests. Appendices B list the free accounts provided by ganache to test out smart contract on local blockchain bases.

3) Truffle Framework: Truffle is a tool used to develop ethereum blockchain while using the solidity programming language. Truffle also provides various functionalities like automated testing, client side development, network and smart contract management. Appendices C shows the test result using truffle framework.

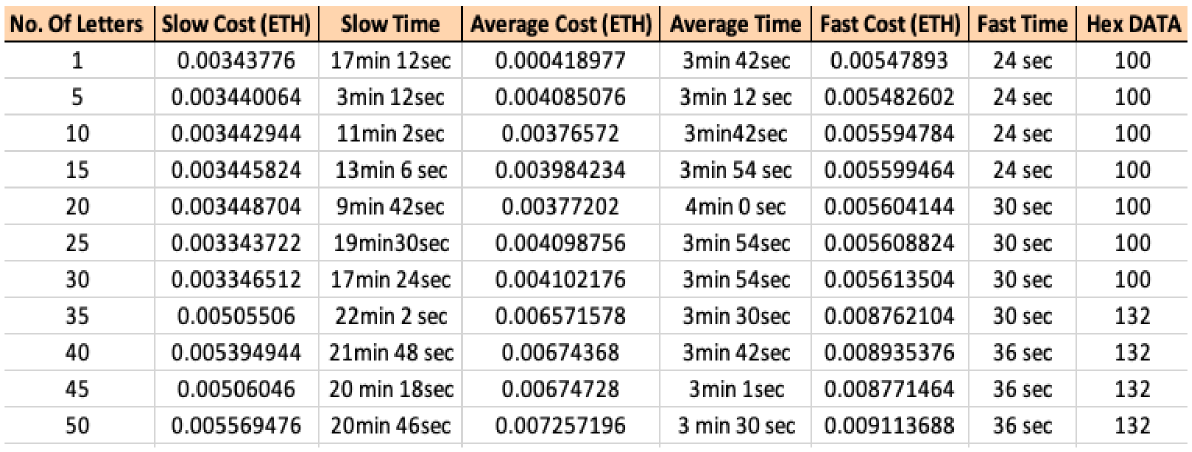
4) Metamask: If we want to use a user interface of any application our browser should support the connection to blockchain network. Metamask is the browser extension or

plugin used to connect to the required blockchain network. We can also manage our personal accounts in Metamask. We can install it in Chrome, Firefox and Opera. Appendices D shows the Metamask account in chrome browser.

Experiment

If we begin using polling application we will get to know that the amount of gas we are paying while using metamask is not constant. There is an option in metamask to choose the amount of gas we have to pay like slow, average and fast time transaction cost. To know how it is varying we had recorded the number of letters used to provide the topic to vote and the amount of gas which is in ETH required for them to complete the transaction as shown in Table 1.

Table 1. Experiment Data Record



No. Of Letters	Slow Cost (ETH)	Slow Time	Average Cost (ETH)	Average Time	Fast Cost (ETH)	Fast Time	Hex DATA
1	0.00343776	17min 12sec	0.000418977	3min 42sec	0.00547893	24 sec	100
5	0.003440064	3min 12sec	0.004085076	3min 12 sec	0.005482602	24 sec	100
10	0.003442944	11min 2sec	0.00376572	3min42sec	0.005594784	24 sec	100
15	0.003445824	13min 6 sec	0.003984234	3min 54 sec	0.005599464	24 sec	100
20	0.003448704	9min 42sec	0.00377202	4min 0 sec	0.005604144	30 sec	100
25	0.003343722	19min30sec	0.004098756	3min 54sec	0.005608824	30 sec	100
30	0.003346512	17min 24sec	0.004102176	3min 54sec	0.005613504	30 sec	100
35	0.00505506	22min 2 sec	0.006571578	3min 30sec	0.008762104	30 sec	132
40	0.005394944	21min 48 sec	0.00674368	3min 42sec	0.008935376	36 sec	132
45	0.00506046	20 min 18sec	0.00674728	3min 1sec	0.008771464	36 sec	132
50	0.005569476	20min 46sec	0.007257196	3 min 30 sec	0.009113688	36 sec	132

Results

With the recorded table we can conclude that the amount as gas is variable with number of letters used as well as the time required to complete the transaction. These can be easily understandable by the following recorded graphs.

Analysing the number of letters with amount of gas required.

Considering the Fast Cost (ETH) from the Table 1 we can see that the amount of ETH required is slightly increasing as we increase the number of letters in our transaction. That means if we want to create a block with a long phrase we need to pay the higher amount of gas for the transaction.

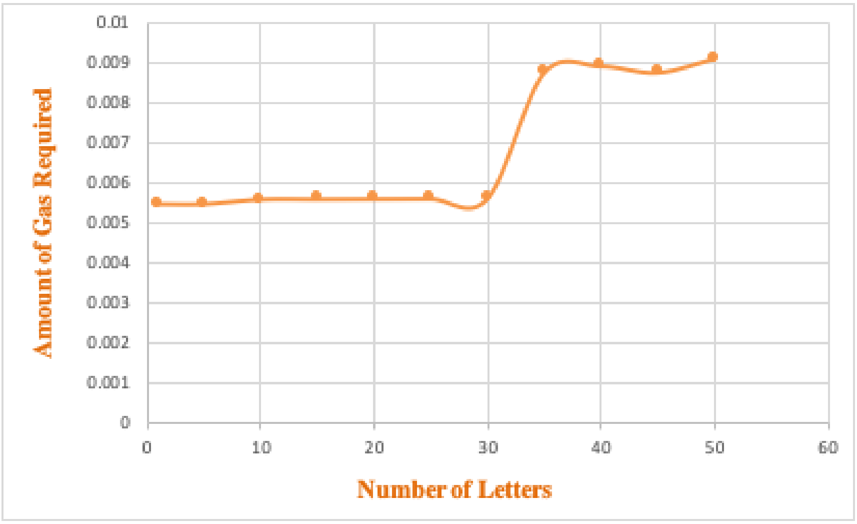


Figure 2. Number of letters vs amount of gas required.

Analysing the number of letters with time required to complete the transaction

Considering the Fast Time (seconds) from Table 1 we can conclude that the time required to complete the transaction is directly proportional to the number of letters in the phrase. More the number of letters in the phrase more time it will take to complete the transaction.

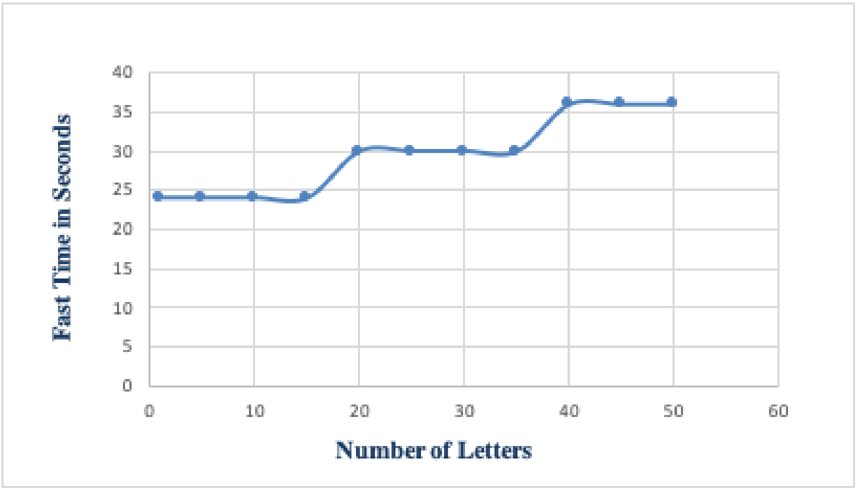


Figure 3. Number of letters vs time required.

Analysing the Efficient way to choose the transaction type.

We can see that Fast transaction type is better than any other type as there is only a slight change in the ETH but the time difference is very high in all other transaction as shown in Table 1. All the transaction type are showing the similar property as compare to the amount of gas which is a direct proportion.

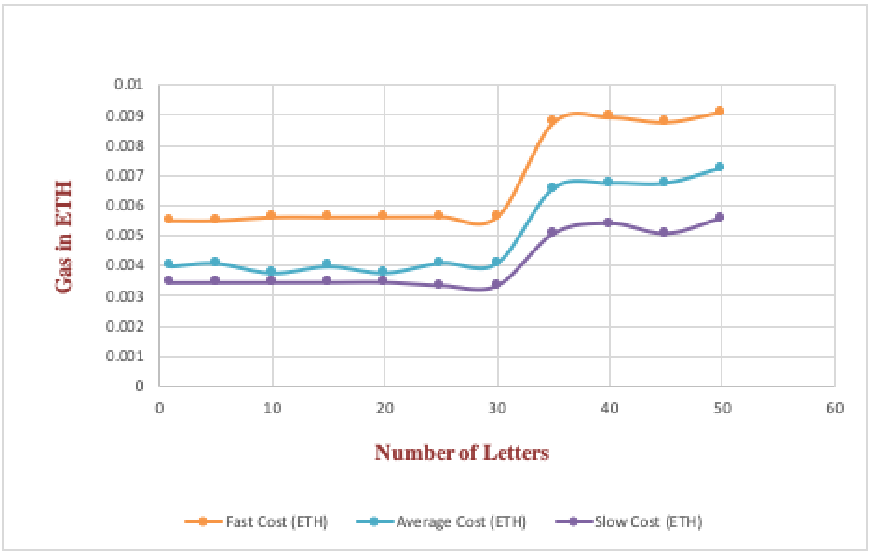


Figure 4. Number of letters vs Fast-Average-Slow Cost (ETH)

Conclusion

The paper analyzes the polling system on real world scenarios and implemented it using the ethereum technology. It also gives the experiment regarding the variability of amount of gas which we pay during transaction using metamask by finding the efficient way to do so. It also gives a brief idea about tools used for the developing an ethereum blockchain based application. By reading this paper carefully one can develop an ethereum blockchain application of their own by following the steps given. This polling system integrates the election voting to a new form where one can vote on real world scenario or one can clears their confusion regarding to any topic with the opinions of people.

Conflict of interest

The authors declare no potential conflict of interest regarding the publication of this work. In addition, the ethical issues including plagiarism, informed consent, misconduct, data fabrication and, or falsification, double publication and, or submission, and redundancy have been completely witnessed by the authors.

Funding

The author(s) received no financial support for the research, authorship, and/or publication of this article

References

Ben Ayed, A. (2017). A Conceptual Secure Blockchain- Based Electronic Voting System. *International Journal of Network Security & Its Applications (IJNSA)*, 9 (3).

Bhosale, K.; Akbarabbas, K.; Deepak, J. & Sankhe, A. (2019). Blockchain based Secure Data Storage. *International Research Journal of Engineering and Technology (IRJET),* 6 (3).

Bulut, R.; Kantarcı, A.; Keskin, S.; Bahtiyar, S. (2018). Blockchain-Based Electronic Voting System for Elections in Turkey. *Istanbul Technical University Istanbul, Turkey.*

Chan Zheng Wei, Clement; Chai Wen Chuah (2018): Blockchain-Based Electronic Voting Protocol. *International Journal On Informatics Visualization,* 2 (4).

H. Bergquist, Jonatan (2017): Blockchain Technology and Smart Contracts. *Uppsala Universitet Examensarbete 30 hp .*

Hatiskar, Vaibhav; G. Pai, Archana (2018): Blockchain and it's Integration with Supply Chain. *International Journal of Computer Applications (0975 – 8887),* 179 (52).

Kaan Koç, Ali; Yavuz, Emre; Can Çabuk, Umut; Dalkılıç, Gökhan (2018): Towards Secure E-Voting Using Ethereum Blockchain. *researchgate.net/publication/323318041.*

McCorry, Patrick; F. Shahandashti Siamak; Hao Feng (2017): A Smart Contract for Boardroom Voting with Maximum Voter Privacy. *School of Computing Science, Newcastle University UK.*

Khan, Tayyab, Karan Singh, Mohamed Abdel-Basset, Hoang Viet Long, Satya P. Singh, and Manisha Manjul. "A novel and comprehensive trust estimation clustering based approach for large scale wireless sensor networks." IEEE Access 7 (2019): 58221-58240.

Pareek, Shubham; Upadhyay, Anuj; Doulani, Satya; Tyagi, Siddarth; Varma, Aditya(2018): E-Voting using Ethereum Blockchain. *International Journal for Research Trends and Innovation,* 3 (11).

Shrinivas, Manoj; S.Chandan; Farhan Shamail, Mohammed; K, Ramyashree (2019): A Decentralized Voting Application using Blockchain Technology. *International Research Journal of Engineering and Technology (IRJET),* 6 (4).

Tso, Raylin; Liu, Zi-Yuan; Hsiao, Jen-Ho(2019): Distributed E-Voting and E-Bidding Systems Based on Smart Contract. *Multidisciplinary Digital Publishing Institute.*

V. Arun; Dutta, Aditya; Rajeev, Sourav; Mathew Varghese, Rohan (2019): E-Voting using a Decentralized Ethereum Application. *International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249-8958,* 8 (4).

www.dappuniversity.com/articles/the-ultimate-ethereum-dapp-tutorial (Building an Ethereum Decentralized Application)

Bibliographic information of this paper for citing:

Shakya, Samarth, & Kapoor, Samarth (2022). A Decentralized Polling System Using Ethereum Technology. *Journal of Information Technology Management,* Special Issue, 1-8.

Analysis of Effects of Change of Gear Parameter Module on Transmission Error in Spur Gear using Interference Volume Method

To cite this article: V Karma *et al* 2022 *IOP Conf. Ser.: Mater. Sci. Eng.* **1225** 012036

Analysis of Effects of Change of Gear Parameter Module on Transmission Error in Spur Gear using Interference Volume Method

V Karma [1, 4], G Maheshwari [2] and S K Somani [3]

[1,2] Mechanical Engineering Department, Institute of Engineering & Technology, Devi Ahilya University, Indore, Madhya Pradesh, 452001, India [3] Vice Chancellor, Oriental University, Indore, Madhya Pradesh, India

vkarma@ietdavv.edu.in

Abstract. Gear drives are the most used elements in power transmission systems. Any defect in the gear of these drives leads to noise and vibrations, which affect power/torque transmission. Transmission Error (TE) is one of the critical causes that arises due to the tooth geometry error, cracks in gear, profile error etc. Our work demonstrates, the consequences occurring due to the changes in gear parameter module on TE for spur gear pair of 1:1 gear ratio with pitch error, using the interference volume method. The spur gear without pitch error and with pitch error having the involute profile of standard tooth dimension system and stub tooth dimension system are modelled and assembled in CAD software SOLIDWORKS. The interference volume values obtained from the gear pairs are recorded for one mesh cycle. Various graphs plotted between the angle of rotation and interference volume values for one mesh cycle. It is found from the analysis that as the module and pitch error increase TE also increases. The effect of variation of module and pitch error is more in the standard tooth dimension system than the stub tooth dimension system.

Keywords: CAD, Gear Ratio, Pitch Error, No of Teeth, SOLIDWORKS, Geometry Error

1. Introduction

Gears drives are the most used elements employed to transfer power/torque. The condition to accurately transfer power/torque is that the drive elements such as gears should be free from any error else it could lead to noise and vibrations, which results in the development of Transmission Error (TE) despite of availability of world class manufacturing and design facilities. TE is a show of reading measured of angular or linear displacement by the side of activity on the base circle [1]. Hotait and Kahraman [2] demonstrated experimentally the association connecting dynamic factor and transmission error measurement. The values were obtained from unabated and adapted spur gears using a gear dynamic test set-up. Kohler and Regan [3] concluded that TE has harmonic components of significant amplitude at tooth contact frequency spectrum. Chang and Tang [4] conducted theoretical analysis under the influence of single and cumulative pitch error to investigate the nonlinear vibration response in a double-helical gear set. Velex et al [5] established theoretically a relationship in the middle of active mesh agitation and transmission errors using 3D analysis modal of

multi mesh gears. They demonstrated that the local confined TE is analogous with each individual mesh controls the dynamic mesh forces and provided a design criterion to minimize dynamic tooth loads by defining tooth shape modifications. Nina [6] developed equations of motion based on time varying mesh stiffness function, loaded quasistatic TE, and no-load or kinematic TE. Studied the effect of tooth pitch error and load on dynamic behavior of gear transmission. She also investigated the relationship between dynamic TE and mesh force or root stress. Sánchez et al [7] work is done to study the contact conditions of modified teeth under load. They widely explained the results by seeing the changes of various parameters of profile modification on actual load sharing quotient and TE. Palermo et al [8] in their work tried to calculate TE in comparison to the direct process which uses low-cost digital encoders, which are embedded on an accuracy gear pair test. Shweiki et al [9] does work to find the change in conduct when a quality and light weight cylindrical gear meshes. An elastic multi body expressions which has hybrid approach in FE method in the background representation is used by them. Zhiying and Pengfei [10] proposed and validated, analytical mesh stiffness and quasi static TE models with finite element method to find the inner working of spur gear meshing pairs. Pleguezuelos et al [11] found out the effect of changes in symmetric long on high contact ratio spur. The change is represented empirically in the optimum length as a function of the contact ratio. Oswald [12] works found out the efficacy in modification of profile in ever changing reduced loads in gears having spacing errors. The analysis considers the modifications of both linear and parabolic profiles. Bruyere et al [13] presented a closed-form analytical formula of TE for helical and solid spur gears for minimizing time varying amplitudes of TE amalgamation of lead crouch and profile relief. They showed that the optimum tooth modifications depend on ratio of profile Vs face contact, the normalized depth and number of modifications in profile in continuation with the normalized lead crown amplitude. Handschuh [14] experimentally, empirically, and mathematically find out the consequences of tooth spacing error on TE and root stresses in spur gear pairs. Bartosova et al [15] uses finite element analysis to find out TE at variety of combination of load states, axial distant variation, and tooth shape elevation modification. Park [16] find out friction in tooth and TE in spur gears which occurs due to result of friction sliding in quasi static condition. Somani [17] worked in his doctorate thesis on studying the effect of changing the various gear parameters on TE in spur gear.
Transmission through gears is used in every area of industry to convey power coming from shaft. A lot of research work is there in public domain in this area. Bruzzone et al [18] constructed model using multi-layered process in quasi-static state to find degree of deflection, tooth deformation and stiffness. Experiments results in terms of STE with different parameters has been found. TE is the main cause of vibration in gear box. Duan et al [19] established a test rig and made comparison between the result obtained theoretically and experimentally. Results shows that under heavy load conditions TE increases, but with the use of thin-walled housing it decreases considerably. A lot of study about the TE in gears is being done. Benaicha et al [20] in his work tries to develop a model which gives the insight efficient inaccurate working of gear TE. Results were compared with the earlier results obtained from classical techniques. Superiority of the model which is made was confirmed in his work. Backlash is also a type of TE which occurs due to the gap between the teeth at the pitch circles. This error is unavoidable as the time passes by. Ambaye et al [21] uses a Gear Trax software to model a gear with backlash with different degrees. In his experiments he uses plane strain analysis and finite element analysis to find mesh convergence for different contact pressure and stress. In this process he was able to predict backlash with a high degree of accuracy. Gearbox is used to transmit power from shaft to rotor in various machines. In case of wind turbine, it is important link to transmit power. Tao et al [22] uses Flank Pitch Error model to find gear faults. He proposed time varying meshing stiffness model to simulate meshing frequency and find severity of gear box vibration. Delay in drive which is popularly known as loaded TE is caused due to tooth deflection, manufacturing defects and assembly error in spur gear. Miguel et al [23] proposed a mathematical model to consider of meshing stiffness, load sharing ratio and TE for spur gears under minimal or varying load conditions. Gear TE causes vibration in moving parts. Chin et al [24] in his work find TE. His work is validated by readings

obtained from encoder and tachometer from a drive gear wear experimental setup. Model prepared by him prove to be very robust under sever conditions also.
From the available literature, it is observed that the researchers are working towards the determination of TE analytically, experimentally and or using finite element analysis for the gears having defects due to manufacturing, geometry error, crack, tooth breakage etc., in the standard tooth dimension system. In today’s scenario there is an urgent need of more simplified methods/procedure using CAD software to analyze the various manufacturing defects, geometry errors etc., in spur gears. A lot of work is to be done by considering the gears in both standard and stub tooth dimension system. Hence, the objective of the current work is to analyze the TE using interference volume method in stub tooth dimension system and standard tooth dimension system for spur gear having pitch error in a simplified procedure. The gears are modelled in SOLIDWORKS software. Various gear parameters used are the modules 3, 3.5, 4, 4.5, and 5 mm, pressure angle 20°, and the number of teeth 18. The pitch error of magnitudes 1% and 2% is deliberately introduced in all the teeth of gear. The gear without pitch error is meshed with a gear having pitch error using the assembly module of SOLIDWORKS software. The interference volume values are then recorded for one mesh angle of the gear pair. The graphs are plotted between angles turned vs. interference volume values, and for each change in module and pitch error, the effect on the TE is predicted for both standard as well as stub tooth dimension system. It is observed that TE is affected when module and pitch error magnitude is change but the effect is more in case of standard tooth dimension system than in stub tooth dimension system. The analysis is done using the procedure as described by the authors in [25]. The work of the paper/outcomes are in line with the work of researchers presented in [3, 4, 6, 12, 14, 17].

2. Modelling of Master Gear and Erred Gear

The spur gears are modelled with true involute curve from mathematical equation of involute curve in CAD software SOLIDWORKS. The following spur gears are modelled for the analysis. The designation of gears and their pairs are described below.

a) Spur Gear of standard tooth dimension system and stub tooth dimension system without any defect or error are designated as Master Gear [MG] and Stub Master Gear [SMG] respectively.
b) Spur Gear of standard tooth dimension system and stub tooth dimension system with pitch error are designated as Erred Gear [E] and Stub Erred Gear [SE] respectively. Gear with 1% and 2% pitch error are designated as E1 and E2 in standard tooth dimension system, SE1 and SE2 in stub tooth dimension system, respectively.

After modelling of the master gear, erred gears are modelled by inducing pitch error of 1% and 2% magnitude. The pitch error is created by increasing the tooth thickness to 1% and 2% of its original value. These deviations are sufficient deviation from the original value in manufacturing of any gear. A nominal face width of 10 mm is considered for the gears. The gears with and without pitch error in standard tooth dimension system and stub tooth dimension system of module 3, 3.5, 4, 4.5 and 5 mm, number of teeth 18 and pressure angle 20° are designated in table 1.
For example as in table1, a gear of module 3 mm, number of teeth 18, pressure angle 20°, and without pitch error in standard tooth dimension system is designated as MG_m3_z18_phi20, and with pitch error of 1% and 2% is designated as E1_m3_z18_phi20 and E2_m3_z18_phi20 respectively (where MG is Master Gear, E1 is Erred Gear (1% pitch error), E2 is Erred Gear (2% pitch error), m is module, z is number of teeth and phi is pressure angle). Similarly a gear of module 3 mm, number of teeth 18, pressure angle 20°, and without pitch error in stub tooth dimension system is designated as SMG_m3_z18_phi20, and with pitch error of 1% and 2% is designated as SE1_m3_z18_phi20 and SE2_m3_z18_phi20 respectively (where SMG is Master Gear, SE1 is Erred Gear (1% pitch error), SE2 is Erred Gear (2% pitch error), m is module, z is number of teeth and phi is pressure angle). The various values of the spur gear parameters for the two tooth dimension systems are given in table 2 and table 3.

Table 1. Designation of various spur gears in two tooth dimension systems

Standard Tooth Dimension System		Stub Tooth Dimension System	
Master Gear / Gear without Pitch Error	Erred Gear / Gear with Pitch Error of 1% & 2%)	Master Gear / Gear without Pitch Error	Erred Gear / Gear with Pitch Error of 1% & 2%)
MG_m3_z18_phi20	E1_m3_z18_phi20	SMG_m3_z18_phi20	SE1_m3_z18_phi20
	E2_m3_z18_phi20		SE2_m3_z18_phi20
MG_m3.5_z18_phi20	E1_m3.5_z18_phi20	SMG_m3.5_z18_phi20	SE1_m3.5_z18_phi20
	E2_m3.5_z18_phi20		SE2_m3.5_z18_phi20
MG_m4_z18_phi20	E1_m4_z18_phi20	SMG_m4_z18_phi20	SE1_m4_z18_phi20
	E2_m4_z18_phi20		SE2_m4_z18_phi20
MG_m4.5_z18_phi20	E1_m4.5_z18_phi20	SMG_m4.5_z18_phi20	SE1_m4.5_z18_phi20
	E2_m4.5_z18_phi20		SE2_m4.5_z18_phi20
MG_m5_z18_phi20	E1_m5_z18_phi20	SMG_m5_z18_phi20	SE1_m5_z18_phi20
	E2_m5_z18_phi20		SE2_m5_z18_phi20

3. Meshing and Analysis of Gear Pair of Standard Tooth Dimension System

After modelling of various gears, it is required to form assemblies/pairs (meshing of Master and Master or Master and Erred Gear) for doing the analysis. The various assemblies or combinations formed, and their designations are shown in table 4. Each of these assemblies or combinations are taken one by one for determining the TE. The interference volume is determined for all these gear pairs/assemblies. From the values of the interference volume obtained, the TE variation is determined for all these spur gear pairs. The methodology to determine the TE is verified from the pair of master gear with master gear and then the master gear and erred gear pair is taken for the analysis. The two gears are meshed using the assembly module of SOLIDWORKS and interference volume between the two gears in mesh is checked for initial meshed position by invoking the command of finding interference volume and then the value of interference volume is recorded.

Table 2. Values of Spur Gear parameters in Standard Tooth dimension System

Paramet ers [26]	Module, mm (m)	No. of Teeth (z)	Pitch Circle Diamete r, mm (m*z)	Pressure Angle, ° (Phi)	Addend um, mm (1*m)	Dedend um, mm (1.25*m)	Base Circle Diamete r, mm (m*z*co s(phi))	Circular pitch, mm (πm)	Tooth Thick- ness, mm (πm/2)
	3.0	18	54	20	3.0	3.750	50.7434	9.4248	4.7124
Master	3.5	18	63	20	3.5	4.375	59.2006	10.9956	5.4978
Gear	4.0	18	72	20	4.0	5.000	67.6578	12.5664	6.2832
(MG)	4.5	18	81	20	4.5	5.625	76.1151	14.1372	7.0686
	5.0	18	90	20	5.0	6.250	84.5723	15.7080	7.8540
	3.0	18	54	20	3.0	3.750	50.7434	9.5190	4.7595
Erred	3.5	18	63	20	3.5	4.375	59.2006	11.1055	5.5528
Gear	4.0	18	72	20	4.0	5.000	67.6578	12.6920	6.3460
(E1)	4.5	18	81	20	4.5	5.625	76.1151	14.2785	7.1393
	5.0	18	90	20	5.0	6.250	84.5723	15.8650	7.9325
	3.0	18	54	20	3.0	3.750	50.7434	9.6133	4.8066
Erred	3.5	18	63	20	3.5	4.375	59.2006	11.2155	5.6077
Gear	4.0	18	72	20	4.0	5.000	67.6578	12.8177	6.4088
(E2)	4.5	18	81	20	4.5	5.625	76.1151	14.4199	7.2100
	5.0	18	90	20	5.0	6.250	84.5723	16.0221	8.0111

For obtaining the next value of interference volume, one of the gears of gear pair is rotated by 1° and due to the meshing, the other gear is also rotated by 1° in opposite direction. The interference volume between the two gears is checked again and the value is noted. In this way for 21 such positions the two gears are rotated and at every instance the interference volume is checked and noted (The interference volume cycle repeats after 20° of rotation for 18 number of teeth). All the interference volume values are recorded in table 5. Figure 1 shows one of the gear pair of master gear and erred gear 1 (MG_E1_m3.5_z18_phi20).

Table 3. Dimensions of Spur Gear in Stub Tooth dimension System

Parameters [26]	Module, mm (m)	No. of Teeth (z)	Pitch Circle Diameter, mm (m*z)	Pressure Angle, ° (Phi)	Addendum, mm (0.8*m)	Dedendum, mm (1.0*m)	Base Circle Diameter, mm (m*z*cos(phi))	Circular pitch, mm (πm)	Tooth Thick-ness, mm (πm/2)
Master Gear (SMG)	3.0	18	54	20	2.4	3.0	50.7434	9.4248	4.7124
	3.5	18	63	20	2.8	3.5	59.2006	10.9956	5.4978
	4.0	18	72	20	3.2	4.0	67.6578	12.5664	6.2832
	4.5	18	81	20	3.6	4.5	76.1151	14.1372	7.0686
	5.0	18	90	20	4.0	5.0	84.5723	15.7080	7.8540
Erred Gear (SE1)	3.0	18	54	20	2.4	3.0	50.7434	9.5190	4.7595
	3.5	18	63	20	2.8	3.5	59.2006	11.1055	5.5528
	4.0	18	72	20	3.2	4.0	67.6578	12.6920	6.3460
	4.5	18	81	20	3.6	4.5	76.1151	14.2785	7.1393
	5.0	18	90	20	4.0	5.0	84.5723	15.8650	7.9325
Erred Gear (SE2)	3.0	18	54	20	2.4	3.0	50.7434	9.6133	4.8066
	3.5	18	63	20	2.8	3.5	59.2006	11.2155	5.6077
	4.0	18	72	20	3.2	4.0	67.6578	12.8177	6.4088
	4.5	18	81	20	3.6	4.5	76.1151	14.4199	7.2100
	5.0	18	90	20	4.0	5.0	84.5723	16.0221	8.0111

Table 4. Combinations/Assemblies of Various Gears (Standard Tooth Dimension System)

Module, mm (m)	No. of Teeth, (z)	Pressure Angle, ° (phi)	Meshing of Master Gear with Master Gear	Meshing of Master Gear with Erred Gear 1	Meshing of Master Gear with Erred Gear 2
3			MG_MG_m3_z18_phi20	MG_E1_m3_z18_phi20	MG_E2_m3_z18_phi20
3.5			MG_MG_m3.5_z18_phi20	MG_E1_m3.5_z18_phi20	MG_E2_m3.5_z18_phi20
4	18	20	MG_MG_m4_z18_phi20	MG_E1_m4_z18_phi20	MG_E2_m4_z18_phi20
4.5			MG_MG_m4.5_z18_phi20	MG_E1_m4.5_z18_phi20	MG_E2_m4.5_z18_phi20
5			MG_MG_m5_z18_phi20	MG_E1_m5_z18_phi20	MG_E2_m5_z18_phi20

The detailed procedure for determining the TE for all the combinations is explained in the sections from 3.1 to 3.3

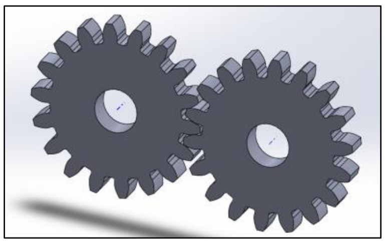


Figure 1. Assembly/Meshing of Master Gear and Erred Gear 1 of standard tooth dimension system having module 3.5 mm, number of teeth 18 and pressure angle 20° (MG_E1_m3.5_z18_phi20).

3.1. Meshing of Master Gear with Master Gear (MG_MG)

Two gears forming a pair with combinations as shown in table 4 are taken one by one for analysis. The two gears are meshed and for this initial position interference between them is checked. Then gear pairs are rotated for contact angle (20° in case of 18 number of teeth). It is observed that the two gears are just touching each other, indicating that there is no interference occurring between them and hence the interference volume is zero. This also confirm the correctness of procedure.

3.2. Meshing of Master Gear with Erred Gear 1 (MG_E1)

In this case master gear is meshed with erred gear 1 (gear with 1% pitch error). The various combinations formed are represented in table 4. The same procedure as discussed in section 3.1 above is followed to determine the interference volume. Here it is observed that there is interference occurring in the two gears in mesh during rotation of contact angle (20° in case of 18 number of teeth). The values of interference volume are noted and tabulated in table 5 for plotting graph between interference volume vs rotation angle.

3.3. Meshing of Master Gear with Erred Gear 2 (MG_E2)

Now finally the master gear is meshed with erred gear 2 (gear with 2% pitch error). The various combinations are given in table 4. Here also similar procedure as discussed in section 3.1 is used to check the interference between the gears. It is again observed that there is an interference occurring in gears in mesh. The interference volume values for contact angle of 20° rotation is noted in table 5. Based on the interference volume values of table 5, various graphs are plotted between angle of rotation and interference volume values. These graphs are shown in the figure 2 and figure 3, respectively.

Table 5. Interference Volume values (mm^3) when Master Gear meshes with the Master Gear, Erred Gear 1, and Erred Gear 2 for Standard Tooth Dimension System

Angle of Rotation (°)	Master Gears [MG]	MG_ E1_ m3_ z18_ phi 20	MG_ E1_ m3.5_ z18_ phi 20	MG_ E1_ m4_ z18_ phi 20	MG_ E1_ m4.5_ z18_ phi 20	MG_ E1_ m5_ z18_ phi 20	MG_ E2_ m3_ z18_ phi 20	MG_ E2_ m3.5_ z18_ phi 20	MG_ E2_ m4_ z18_ phi 20	MG_ E2_ m4.5_ z18_ phi 20	MG_ E2_ m5_ z18_ phi 20
0	0	0.4000	0.5400	0.7000	0.8800	1.1000	1.1000	1.4800	1.9400	2.4400	3.0200
1	0	0.3700	0.5000	0.6500	0.8200	1.0000	1.0500	1.4300	1.8700	2.3600	2.9300
2	0	0.3600	0.5000	0.6300	0.8100	1.0000	1.0100	1.3800	1.8000	2.2900	2.8200
3	0	0.3600	0.4900	0.6500	0.8200	1.0100	1.0300	1.3900	1.8200	2.3000	2.8300
4	0	0.3600	0.5000	0.6500	0.8200	1.0100	1.0300	1.4000	1.8300	2.3100	2.8600
5	0	0.3700	0.4900	0.6600	0.8200	1.0100	1.0400	1.4000	1.8300	2.3200	2.8700
6	0	0.3600	0.5000	0.6500	0.8200	1.0100	1.0300	1.4000	1.8300	2.3200	2.8600
7	0	0.3600	0.4900	0.6500	0.8200	1.0100	1.0200	1.3900	1.8200	2.3000	2.8400
8	0	0.3600	0.4900	0.6300	0.8100	1.0000	1.0100	1.3800	1.8000	2.2900	2.8200
9	0	0.3700	0.5000	0.6500	0.8200	1.0100	1.0600	1.4500	1.9000	2.3900	2.9700
10	0	0.4000	0.5500	0.7000	0.9000	1.1000	1.1000	1.5200	1.9800	2.5000	3.0800
11	0	0.3700	0.5000	0.6500	0.8200	1.0100	1.0600	1.4500	1.9000	2.3900	2.9700
12	0	0.3600	0.4900	0.6300	0.8100	1.0000	1.0100	1.3800	1.8000	2.2900	2.8200
13	0	0.3600	0.5000	0.6500	0.8200	1.0100	1.0200	1.3900	1.8200	2.3000	2.8400
14	0	0.3600	0.4900	0.6500	0.8200	1.0100	1.0300	1.4000	1.8300	2.3200	2.8600
15	0	0.3700	0.5000	0.6600	0.8200	1.0100	1.0400	1.4000	1.8300	2.3200	2.8700
16	0	0.3600	0.4900	0.6500	0.8200	1.0100	1.0300	1.4000	1.8300	2.3100	2.8600
17	0	0.3600	0.5000	0.6500	0.8200	1.0100	1.0300	1.3900	1.8200	2.3000	2.8300
18	0	0.3600	0.4900	0.6300	0.8100	1.0000	1.0100	1.3800	1.8000	2.2900	2.8200
19	0	0.3700	0.4900	0.6500	0.8200	1.0000	1.0500	1.4300	1.8700	2.3600	2.9300
20	0	0.4000	0.5400	0.7000	0.8800	1.1000	1.1000	1.4800	1.9400	2.4400	3.0200

4. Meshing and Analysis of Gear Pair of Stub Tooth Dimension System

In this case the various gear pairs formed are listed in table 6. First the master gear is meshed with the master gear and the interference volume values obtained by rotation of gear pair for one mesh cycle is listed in table 7. Next using the same procedure, the interference volume values are obtained for master gear - erred gear 1 and master gear - erred gear 2 pair. All these values are listed in Table 7. Using these values various graphs are plotted between the interference volume values and angle of rotation. These are represented in figure 4 and figure 5 respectively.

Table 6. Combinations/Assemblies of Gears (Stub Tooth Dimension System)

Module, mm (m)	No. of Teeth, (z)	Pressure Angle, ° (phi)	Meshing of Master Gear with Master Gear	Meshing of Master Gear with Erred Gear 1	Meshing of Master Gear with Erred Gear 2
3			SMG_SMG_m3_z18_phi20	SMG_SE1_m3_z18_phi20	SMG_SE2_m3_z18_phi20
3.5			SMG_SMG_m3.5_z18_phi20	SMG_SE1_m3.5_z18_phi20	SMG_SE2_m3.5_z18_phi20
4	18	20	SMG_SMG_m4_z18_phi20	SMG_SE1_m4_z18_phi20	SMG_SE2_m4_z18_phi20
4.5			SMG_SMG_m4.5_z18_phi20	SMG_SE1_m4.5_z18_phi20	SMG_SE2_m4.5_z18_phi20
5			SMG_SMG_m5_z18_phi20	SMG_SE1_m5_z18_phi20	SMG_SE2_m5_z18_phi20

Table 7. Interference Volume values (mm^3) when Master Gear meshes with Master Gear, Erred Gear 1, and Erred Gear 2 for Stub Tooth Dimension System

Angle of Rotation (°)	Master Gears [SMG]	SMG_SE1_m3_z18_phi 20	SMG_SE1_m3.5_z18_phi 20	SMG_SE1_m4_z18_phi 20	SMG_SE1_m4.5_z18_phi 20	SMG_SE1_m5_z18_phi 20	SMG_SE2_m3_z18_phi 20	SMG_SE2_m3.5_z18_phi 20	SMG_SE2_m4_z18_phi 20	SMG_SE2_m4.5_z18_phi 20	SMG_SE2_m5_z18_phi 20
0	0	0.2590	0.3526	0.4604	0.5828	0.7194	0.7310	0.9950	1.2994	1.6446	2.0304
1	0	0.2583	0.3517	0.4594	0.5814	0.7178	0.7495	1.0202	1.3324	1.6864	2.0819
2	0	0.2935	0.3995	0.5219	0.6605	0.8154	0.8449	1.1499	1.5020	1.9009	2.3468
3	0	0.3476	0.4731	0.6181	0.7822	0.9658	0.9557	1.2989	1.6992	2.1506	2.6550
4	0	0.3651	0.4969	0.6492	0.8215	1.0143	1.0238	1.3937	1.8201	2.3036	2.8439
5	0	0.3665	0.4988	0.6516	0.8246	1.0181	1.0306	1.4028	1.8323	2.3189	2.8629
6	0	0.3651	0.4971	0.6492	0.8217	1.0144	1.0261	1.3965	1.8242	2.3087	2.8504
7	0	0.3497	0.4759	0.6216	0.7867	0.9713	0.9646	1.3124	1.7150	2.1705	2.6796
8	0	0.2963	0.4032	0.5267	0.6666	0.8229	0.8558	1.1641	1.5215	1.9256	2.3774
9	0	0.2585	0.3518	0.4592	0.5816	0.7181	0.7556	1.0279	1.3433	1.7001	2.0989
10	0	0.2588	0.3522	0.4602	0.5824	0.7190	0.7300	0.9936	1.2978	1.6426	2.0638
11	0	0.2585	0.3518	0.4624	0.5816	0.7181	0.7556	1.0289	1.3677	1.7001	2.0989
12	0	0.2963	0.4032	0.5402	0.6666	0.8229	0.8558	1.1656	1.5215	1.9256	2.3774
13	0	0.3497	0.4759	0.6303	0.7867	0.9713	0.9646	1.3137	1.7150	2.1705	2.6796
14	0	0.3650	0.4971	0.6495	0.8217	1.0144	1.0261	1.3968	1.8242	2.3087	2.8504
15	0	0.3664	0.4988	0.6514	0.8246	1.0181	1.0306	1.4029	1.8323	2.3189	2.8629
16	0	0.3655	0.4969	0.6490	0.8215	1.0143	1.0238	1.3934	1.8201	2.3036	2.8439
17	0	0.3548	0.4731	0.6074	0.7832	0.9658	0.9557	1.3003	1.6992	2.1506	2.6550
18	0	0.3043	0.3995	0.5090	0.6605	0.8154	0.8449	1.1493	1.5020	1.9009	2.3468
19	0	0.2602	0.3517	0.4595	0.5814	0.7178	0.7495	1.0198	1.3324	1.6864	2.0819
20	0	0.2590	0.3526	0.4604	0.5828	0.7195	0.7310	0.9950	1.2994	1.6446	2.0304

5. Result and Discussions

Based on the values of interference volume values for different cases of change of module and pitch error as reported in table 5 and table 7 for standard tooth dimension system and stub tooth dimension system respectively, various graphs are plotted between the angle of rotation on x axis and the interference volume values on y axis, respectively.

a. Figure 2 and Figure 3 shows the graph between interference volume values and rotation angle of gear pair for 1% and 2% pitch error cases with module changes from 3 to 5 mm respectively for standard tooth dimension system. The x axis contains the rotation angle in degree (°) and the y axis represents the interference volume values in cubic mm.
b. Similarly, the graph between interference volume values and rotation angle of gear pair for 1% and 2% pitch error cases with module changes from 3 to 5 mm for stub tooth dimension system is shown in Figure 4 and Figure 5, respectively. The x axis contains the rotation angle in degree (°) and the y axis represents the interference volume values in cubic mm. The angle of rotation is 20° since the curve repeats after 20° for 18 number of teeth.
c. In this paper the effects of variation of module on TE along with pitch error in gear is considered for the analysis purpose.

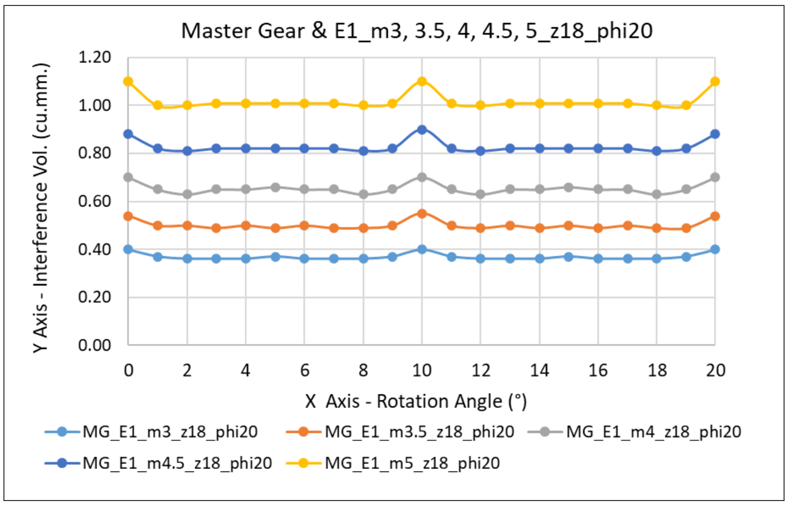


Figure 2. Graph between interference volume in cu. mm and angle of rotation (°) for analysing the effects of change of Module (3, 3.5, 4, 4.5 and 5) on TE with pitch error of 1% (Master Gear & Erred Gear 1). [Standard Tooth Dimension System]

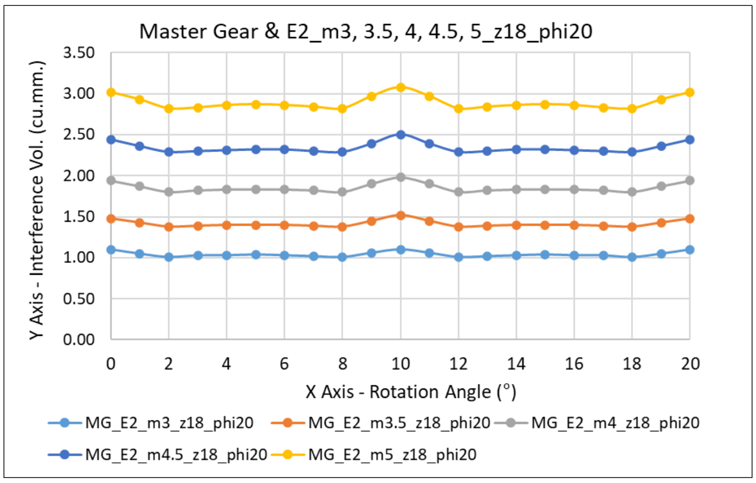


Figure 3. Graph between interference volume in cu. mm and angle of rotation (°) for analysing the effect of change of Module (3, 3.5, 4, 4.5 and 5) on TE with pitch error of 2% (Master Gear & Erred Gear 2). [Standard Tooth Dimension System]

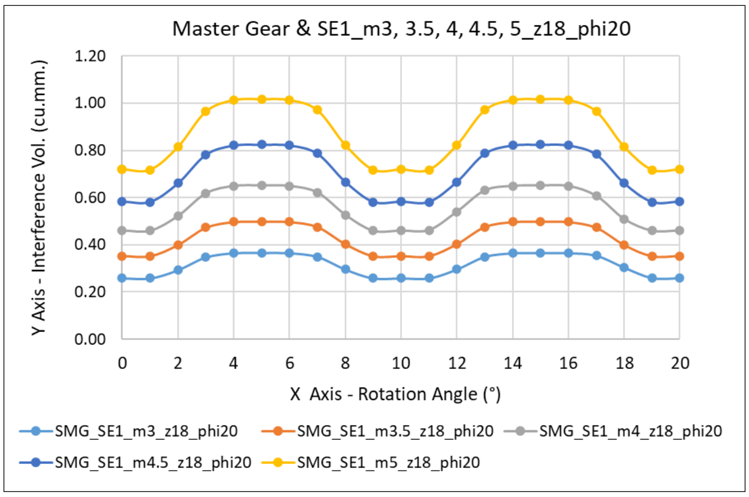


Figure 4. Graph between interference volume in cu. mm and angle of rotation (°) for analysing the effects of change of Module (3, 3.5, 4, 4.5 and 5) on TE with pitch error of 1% (Master Gear & Erred Gear 1). [Stub Tooth Dimension System]

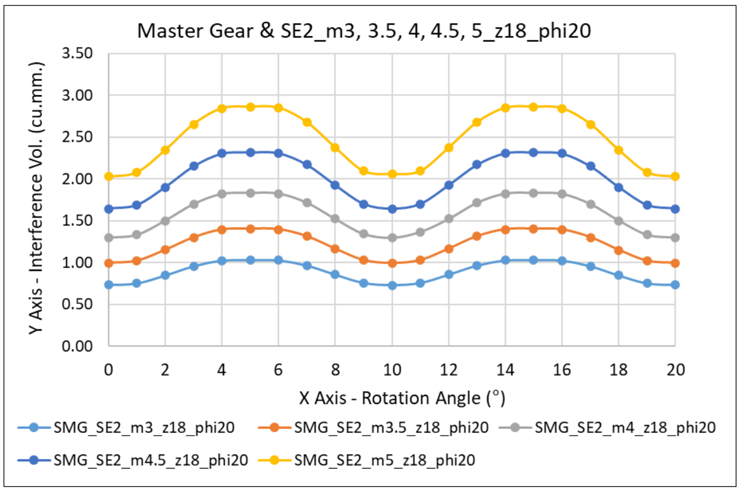


Figure 5. Graph between interference volume in cu. mm and angle of rotation (°) for analysing the effects of change of Module (3, 3.5, 4, 4.5 and 5) on TE with pitch error of 2% (Master Gear & Erred Gear 2). [Stub Tooth Dimension System]

Following are the observations drawn from the graphs of interference volume values and angle of rotation (figure 2 to figure 5)

1. There is a fluctuation in the values of interference volume, when angle of rotation changes from 0 to 20° (start and end of mesh cycle), there are some points having peak values of interference volume. This means that the TE is not constant for angle of rotation. The points having peak values are the point of concerned for the noise and vibrations in gears. This is observed for both the tooth dimension system i.e., standard tooth dimension system and stub tooth dimension system.
2. As the module changes from 3 to 5 mm, the interference volume values increases and hence the TE increases in standard as well as in stub tooth dimension system.
3. The interference volume values are found to be more in case of 2% pitch error than 1% pitch error and consequently the TE is more in 2% pitch error case. This is also observed in both tooth dimension system.

4. The graph is identical about its mid-point. i.e., it repeats after 10° for 18 number of teeth for both standard tooth dimension and stub tooth dimension system.
5. It is also observed that the interference volume or TE is more in standard tooth dimension system than stub tooth dimension system.

The work of the paper/outcomes are in line with the work of researchers presented in [3, 4, 6, 12, 14, 17]

6. Conclusions

In this work the consequences occurring due to the changes in gear parameter module on TE for spur gear pair with pitch error, using the interference volume method is presented. The spur gear without pitch error and with pitch error consisting of involute profile for standard tooth dimension and stub tooth dimension systems are modelled and meshed/assembled in SOLIDWORKS software. The interference volume values obtained from the meshing of gear pairs are recorded for one mesh cycle. Various graphs were plotted between the angle of rotation and interference volume values. It is found from the analysis that, as the module increases the TE also increases, TE also increases as the pitch error increase from 0 (no pitch error) to 2%. The effect of variation of module and pitch error is more in the standard tooth dimension system than that in the stub tooth dimension system.

References

[1] Beghini M, Presicce F and Santus C 2004 A method to define profile modification of spur gear and minimize the transmission error *Technical Paper American Gear Manufacturers Association* (Alexandria, Virginia) pp 1-9
[2] Hotait M A and Kahraman A 2013 Experiments on the relationship between the dynamic transmission error and the dynamic stress factor of spur gear pairs *Mech Mach Theory* **70** 116–128
[3] Kohler H and Regan R 1985 The Derivation of Gear Transmission Error from Pitch Error Records *Proc. Inst. Mech. Eng.* Part C **199** 195-01.
[4] Siyu C. and Jinyuan T. 2017. Effects of staggering and pitch error on the dynamic response of a double-helical gear set *J. Vib. Control* **23** 1844-56.
[5] Velex Ph, Chapron M, Fakhfakh H, Bruyère J and Becquerelle S 2016 On transmission errors and profile modifications minimising dynamic tooth loads in multi-mesh gears *J. Sound Vib.* **379** 28–52.
[6] Nina Sainte-Marie 2016 *A transmission-error-based gear dynamic model: Applications to single- and multi-mesh transmissions* (Doctoral dissertation, Université de Lyon)
[7] Sánchez Miryam B, Pleguezuelos M and Pedrero J 2019 Influence of profile modifications on meshing stiffness, load sharing, and transmission error of involute spur gears *Mech Mach Theory* **139** 506–25
[8] Palermo A, Britte L, Janssens K, Mundo D and Desmet W 2018 The measurement of Gear Transmission Error as an NVH indicator: Theoretical discussion and industrial application via low-cost digital encoders to an all-electric vehicle gearbox *Mech Syst Signal Process* **110** 368–89
[9] Shweiki S, Rezayat A, Tamarozzi T and Mundo D 2019 Transmission Error and strain analysis of lightweight gears by using a hybrid FE-analytical gear contact model *Mech Syst Signal Process* **123** 573–90
[10] Zhiying C and Pengfei J 2020 Research on the variation of mesh stiffness and transmission error for spur gear with tooth profile modification and wear fault *Eng. Fail. Anal.* **122** 2020 105184
[11] Pleguezuelos M, Sánchez Miryam B, and Pedrero JoséI 2020 Control of transmission error of high contact ratio spur gears with symmetric profile modifications *Mech Mach Theory* **149** 103839

[12] Padmasolala G., Lin H.H., Oswald F.B. 2000 *Influence of tooth spacing error on gears with and without profile modifications* (NASA/TM-2000-210061 PTG14436)

[13] Bruyere J, Velex P, Guilbert B, and Houser D R 2019 An analytical study on the combination of profile relief and lead crown minimizing transmission error in narrow-faced helical gears *Mech Mach Theory* **136** 224-43

[14] Handschuh M J 2013 *An Investigation into the impact of random spacing errors on static transmission error and root stresses of spur gear pairs* (Master of Science Thesis, The Ohio State University)

[15] Bartosova D, Otipka V and Rehak K 2018 Determination of transmission error in spur gear by numerical approach *Vib. Proced.* **19** 284-88

[16] Park C I 2019 Tooth friction force and transmission error of spur gears due to sliding friction *J. Mech. Sci. Technol* **33** 1311-19

[17] Somani S K 2004 *Transmission Error in Spur Gear* (PhD Thesis, Institute of Engineering & Technology, Devi Ahilya University, Indore)

[18] Bruzzone F, Maggi T, Marcellini C and Rosso C 2021 2D nonlinear and non-Hertzian gear teeth deflection model for static transmission error calculation *Mech Mach Theory* **166** 104471

[19] Duan T, Wei J, Zhang A, Xu Z and Lim T C 2021 Transmission error investigation of gearbox using rigid-flexible coupling dynamic model: Theoretical analysis and experiments *Mech Mach Theory* **157** 104213

[20] Benaïcha Y, Joël P-L, Beley J-D, Rigaud E, Thouverez F 2021 On a flexible multibody modelling approach using FE-based contact formulation for describing gear transmission *Mech Mach Theory* **167** 104505.

[21] Ambaye, G.A. and Lemu, H.G. 2020 Effect of backlash on transmission error and time varying mesh stiffness *Proc. Int. Work. of Advanced Manufacturing and Automation* (Springer, Singapore) pp18-28.

[22] Tao L, Tian D, Tang S, Wu X and Li B 2021 Dynamical modelling and simulation of spur gears with flank pitch error *Res. Sq.* **1** (*Preprint* rs.3.rs-534359/v1)

[23] Pleguezuelos M, Sánchez M B and Pedrero J I 2021 Analytical model for meshing stiffness, load sharing, and transmission error for spur gears with profile modification under non-nominal load conditions *Appl. Math. Model.* **97** 344–65

[24] Chin Z Y, Smith W A, Borghesani P, Randall R B and Peng Z 2021 Absolute transmission error: A simple new tool for assessing gear wear *Mech Syst Signal Process* **146** 107070

[25] Karma V K, Maheshwari G and Somani S K 2020 Analysis of effects of pressure angle and pitch error variation on transmission error in spur gear using interference volume method *Int. J. Adv. Eng. Sci. Technol.*15-22

[26] Maitra Gitin M 2001 *Handbook of Gear Design* 2nd Edn (Tata McGraw Hill Publishing Company Limited New Delhi)

Automated Software Engineering

Source Code Change Analysis with Deep learning based programming model

--Manuscript Draft--

Manuscript Number:	AUSE-D-21-00066R1
Full Title:	Source Code Change Analysis with Deep learning based programming model
Article Type:	S.I. : Software Engineering for Parallel Programming
Keywords:	Change Impact Analysis, Abstract syntax tree, Path2Vec, Deep learning, word embedding, distance metrics, attention, un-parsing.
Corresponding Author:	Babita Pathik, Phd Scholar DAVV: Devi Ahilya Vishwavidyalaya Indore, INDIA
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	DAVV: Devi Ahilya Vishwavidyalaya
Corresponding Author's Secondary Institution:	
First Author:	Babita Pathik, Phd Scholar
First Author Secondary Information:	
Order of Authors:	Babita Pathik, Phd Scholar
	Meena Sharma, PhD
Order of Authors Secondary Information:	
Funding Information:	

Reviewer #1:

1. **In section 4, paragraph 3, the authors explained the flow of their process in four parts. Instead of writing steps as 'one', 'two' etc., they should give a number or can write step 1, step 2, etc., for clarity.**

   Response: *Done*

2. **In figure 4, two parts (a) and (b) are there, but the authors did not consider them separately in the paragraph. It is suggested to mention them in the text.**

   Response: *Done*

3. **ALGORITHM 1 and ALGORITHM 2 caption is missing. Give proper caption based on the task of these algorithms.**

   Response: *Captions are added*

4. **Some references are not in the prescribed format. The formatting of all the references should be uniformed.**

   Response: *Reference formatting is Done*

5. **The author should add some latest works of literature in the paragraphs of the introduction section, especially from 2019-20.**

   Response: *Latest research papers are added.*

6. **Sub-heading 5.2 'Results' should be renamed according to the experimental evaluation like the other paragraph's heading. That looks appealing.**

   Response: *Done*

7. **I recommend the authors to proofread the manuscript with a native English speaker.**

   Response: *Done*

8. **Discuss the comparison of the accuracy of the relevant studies**

   Response: *A comparative table is added.*

Review #2:

1. **Add some more references in support of the work in the literature review section. Briefly explain these references in that section that will show the novelty of your work.**

   Response: *Latest references are added along with the limitations of existing work in tabular form.*

2. **The meaning of variables is not clear. Readers will be confused. The authors should explain them properly.**

   Response: *Detailed explanation is added.*

3. **Write the caption and number separately for figure 11, as it seems the caption is missing.**
   Response: *Done*

4. **Authors have used abbreviations in some places and their full form somewhere. Once the abbreviation is declared, use it throughout the paper where it is required.**

   Response: *Correction is done.*

5. **Discriminate the use of LSTM and Bi-LSTM for the proposed approach.**

   Response: *Done*

6. **In figure 3, a very small code segment has been taken; why? Give reason for that.**

   Response: *The proposed model is also applicable to large program code, but we portray results on small code segments due to the large space requirement for graphical representation of ASTs.*

7. **The conclusion part should include result-specific data or focus on the outcome of work done.**

   Response: *The results are included in the conclusion part.*

Review #3:

1. **The drawbacks of conventional techniques should be described clearly. The authors should emphasize the difference with other methods to clarify the position of this work further. The author can prepare a table that will help to understand.**

   Response: *Information is added in a tabular form. Refer Table 1.*

2. **The attention mechanism should be explained in more detail, supporting your work. A separate sub-section can be added in the main section.**

   Response: A *separate section is added explaining the Attention mechanism (section. 4.6.1)*

3. **What are the values for tuning Hyperparameter? Specify the values for the parameters that are mentioned in the last paragraph of section 5?**

   Response: *Hyperparameter values are mentioned in the new paragraph.*

4. **In figure 6, the description of the node's type is missing. It will better to use a table in place of the figure to describe the nodes. For example, ClassDef → ?????.**

   Response: *The description of Figure 6 is shown in the form of a table. (refer Table 2)*

5. **In section 3, subsection 3.1 abbreviation EIS is not described before. Although before using the abbreviation, its description must be written.**

   Response: *Done*

6. **The term IABLSTM is missing from the abstract or title. If it is significant in this paper, it should be added in the abstract part. The authors should write its expanded form also for more clarity.**

   Response: *Done*

7. **Authors are advised to present a discussion on the accuracy of similar studies.**

   Response: *A detailed description is added in the form of a table. (refer Table 5)*

Reviewer #4:

1. **In this manuscript, the contributor has not emphasized how the Deep Learning Framework is utilized to handle industrial manufacturing. A deep learning framework is provided in diagram 1, but its incorporation with the testing and its relevant information is missing.**

   Response: *Thanks for your suggestion. Deep Learning Framework is fit for industrial manufacturing for a reason dealing and manufacturing substantial data. This is useful where these data are used efficiently. As far as our paper is concerned, the results of the framework can be considered for regression testing, but we have not extended it in this article. We have focused on finding code changes and their impacts.*

2. **How the two purposes achieve it with the loss function is completely loaded with the equations. Therefore, it is not easy to follow. Applying the operational flow of approach Path2Vec provided in a diagram appears well, but its implementation in this manuscript and its functional description are not justified.**

   Response: *Done*

3. **Moreover, the usage of the algorithm along with the data by several processes they are general information storage, implementation of the user interface, privacy design, and support design and its necessary information, are not presented well.**

   Response: *The proposed algorithm is specifically covering the methodology to achieve our objective of change detection. We have not covered the above mentioned area of software products in this article.*

4. **Besides, there is no clear description provided for how the Effective Inspection System for the smart manufacturing industry and its discussions are not considered in this manuscript; therefore, it deficits in studying the central theme of this work.**

   Response: *The idea presented in this article will facilitate the software maintenance process in the information technology industry. Software engineers and developers also take help for regression testing.*

5. **In the result analysis, how the Receiver Path2Vec is used to evaluate the rate and its necessary information is not provided. Furthermore, how the blob, edge, corner, and spots are depicted in diagram 8 are not elucidated.**

   Response: *As two additional diagrams are added as per review, so Diagram 8 has become diagram 10. It is updated with the necessary information. In this presented work, IABLSTM is evaluated as a whole on accuracy rather than Path2Vec separately. Rate is not considered as evaluation.*

6. **The conclusion does not summarize how the images are captured to analyze the manufacturing system's deficits efficiently.**

   Response: *This review seems to be irrelevant with respect to the presented work as we have neither captured any images and nor analyzed any manufacturing system.*

Review #6

1. **To support the novelty of the presented work, a clear comparative study is required for a crisp conclusion. Need to refer to more recent papers to fulfill the said task.**

   Response: *Recent papers are added (2019, 2020). A table is added showing the limitations of various existing methods.*

2. **Hyper-parameter tuning is specifically not mentioned. A detailed description is equired for more clarity.**

   Response: *Detailed description about Hyper-parameter tuning is added.*

3. **A comparative analysis based on the performance of the proposed model must be provided by the authors.**

   Response: *A table is added for comparative analysis of presented work with existing methods based on performance.*

4. **In fig.2 LSTM network is missing from the overall working flow, even though it is a part of the proposed approach, that should be included in the given figure.**

   Response: *Figure 2 is modified with said content.*

5. **The description of variables or the meaning is not given properly. The description gives a better understanding and authors must focus on this point.**

   Response: *All the considered variables are described for understanding.*

6. **Authors have given Figure 1 for the LSTM network. However, the given figure seems to be a standard figure. Authors are required to give an explanation about the figure in the interest of the readers of the journal. As many parameters are given in the figure, authors are required to give a short explanation about them.**

   Response: *The explanation of Figure 1 with all parameters and variables is done*.

7. **In the text authors have mentioned "Figure 1 depicts the overall architecture of the LSTM network". However, this representation could be better in reference to the author's work.**

   Response: *We have replaced the text with a better statement.*

8. **Authors have used multiple abbreviations. Either authors should prepare the table of abbreviations or each abbreviation must be written in full form at its first instance.**

   Response: *All the abbreviations have been elaborated before their use*.

9. **Figure 2 has a caption as "Figure 2. Overall working flow of our approach IABLSTM". I recommend that this caption can be more comprehensive to enlighten the details.**

   Response: *We have updated the caption in detail in Figure 2.*

10. **There are divisions of Figure 4 (a) and Figure 4 (b) and a separate caption is given. However, I recommend that there should be only one caption for Figure 4 where both the captions can be referred using (a) and (b) and similarly figures can be referred through (a) and (b). This is similar to the Figure 3 used in the manuscript.**

    Response: *Only one caption is written for Figure 4(a) and (b)*

11. **Equation 10 is used in the text. But what about other equations? Other equations should also be used in the text.**

    Response: *All the equations (1-13) are used and cited in the text.*

12. **Authors are required to discuss what is BOW, TF, or TF-IDF and how these are used?**

    Response: *We have not evaluated the considered program and dataset on above said technique. We have applied the concept of Word2Vec to proposed a new concept of Path2Vec in this article. We have mentioned the background of all of these terminologies in section 3, 2nd paragraph.*

13. **The authors are required to discuss the available relevant methods.**

    Response: *Done in the form of tables. (refer Table 1)*

14. **The authors are required to compare their results with the current state-of-the-art methods.**

    Response: *Done in the form of tables. (refer Table 5)*

15. **Include more relevant work in the related work section.**

    Response: *Done*

16. **Overall recommendation is to improve the language to increase the readability of the paper.**

    Response: *Done*

Review #7

1. **The difficulties faced in the existing system based on object-oriented based parallel programming source code have not depicted along with their constraints.**

   Response: *Done (refer section 7)***.**

2. **When compared to Path2Vec, how much accuracy is achieved by the conventional models despite of generating abstract syntax tree?**

   Response: *Done (refer Table 5).*

3. What are the characteristics of change impact analysis (CIA) software involved in identifying the changes?

   Response: *Static Code analysis is performed on object-oriented method as granularity level.*

4. **If the software change is implemented after the maintenance, then how do the change impacts the backend and frontend source code and what are the drawbacks that occur due to the impact made on source code?**

   Response: *As we have proposed a methodology for change detection if it occurred and applied it to small code segments. We have also applied our approach to the AST dataset to verify the correctness of the model proposed, but we have not considered or coordinated with backend and frontend source code. Source code change impact analysis is part of the maintenance process, and it definitely affects the related modules (backend and frontend). Added this point in section 8 as future work.*

5. **lack of information concerning the structure of source code tree illustration utilized by the Abstract syntax tree (AST).**

   Response: *Done*

6. **While representing about the contribution of this paper related to the change detection in source code Bi-LSTM is suggested but in abstract section only Path2Vec approach is described what about the Bi-LSTM method?**

   Response: *Done*

7. **The author did not state whether the proposed Word2vec word embedding technique is either a supervised or unsupervised model.**

   Response: *Done*

   Although word embeddings are considered unsupervised, they are trained using a fictitious supervised learning problem.

8. **There is no sufficient information provided for the existing system LSTM and MLP architecture for processing abstract syntax tree.**

   Response: *Done*

9. **How does the tokens sequence is generated for the input data and how does it maps into vector through embedding layer?**

   Response: *Done*

10. **Comparative data between several word embedding techniques BOW, TF-IDF, Word2Vec have not elucidated in this paper.**

    Response: *We have not evaluated above said technique on considered data. We have applied the concept of Word2Vec to proposed a new concept of Path2Vec in this article.*

11. **Working procedure of Bi-LSTM neural network approach is not explained well and seems to be difficult to understand without testimonials. The function of softmax activation layer and its benefits is not represented in this paper.**

    Response: *Explained with separate section.*

Source Code Change Analysis with Deep learning based programming model

Babita Pathik · Meena Sharma

Abstract Analyzing the change in source code is a very crucial activity for object-oriented parallel programming software. This paper suggested an Impact analysis method with Attention BiLSTM (IABLSTM) for detecting the changes and their affected part in the object-oriented software system. Classical approaches based on control flow graph, program dependence analysis, latent dirichlet allocation, and data mining have been used for change impact analysis. A Path2Vec approach is presented in the paper, combining a deep learning technique with word embedding to analyze and identify the change. The paper considers two versions of a python program for experiment and generates the abstract syntax tree (AST). Then extract the path to produce a token sequence. Next, convert the token sequence into unique vectors by applying a word embedding layer. The BiLSTM network encodes the sequence into a vector representation. After that, compare the embedded output with the use of cosine distance metrics. We trained the neural network model with the embedded outcome. Then decode the resultant token sequence into a path of AST. Finally, convert the AST path back to code using the un-parsing technique. To strengthen the parallel programming based proposed model, we combined the attention mechanism to emphasize and detect the differences in the code. The model is detecting the change of code efficiently. The experimental results show that our proposed model's change detection accuracy increases significantly compared with other conventional models for change impact analysis. The proposed method can also be applied for impact analysis on object-oriented based parallel programming. The empirical evaluation shows that the model outperforms change detection with approximately 85% validation accuracy.

Keywords: Change Impact Analysis, Abstract syntax tree, Path2Vec, Deep learning, word embedding, distance metrics, attention, un-parsing.

Babita Pathik (✉)
IT, Institute of Engineering & Technology, DAVV, India
e-mail: babitapathik@gmail.com

Meena Sharma
Department of Computer Engineering, Institute of Engineering & Technology, DAVV, India
e-mail: msharma@ietdavv.edu.in

1 Introduction

Due to the continuous growth of computer based applications and usage, the maintenance of the software has become a crucial task. For reducing the maintenance cost and effort, it is necessary to find the changes and their impact on any part of the code very efficiently. Software change impact analysis enables testers to reduce the maintenance cost by identifying changes and their impact on the code. Once the change is implemented in code, it can affect anywhere any part of the code. This effect is commonly known as the ripple effect. The change in software may cause many side effects. Sometimes it may cause errors too. The object-oriented program comprises influential behavior. The object-oriented paradigms have programming concepts, e.g., polymorphism, inheritance, abstraction and encapsulation. The change analysis techniques take these concrete features of object-oriented programs into account and generate potentially impacted classes, class methods or class fields. In order to help testers and developers, it is reasonably necessary to analyze the changes and their impact on code efficiently and accurately. It may reduce testing costs up to some extent. Change Impact Analysis(CIA) of software is a process of finding changes and their potential impacts on the part of the software system (Bohner et al. 1996).

This paper focuses on finding all the changes in the revised version of the software by evaluating both versions using a parallel programming model. The presented work will facilitate the software maintenance process in the information technology and software industries. The model framework is useful for software engineers to perform regression testing. The granularity of a program may be file level, code level, function level, change level. We choose the code level of the program for an experiment in this paper. The CIA technique suggested by various researchers for static code analysis gives the approximate result. Metrics like McCabe and CK are not sufficient to analyze the basic structure of the code. Traditional methods like software edit history (Kitsu et al. 2013), repositories mining (Moldrez et al. 2017), Control call graph (Badri et al. 2005), Program Dependence Graph (PDG) (Baah et al. 2010), Aspect-Oriented Dependence Flow Graph (Ahmad et al. 2014) considered syntax structure and its relationship among function and classes like elements. But these methods did not consider semantic information instead.

There are some more techniques used by various researchers, such as LDA (Thomas et al. 2010), Latent Semantic Indexing (LSI) (Gethers M. et al. 2011). These improved the performance of code change analysis up to some extent. A deep learning model with a framework of encoder-decoder is applicable in source code generation and its modeling, according to Le TH et al. (2020). Meng and Liu (2020) present a BiLSTM network with a self-attention layer to detect source code.

The syntactic and semantic information are the features that contain such structural and semantic information that may advance the performance of impact analysis. An AST of every program has detailed semantic information and conceives its syntactic structure (Alon et al., 2018, Wang W et al., 2020). A neural network based on AST has been developed (Zhang J. et al., 2019). The change and change impact can be searched and analyzed more accurately with the help of the AST of the program code. The semantic information helps to find the difference between two different versions of the code. Some change does not affect other code. Those set of change is termed as the actual impact set (AIS). That can be traceable by semantic information. Extract all the paths from the generated AST for further process. The paths are extracted as a whole then each of which is separated by splitting them. Each path is converted in vector representation.

Following are the significant contributions in this paper:

(1) We suggest a BiLSTM based change detection that learns useful features related to the source code's semantic and syntactic information. BiLSTM with attention is used for training and named IABLSTM.
(2) We do parsing on a program to evaluate the syntax tree and follow the path of the tree in a depth-first traversal manner. Split the path to generating the tokens.
(3) We apply Path2Vec that uses the Word2vec word embedding technique to convert code’s ASTs into high dimensional real-valued vectors taken as input to the LSTM based model for training.

The rest of the paper is organized as section two briefly summarizes the work achieved by various researchers in CIA and software code analysis. The background of our work is mentioned in section three. The fourth section describes the proposed methodology. The fifth section contains the experimental setup with results, and finally, the sixth section concludes the paper with expected future work.

2 Literature Review

The motivation behind this work is that if changes and their consequences can be detected efficiently, the testing and maintenance costs can be reduced proportionally. It is always a thrust area of research due to the industry's need. Here we brief the work presented by the author, which we have referred to in our work. Table 1 briefs the approaches and threats studied from the above articles. Angerer et al. (2019) presented a CIA approach, which determines the source code's impacted element. They used to control flow and data flow analysis for their approach. Goknil A et al. (2016) utilize semantics of requirements relations, change requests and traces between requirements and architecture to improve CIA in software architecture. Musco V et al. (2016) present LCIP, a learning system that uses historical data to forecast future impacts. The artifacts investigated for CIA are object-oriented software methods. A multi-level word and character embedding were adapted to record the semantics of code modifications and reviews. The embeddings are trained using a suggested attentional deep learning model (Siow JK et al., 2019). Tiwang et al. (2019) suggested a deep learning model for source code generation and completion. AST is processed for structure evaluation of source code. They utilize LSTM, deep learning, and MLP architectures to generate the model.

A learning-based approach is presented for detecting code clones. Code analysis enables the automatic connection at lexical and syntactic levels of patterns mined with a system based on deep learning (White M et al., 2016). Using source code analysis techniques, Eid S et al. (2020) proposes a novel approach to automatically identify probable code changes that result in performance degradation between system versions. A deep learning method is combined with word embedding in a framework for predicting the program's defect (Liang et al. 2019). Token sequence extracted from the generated AST. These tokens are mapped with a real-valued vector using a mapping table. They provided unsupervised training using the word embedding model and LSTM network using vector sequences and labels.

Table 1. Proposed approach and limitations

Author	Proposed Approach	Limitations
Goknil A. et al. (2016)	To identify architectural aspects for the change impacts in architecture requirements, employ the formal semantic of requirement relations and traces between Requirement & Architecture.	Returns the impact on new requirement only if an existing requirement relating to a new requirement exists.
Angerer et al. (2019)	Configuration-aware CIA and interprocedural approach using conditional system dependence graph	The assessment is based on a less customizable code base; some alternative functionalities are not included. It could not be applied to full code.
Musco V et al. (2016)	CIA for object-oriented methods artifacts presented by considering Class-Hierarchy-Analysis call graph	Valid only for Java software, even only for the studied projects.
Siow JK et al.(2019)	Developed a multi-level word and character embedding technique to express the semantics of code modifications and reviews.	Negative data may exist in the training set, while actual positive data is available in the test set. The model is learning some sections of the test set in the training phase.
Eid S et al. (2020)	Identify probable code changes for test cases with a genetic algorithm-based performance tool.	The approach has one basic problem: it cannot identify newly added, instead of updated, source code performance regressions.

Dam HK et al. (2018) designed a prediction model capable of automatically learning and utilizing attributes for representing and predicting defects in source code. The approach is based on a sophisticated deep learning, tree-structured LSTM that corresponds directly to the source code's AST representation. LSTM with an attention mechanism based intelligent model proposed by Rahman et al. (2020) for source code completion. The model classifies the erroneous source code and clean code. The goal of the work is to detect the error in code line by line. Meng and Liu (2020) presented a unique model for detecting source code with a self-attention layer based on a BiLSTM network. The model encodes the series of statement vectors using the model, which is a well-known deep learning framework. The model maintains both the syntactic and semantic properties of the source code during the encoding process.

From the briefing of the literature survey, we can conclude that syntactic and semantic features help source code analysis. The method proposed here is generalized and applicable in any source code except parsing of the code. In this work, we get an AST to extract both the features information. Our model, unlike others, is using BiLSTM with self-attention for detecting the change code.

3 Background

3.1 Change Impact Analysis

Change Impact Analysis is a crucial activity in the software development process due to software evolution. There are so many changes that emerge in software during maintenance. The conception of the CIA is to recognize the changes and their side effects. The impacted part needs to be analyzed effectively to reduce maintenance costs. For large scale software, the CIA is a rigorous task. Assessment of Estimated Impact Set (EIS) is the CIA's primary goal, and it must be as close to AIS. Several researchers suggest various methods to detect impacted parts in source code, including the control call graph and other CIA techniques.

3.2 Abstract Syntax Tree

Abstract Syntax Tree (Büch et al. 2019) is a mode of representing the source code in graphical notations. AST is used by compilers, which reads the code by parsing it and generate the object binaries. It is a tree that represents the abstract syntactic structure of a selected source code. AST completely restores the structural information of the given source code. Each node of the tree corresponds to the essential elements of the code. It efficiently characterizes the programs with any source code and is widely adopted in the software engineering field. AST holds syntactic and semantic information in an exemplary manner and is frequently used by researchers and IT industries. The purpose is to extract hidden information from the code. In this paper, we traverse the tree to process the path.

3.3 Word Embedding

Word embedding is a broadly accepted technique for text in the field of natural language processing. (Hoang et al. 2020). It is a vector representation of words of given documents. Word embedding is mainly a feature extraction technique used in text processing (Hameed et al. 2020). A vector of real value represents each word of the document. We use embedding to encode the token sequence and map it into a vector. Then these vectors are required as input for our model. There is various embedding technique we have gone through, such as BOW, TF-IDF, Word2Vec. We add an embedding layer to our network to extract features.

The BOW model is often employed in the categorization of documents, and each word is used as a training feature for a classifier. BOW doesn't work very well if statements have the same meaning but only with different terms. TF-IDF is a statistic that reflects the importance of a word in a corpus of documents. Word2Vec model is used for vector representations of words known as word embedding. This is done in a preprocessing phase through which the acquired vectors are put into a NN model for predictions and some other task. This model utilizes the semantics of words. We extend the working of the Word2Vec model for our method accordingly.

The purpose of this training is to drag the semantic difference between paths. Embedding layer added as part of a neural network model. The layer is usually built for dealing with natural language processing tasks, specifically classification, language modeling. The document should be preprocessed for encoding. The vector size is specified with a particular dimension such as 50, 100, or 300 and becomes part of the model. One-hot encoded word is mapped to the word vectors through mapping. The network takes the encoded input sequentially.

The widely used embedding method is Word2Vec, an efficient statistical method (Meng and Liu, 2020). Refer the program to as a text document. Skip-gram and Continuous Bag of words (CBOW) are the two architectures on which Word2Vec works. The word w_p is predicted in CBOW, if wp-2, wp-1, wp+1, wp+2 are given context or words. In this work, a token is a path sequence that is a combination of nodes. We adopt the Word2Vec word embedding technique to encode the targeted token. The basic reason for selecting this technique is to retain the order of the words intact. It is the way we can maintain the sequence of the token

3.4 LSTM Network

LSTM is a gated version of RNN (Liang et al., 2019). It is well suitable to process sequential data. LSTM clenches mainly three gates as input gate (i^t), forget gate (f^t), and output gate (o^t). The working of the cell state is transmitting relevant data through a sequence forming a chain. The cell state remembers the material that comes out of the previous time step during the processing in sequences. Data is collected or erased into the cell state via gates as far as the cell state comes into the movement. The neural network-integrated gates determine which cell state data is enabled.

The first gate is the "forget gate," which decides which information should store and discard. The current state input and information from the previously hidden state pass by the sigmoid function. The function returns the value between 0 and 1. If the value it returns is nearer to 0 means inhibiting the information, and if the value is closer to 1, keep and pass all information. We get the value by equation (2).

We've got the input gate to modify the cell state by equation (1). To further control the network, transfer the current input and hidden state output through the *tanh* function, which ranges the output values from -1 to 1. Then the sigmoid result is multiplied with the *tanh* output. To calculate cell state, we received sufficient information. First, the pointwise multiplication is held between the cell state and forgot vector. If values compound it near 0, it drops values in the cell state. Next, take the input gate's output, and pointwise addition is applied, which upgrades the cell state with new values found appropriate by the considered neural network. The state is our new cell state and evaluated using equations (4) and (5).

The output gate is appropriate to determine the next hidden state using equation (3). For predicting the value, the hidden state is having information about previous inputs. First, new input and the previously hidden state transfer into the sigmoid function. Then transfer the recently changed cell state to the *tanh* function. The sigmoid output is multiplied with *tanh* output to determine hidden state data with equation (6). The hidden cell and new cell state are then passing to the subsequent stage. Fig. 1 depicts an internal architecture of the LSTM cell.

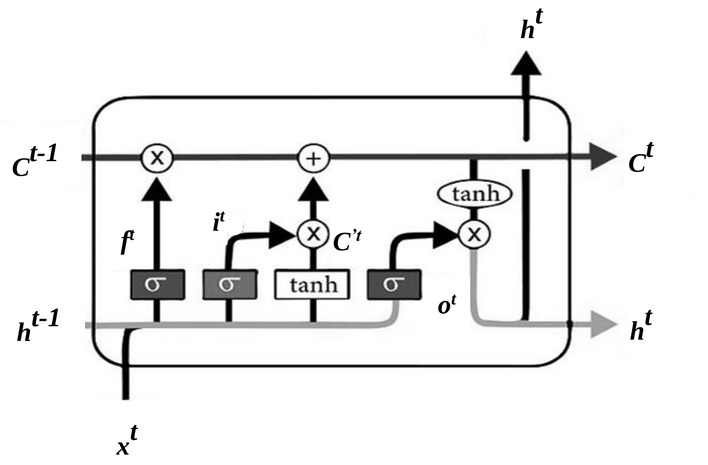


Fig. 1 Internal working of an LSTM cell

The equations for the gates in LSTM are:

Input gate $i^t = \sigma\,(w^i[h^{t-1}, x^t] + b^i)$ (1)

Forget gate $f^t = \sigma\,(w^f[h^{t-1}, x^t] + b^f)$ (2)

Output gate $o^t = \sigma\,(w^o[h^{t-1}, x^t] + b^o)$ (3)

σ = Sigmoid function
$w^i / w^f / w^o$ = Weight for the input/forget/ output gate neuron
h^{t-1} = Output of the previous hidden state is at time *t-1*
x^t = Current state input, i.e., at time-stamp *t*
$b^i / b^f / b^o$ = Biases for the input/forget/ output gates

New Candidate for cell state at time *t* : $C'^t = tanh\,(w^c[h^{t-1}, x^t] + b^c)$ (4)

Cell state at time *t* : $C^t = f^t \times C^{t-1} + i^t * C'^t$ (5)

Final output at *t* : $h^t = o^t \times tanh(C^t)$ (6)

C^{t-1} = Cell state at time *t-1*
w^c = weight for a new candidate
b^c = Bias for a new candidate

With cell state and the gates mentioned above, gratuitous information is automatically dropped by LSTM cell as the time step grows. LSTMs are widely used in software engineering problems (Liang *et al.*, 2019, Meng and Liu, 2020) and natural language processing (Hameed and Garcia, 2020), where semantic relation is essential. In the paper, we train the model using BiLSTM to learn source code.

3.5 BiLSTM

BiLSTM is an advanced form of RNN. These can significantly increase model performance when used to solve sequence classification issues. The model consists of two LSTMs: one that takes the input in one direction and the other in the opposite direction (forward and backward direction). BiLSTM significantly improves the quantity of data presented to the network, which benefits the algorithm's context.

The BiLSTM is designed to achieve the objective of long term dependence at the moment around t. Source code comprises contextual information that is important to spot potential issues. Each program has its own context-sensitive syntax and semantics. The emergence of a different code segment is normally relevant to both preceding or succeeding code. In the implementation of BiLSTM, bidirectional processing runs given inputs in two directions; forward direction leads from past to future, and backward is from future to past. We use this model two combined hidden states; one can preserve information from both the past and future at any point in time.

3.6 Attention Mechanism

We can obtain hidden features of all time nodes in a series from the output of the BiLSTM network. We insert an attention layer after the Bi-LSTM layer in order to improve the influence of crucial nodes. When the attention mechanism is applied, critical nodes that are important for the sequence are aggregated together to produce a sequence vector.

4 Proposed Approach

This section of the paper describes the overall working of our proposed approach. Multiple version of source code is taken for experiment purpose to analyze the change. Our proposed IABLSTM is a system that automatically traces syntactic features and semantic information from the source code through parsing to extract key features to find change code.

Very first, we apply preprocessing on input source codes which involve the removal of comment lines. Our target is to train the model with a different version of the same program with or without error and, after that, successfully use it to detect changes by giving another version of the same program. Fig. 2 demonstrates the proposed method of IABLSTM.

The LSTM based approach has two working parts. The first one is the representative part, and the second is the discriminative part. In the first part, we represent the code in dimensionality vector, which is achieved with the following steps:

Step 1: Both versions' selected python source code is presented into ASTs by utilizing available AST tools. These ASTs generate at the statement level.
Step 2: We perform pre-order traversal on the tree to extract path sequence with a granularity of statement. Now we split the sequence path up to the leaf nodes.
Step 3: According to the deep-first traversal technique, each split path sequence is converted into a real-valued vector with an LSTM encoder's help. All path fragment is transformed into an ordered set of path vectors.
Step 4: The proposed model generates a final characteristic vector from a path vectors sequence through a bidirectional LSTM (BiLSTM) encoder with the self-attention layer. A set of code vectors calculated by applying cosine distance using this model is further loaded into the designated model. The model predicts the probability of change for the pair of codes.

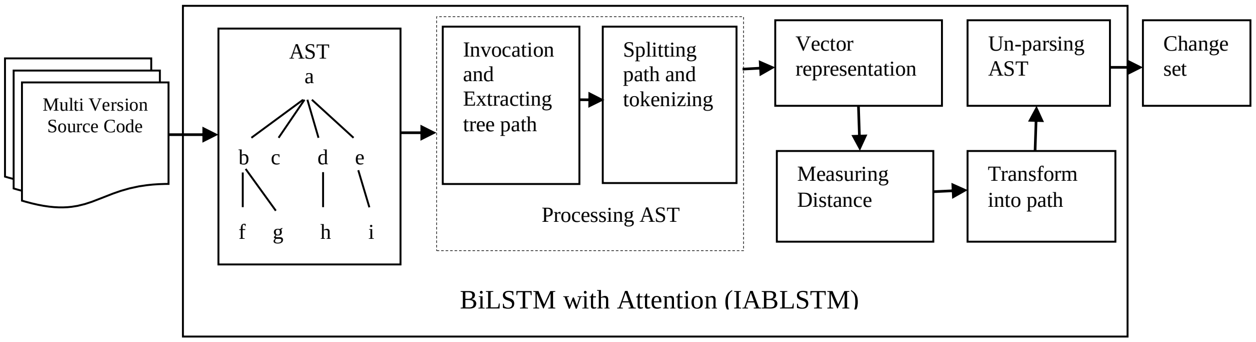

Multi Version Source Code
AST
a
b c d e
f g h i
Invocation and Extracting tree path
Splitting path and tokenizing
Processing AST
Vector representation
Measuring Distance
Transform into path
Un-parsing AST
Change set
BiLSTM with Attention (IABLSTM)

Fig. 2 A process diagram for the proposed IABLSTM approach with AST.

4.1 Code Preprocessing

Before training the model, we filtered raw source codes by removing unnecessary items. First, all irrelevant elements have been eliminated from the code, such as new lines, comments (#), and tabs (\t). Then, every remaining code element like keywords, numbers, functions, variables, classes, and characters have been translated into sequences of terms. The filtered code was parsed through the parser.

4.2 Source Code Parsing

AST is a syntactical structure representation of a code as a tree. It is an intermediate representation of a program during the construction of a compiler. The construction of AST is a part of parsing that is a syntactic analyzer.

```
class EleVehicl(Vehicl):                          class EleVehicl(Vehicl):
    def __init__(self, make, model, type):            def __init__(self, make, model, type):
        self.chrg_lvl = 0                                 self.chrg_lvl = 0
                                                      def charg(self):
                                                          self.chrg_lvl = 100
                                                          print('charged')

                  (a)                                               (b)
```

Fig. 3 Python programs for charging vehicle (a) first version (b) revised version

We generate AST for a different version of a program to extract the various existing path. The semantic analyzer utilizes the information provided by the AST. Every path ends with a leaf node. The leaf node or terminal node contains tokens of the program code.

There are two small python codes given in Fig. 3(a) and 3(b) for the first and second versions, respectively. Fig. 4(a) and 4(b) depict the AST of code given in Fig. 3(a) and 3(b), respectively. The proposed model is also applicable to large program code, but we portray results on small code segments due to the large space requirement for graphical representation of ASTs.

In Fig. 4(a) and 4(b), the nodes of the tree comprise keywords, characters. These fundamental elements of the source code are tokens. The parent-child node pair can also be extracted from AST. A parent-child relationship is valuable when it is required to trace back for analysis. There are various types of nodes that emerged in AST. Each node shows either a code component or a specific characteristic of code. Every tree starts with a module as a root node.

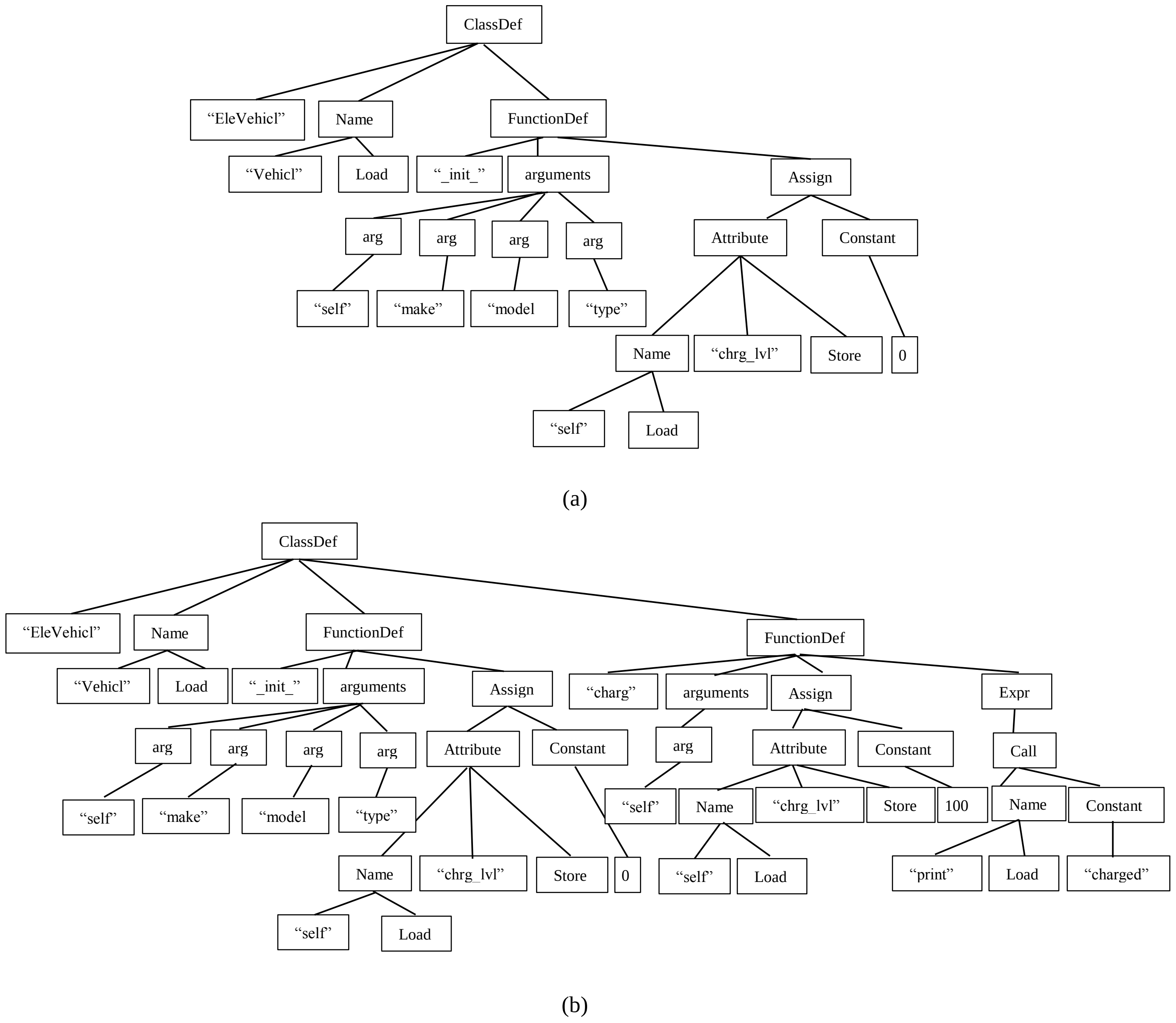


Fig. 4 AST for (a) first version (b) revised version

Fig. 5 enlisted the character hold by the nodes of a tree. The list may vary, here we include some of them.

4.3 Processing of AST

Path extracted from AST is considered for further processing. Visit every node of the tree through pre-order. We traverse the nodes of the tree in the depth-first traversal technique. After traversing, the path is extracted as shown in Figures 6 and 7 for source codes shown in figures 3(a) and 3(b), respectively. It shows the output as a complete string. To convert this string into words, we need to break it down to generate tokens. The paths are separated by splitting with the terminal node, which is having variable path length. These separated paths are tokens that construct a sequence vector. All the nodes, including the terminal node of each truncated path, show the program code's characteristics or code elements. Algorithm 1 in Figure 8 explains the generating and parsing of the source code.

ClassDef	Module	Alias	Store	Store
NameConstant	comprehension	ListComp	Return	Num
FunctionDef	Assign	Slice	Constant	Tuple
Name	Call	For statement	Mult	NotEq
Arguments	str	While statement	AugAssign	Eq
Print	Expr	If statement	List	Mod
BinOp	True	Gt	Attribute	LtE
Load	Subscript	GtE	Sub	Add
Compare	arg	Lt	Div	Index

Fig. 5 Types of nodes in AST

4.4 Path2Vec

All the separate paths transform into a vector representation. Paths are tokens and in the form of strings that cannot be used as input directly. So, first of all, the tokens are converted into the token id. Then the token ids are transformed into a vector. Vectors are of varying length of n number of tokens. The token ids are unique integer values. Vectorization is achieved through the encoding technique by using word embedding.

```
["ClassDef(name='EleVehicl', bases=[Name(id='Vehicl', ctx=Load())],"
 "keywords=[], body=[FunctionDef(name='__init__',"
 "args=arguments(args=[arg(arg='self', annotation=None), arg(arg='make',"
 "annotation=None), arg(arg='model', annotation=None), arg(arg='type',"
 'annotation=None)], vararg=None, kwonlyargs=[], kw_defaults=[],   kwarg=None,'
 "defaults=[]), body=[Assign(targets=[Attribute(value=Name(id='self',"
 "ctx=Load()), attr='charg_level', ctx=Store())], value=Num(n=0))],"
 'decorator_list=[], returns=None)], decorator_list=[])']
```

Fig. 6 Extracted path for code in Fig. 3(a)

```
["ClassDef(name='EleVehicl', bases=[Name(id='Vehicl', ctx=Load())],"
 "keywords=[], body=[FunctionDef(name='__init__',"
 "args=arguments(args=[arg(arg='self', annotation=None), arg(arg='make',"
 "annotation=None), arg(arg='model', annotation=None), arg(arg='type',"
 'annotation=None)], vararg=None, kwonlyargs=[], kw_defaults=[], kwarg=None, '
 "defaults=[]), body=[Assign(targets=[Attribute(value=Name(id='self',"
 "ctx=Load()), attr='charg_level', ctx=Store())], value=Num(n=0))],"
 "decorator_list=[], returns=None), FunctionDef(name='charg',"
 "args=arguments(args=[arg(arg='self', annotation=None)], vararg=None, "
 'kwonlyargs=[], kw_defaults=[], kwarg=None, defaults=[]),'
 "body=[Assign(targets=[Attribute(value=Name(id='self', ctx=Load()),"
 "attr='charg_level', ctx=Store())], value=Num(n=100)),"
 "Expr(value=Call(func=Name(id='print', ctx=Load()), args = [Str (s = 'charged')],"
 'keywords=[]))], decorator_list=[], returns=None)], decorator_list=[])']
```

Fig. 7 Extracted path for code in Fig. 3(b)

The description of the tree component as nodes are given in Table 2.

Table 2. Description about tree components

Nodes	Description
ClassDef	Class definition
args	arguments
FunctionDef	Function definition
vararg	Variable argument
attr	attribute
kwonlyargs	lists of arg nodes
kw_defaults	Default keywords
Expr	Expression
Str	String
kwarg	passed arguments by keyword
ctx	Store an assignment
Fun	function

```
Algorithm 1:
Input: Two Version Source Codes $S = \{S_i, S_{i+1}\}$
       Set of featuring nodes $F = \{f_1, f_2, f_3, ...f_n\}$
Output: Path List $P = \{P_i, P_{i+1}\}$
        Vector $V = \{V_i, V_{i+1}\}$ for $\{S_i, S_{i+1}\}$
for all source files do
  for $i = 1$ to $n$ do
    $AST_i$ = Generating $AST$ from $S_i$
    Visiting $ASTs$ each node $F$ by depth first manner
    Accumulate $P_i = \{f_{i1}, f_{i2}, f_{i3}, ...f_{in}\}$
  end for
end for
for each $P$ do
  Splitting $P_i$ as Tokens $\longrightarrow \{P_{i1}, P_{i2}, P_{i3}, \ldots, P_{in}\}$
  Adding each Token to $V$
  return $V$
end for
```

Fig. 8 Algorithm 1 for Parsing the files and vector representation of AST

The token id is represented in a real-valued vector through the Word2Vec embedding technique. Dealing with the paths and converting them into a vector is termed as Path2Vec. For example, '*ctx=Load()*' is a token mapped to some integer value and emerged in an array of vector V_p. In this way, the vector sequence is generated as $v_1, v_2, v_3, v_4, \ldots\ldots v_n$ and given as input to the network. V_p and V_c are the vectors of previous code and current code, respectively. The hidden state outcome to form a new state as:

$$h^t = LSTM(V_p),\ t = \{1..n\} \tag{7}$$

$$h^t = LSTM(V_c),\ t = \{1..n\} \tag{8}$$

V_p = Vector for previous code
V_c = Vector for current code
h^t = hidden state outcome for both code

4.5 Change Detection

We apply the cosine distance metric for searching the changes that occurred in the revised version of the code. The cosine distance metric formula is the dot product of two attributes. It is the measurement of the cosine angle of two vectors. The outcome of the equation is a normalized value. Resultant values of the metric laid between 0 to 1. The idea comes from determining the angle between the two objects. Researchers extensively use the cosine distance metric to search for the similarities between two components. In this paper, we use the metric to observe the dissimilar part of a program to detect the imposed changes. Cosine similarity is widespread because it is efficiently evaluated on vectors, significantly on sparse vectors. It determines how much two vectors are similar or dissimilar irrespective of their size. The vectors are path sequence embedded vectors in our context. Cosine similarity is much helpful for cases when there are duplicate data matters. Analyzing text similarity is an NLP-based application of the metric. The formula for cosine metric is given as equation (9).

$$\text{dis}(V, V') = \cos(\theta) = \frac{V \cdot V'}{\|V\|\|V'\|} = \frac{\sum_{i=1}^{n} V_i\ V_i'}{\sqrt{\sum_{i=1}^{n} V_i^2}\sqrt{\sum_{i=1}^{n} V_i'^2}} \tag{9}$$

where $v \cdot v' = \sum_1^n v_i \cdot v_i' = v_1v_1' + v_2v_2' + v_3v_3' \ldots + v_nv_n'$ = two vector's dot product.

In equation (9), dis (V, V') is the difference between two vectors. Here V and V' are path vectors of the program's previous version and the current version.

dis (V, V') = 1 if V = V'

dis (V, V') = 0 or < 1 if V ≠ V'

Algorithm 2 in Figure. 9 describes the pseudo convention for vectorization and difference measurement of ASTs. The P_i and P_{i+1} are the two path tokens of some length *m* we get from Algorithm 1. These vectors held the unique ID of the tokens such as:

$P_i = \{t_i^1, t_i^2, t_i^3, \ldots\ldots\ldots t_i^m\}$

$P_{i+1} = \{t_{i+1}^1, t_{i+1}^2, t_{i+1}^3, \ldots\ldots\ldots t_{i+1}^m\}$

t_i^m = mth token of ith path

t_{i+1}^m = mth token of $(i+1)$th path

4.6 Neural Network Model for Difference Measurement

After completing the embedding and tokenization process, we trained our proposed models and other associated, cutting-edge models with the past versions of source codes for *Vehicle* problems. The next move is to review the model's output on the code variant detection task at the end of the training process. How correctly do the differences are identified, and changes are predicted?

Our proposed model is trained with BiLSTM. Supervised learning gives better performance than unsupervised learning.

4.6.1 Attention with BiLSTM

The attention mechanism with BiLSTM enhances the hidden feature of nodes in sequence. Figure 10 depicts the process of attention mechanism. The intention of incorporating the attention mechanism into BiLSTM is to strengthen our model and predict long sequences of source codes. So an attention layer is embedded with the BiLSTM layer. The attention layer aggregates all sequences and from a sequence vector. Merging all the hidden layer output and giving the attention function improves the performance of the model. The BiLSTM is used here to generate a sequence of annotations (h_1, h_2, ….., h_n) for each input sentence. All the vectors h_1, h_2, .., etc., used in work are mainly the interconnection of forward and backward hidden states in the network as given in equation (10).

$$h_j = \left[\overrightarrow{h}_j^T ; \overleftarrow{h}_j^T\right]^T \tag{10}$$

h_j = annotation sequence

```
Algorithm 2:
Input: ASTs' path vectors $P_i, P_{i+1}$, the fixed length of each vector is m;
Output: distance vector (diff_list), sent;
Initialize a list V, dictionary tokenID;
for i=1 to n do
   for j = 1 to $len(P_i)$ do
      if $split(P_i)$ is TRUE
         $tokenID = int\_value(p_i^j)$ ;
         $append(tokenID)$;
      end if
   end for
end for
//Creating a list of tokens for both program separately on the basis of term frequency:
$P_{vect1}$ and $P_{vect2}$
for i = 1 to $len(P_{vect1}[1 .. len(P_i)])$ & $len(P_{vect2}[1..len(P_{i+1})])$ do
     for j = 1 to i do
        for k= $len(j)$ to $len(max\_length)$ do   //max_length is for adjusting the max dimension
              $P_{vect1}$ [k] = 0;
           end for
      end for
end for
for i = 1 to $len(P_i)$ & $len(P_{i+1})$ do
     $cosine(P_{vect1}, P_{vect2})$
     if $cosine == 1$
          $continue$
     else
          $append(diff\_list[P_{vect2}])$
     end if
end for
for i = 1 to $len(diff\_list)$ do
     $out = invert\_decode(i)$
     $sent = unparse(out)$
end for
return diff_list, sent
```

Fig. 9 Algorithm 2 for vectorization and difference measurement

We input the annotation h_j to the multilayer perceptron to generate a hidden representation a_t by the equation (11) and depicted in Figure 10.

$$a_t = tanh\,(W_n\, h_j + \; b_n) \tag{11}$$

Here W_n is new weight and b_n is a new bias.

Furthermore, we measured cross-entropy at the softmax layer for every epoch to evaluate the model loss function. The difference between the actual and predicted results is referred to as cross-entropy. The softmax function is used to normalized the weight at attention. Softmax is typically used as the final layer of neural networks. The softmax function's performance range is between 0 and 1.

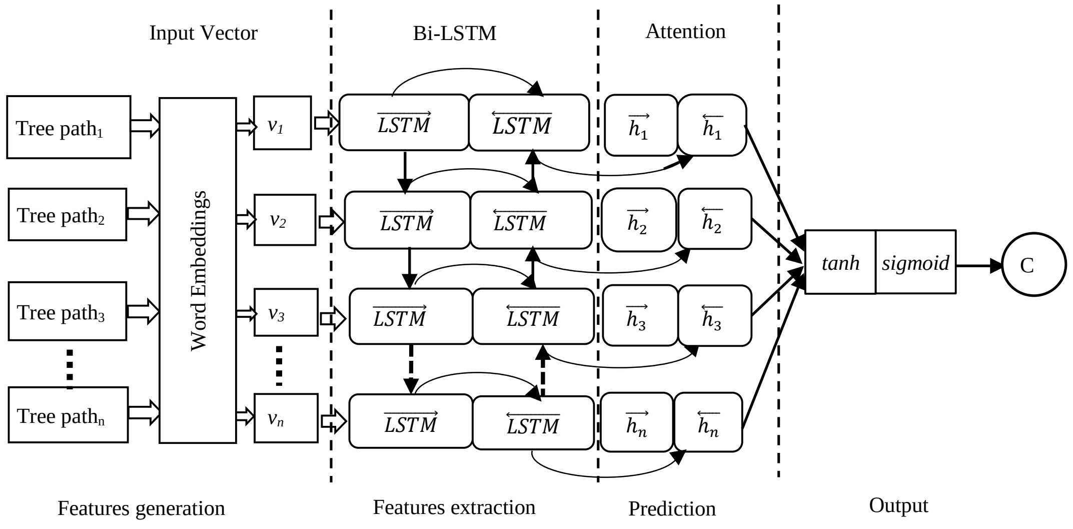


Fig. 10 BiLSTM Model with self-attention

The input to the softmax layer is context vector $x \;= [\,x_1, x_2, x_3, \dots \dots . . x_n]$ multiplied with the output a_t and returned normalized weights $p \;=\; [\,p_1, p_2, p_3, \dots \dots . p_n]$, that can be defined as in equation(12) :

$$p_i = \frac{exp(a_t x_i)}{\sum_{j=1}^{k} exp(a_t x_j)} \tag{12}$$

Finally, we construct the sequence vector by summing all the nodes with corresponding weights using equation (13). The context vector at the node level is randomly initialized and updated during training.

$$s_i \;= \sum_j p_i\, h_j \tag{13}$$

s_i = weighted sum

5 Experimental Setup and Results

5.1 Dataset and System Selection

There are barely any datasets available for code change detection, which gives specifications related to our concern, and it's quite challenging to identify change sets. The empirical setup is done on 150k Python Dataset [sri.inf.ethz.ch/py150]. We select an AST file python100k_train for around 2GB in size and JSON format for training and python100k_test for testing. The file contains AST of 100,000 python programs.

These python programs are gathered from the GitHub repository. The programs those AST have a maximum of 30,000 nodes are included in these repositories. According to the system's capacity, we further divide this big-sized JSON file into small chunks with the help of a JSON splitter. Tensorflow with Keras using Python development environment on google cloud. Windows 10, 8GB RAM, Intel Core i7,4690T central processing unit, 2.50GHz, 64-bit OS, x64 based processor.

5.2 AST Encoding

ASTs are capable of storing both the structural as well as semantic information about a program module. Because the vector is composed of path tokens, it cannot be used directly as an input to IABLSTM. As a result, we create a dictionary for mapping tokens to integers. If there are m tokens and each token relates to a distinct integer, the mapping range is 1 to m. To begin, we count the tokens' frequency and then arrange them according to their frequency. Next, we create an indexed dictionary for the ordered tokens, with the most often occurring tokens at the top. Afterward, in the mapping stage, we equalize the length of these numeric vectors. A length of the vector should be chosen with a size less than the required length is denoted by 0 because 0 has no meaning when tokens are mapped starting at 1. If the length of a vector exceeds the required length, the additional component is truncated. Since the tokens with a greater frequency are translated to a smaller integer, the tokens with a lower frequency are mapped to the largest integer. As a result, we find the index of the largest integer in the vector and remove it one by one until the vector length equals the set length. Figure 11 describes the processing of the AST.

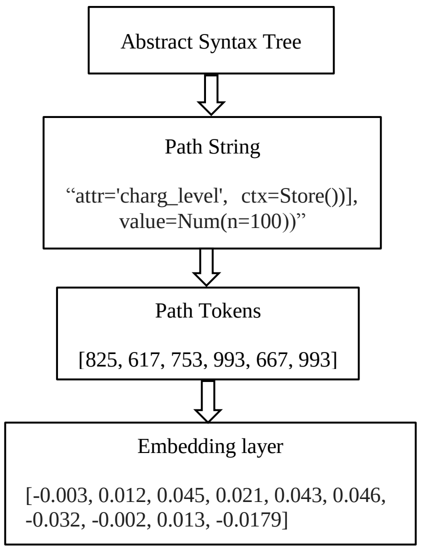


Fig. 11 Processing of AST with embedding layer

Finally, we apply word embedding for high-dimensional vector representation of each token using a trainable word dictionary embedded in the network. Although word embeddings are considered unsupervised, they are trained using a fictitious supervised learning problem.

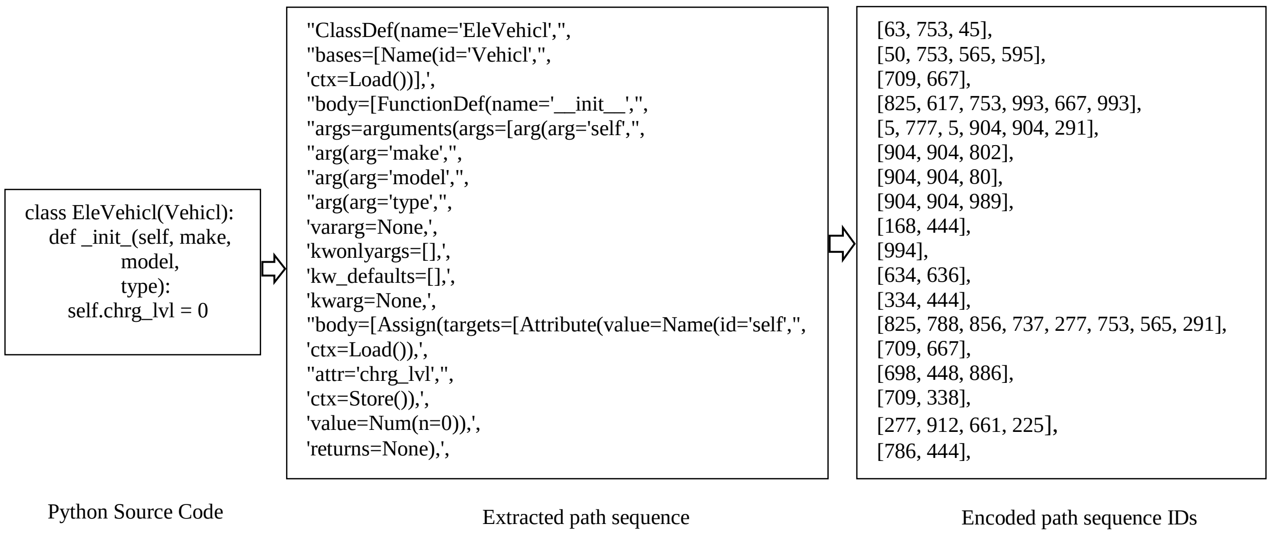
5.3 Path Extraction and Vectorization

The experiment is initially performed on a small dataset with various python codes with multiple entries, e.g., the category of program, type of file, file name, and program's status. The program's status has entries regarding the change or no change and the program itself. We first fetched the program its status according to the program category and file type. In the next step, we preprocessed all these programs and removed the comment lines. Next, we generated an AST and then got a path with pre-order traversal from it. Table 3 briefs the various kinds of tokens extracted from the tree's path and their corresponding integer value. We have enlisted some of them in the table. The table with the tokens column contains various parts of the tree we got by traversing the pre-ordered manner and tokenizing it. Another column of the table includes the token id of these tokens.

Figure 12 portrays an example code for this paper, then we extracted the abstract tree and got the sequence path. Next, we tokenize the path sequence and generate tokens, and then we converted these tokens into identifiers and got token IDs. Every token itself has various tokens.

Table 3 Tokens and its corresponding ID

Tokens	Integer value
ctx=Load()	[709, 667]
body=[FunctionDef(name='__init__')]	[825, 617, 753, 993, 667, 993]
ctx=Store()	[709, 338]
keywords=[]	[958]
annotation=None	[693, 444]
vararg=None	[168, 444]
decorator_list=[]	[206, 370]
kwonlyargs=[]	[994]
returns=None	[786, 444]
kw_defaults=[]	[634, 636]
kwarg=None	[334, 444]
defaults=[]	[636]



Python Source Code

```
class EleVehicl(Vehicl):
   def _init_(self, make,
              model,
              type):
      self.chrg_lvl = 0
```

Extracted path sequence

```
"ClassDef(name='EleVehicl',",
"bases=[Name(id='Vehicl',",
'ctx=Load())],',
"body=[FunctionDef(name='__init__',",
"args=arguments(args=[arg(arg='self',",
"arg(arg='make',",
"arg(arg='model',",
"arg(arg='type',",
'vararg=None,',
'kwonlyargs=[],',
'kw_defaults=[],',
'kwarg=None,',
"body=[Assign(targets=[Attribute(value=Name(id='self',",
'ctx=Load()),',
"attr='chrg_lvl',",
'ctx=Store()),',
'value=Num(n=0)),',
'returns=None),',
```

Encoded path sequence IDs

```
[63, 753, 45],
[50, 753, 565, 595],
[709, 667],
[825, 617, 753, 993, 667, 993],
[5, 777, 5, 904, 904, 291],
[904, 904, 802],
[904, 904, 80],
[904, 904, 989],
[168, 444],
[994],
[634, 636],
[334, 444],
[825, 788, 856, 737, 277, 753, 565, 291],
[709, 667],
[698, 448, 886],
[709, 338],
[277, 912, 661, 225],
[786, 444],
```

Fig. 12 Example python code and Tokenization and Encoding Process

5.4 Distance Measurement

We calculated the cosine distance and found the dissimilar part of the code. The cosine metrics give better results on a sequence of the vector. Cosine metrics take the total length of the vectors. These vectors are prepared from BOW, TF, or TF-IDF. Through Figure 13(b), the probable changes are marked in a different color that has been integrated into the code given in 13(a). This python code is for showing the charging level of a vehicle. We have taken the code just for demonstration purposes. In the first code, there is a class defined for an electric vehicle with an initialization function. The revised version has one new *charg* function added to it, and the rest of the code is as same as the previous one. The function sets the charging level to 100. The code from keyword '*def*' to the symbol '*)*' is newly included syntax into the same existing program. The model identified these codes with the highest probability.

5.5 Training and Validation

We evaluated both the model BiLSTM and IABLSTM on the chosen python program of charging the vehicle's battery and Python 150k dataset. The model accuracy and cross-entropy are portrayed with the help of the pyplot library.

```
class EleVehicl(Vehicl):
    def __init__(self, make, model, type):
        self.chrg_lvl = 0
```

(a)

```
class EleVehicl(Vehicl):
    def __init__(self, make, model, type):
        self.chrg_lvl = 0
    def charg(self):
        self.chrg_lvl = 100
        print('charged')
```

(b)

Fig. 13 Previous version (a) current version with change part (b)

The following figures are showing the results of empirical evaluation sequentially. The accuracy and cross-entropy are evaluated on the selected AST dataset for the BiLSTM model, as depicted in Figure 14 and Figure 15, respectively. The training accuracy goes up to 85%, and validation accuracy has grown up to 75%. Likewise, training cross-entropy is down to 35%, whereas validation cross-entropy goes down to 56%.

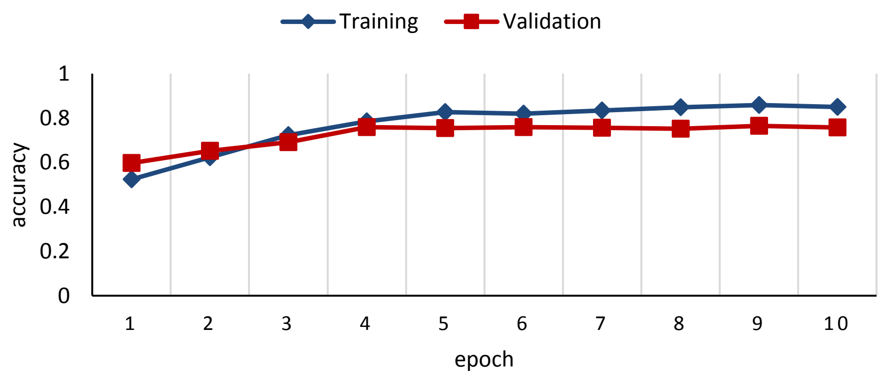
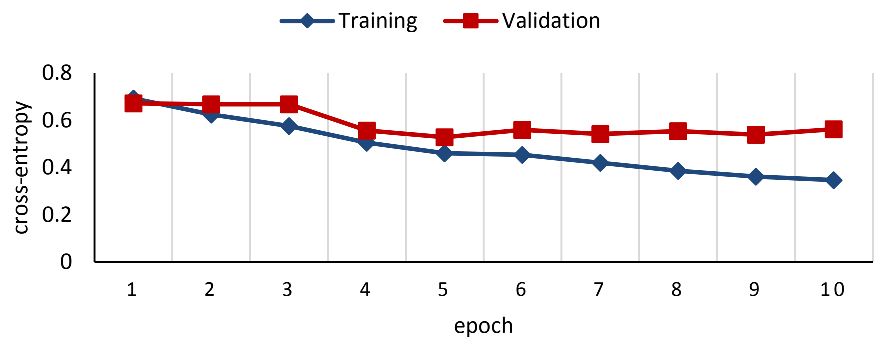


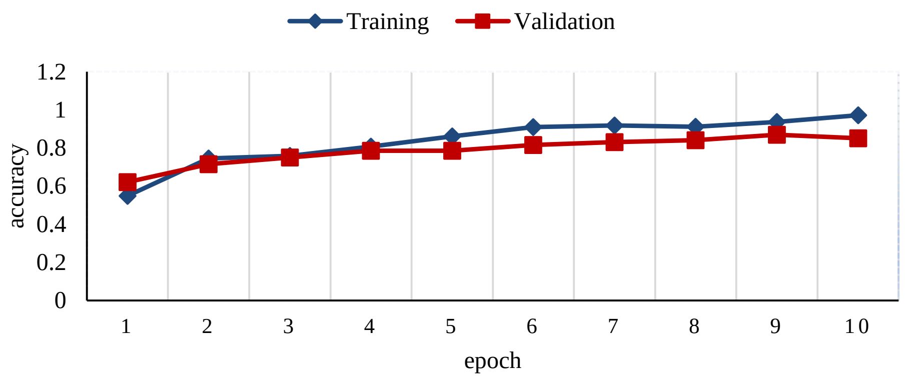
Fig. 14 Accuracy plot for Python150k using BiLSTM



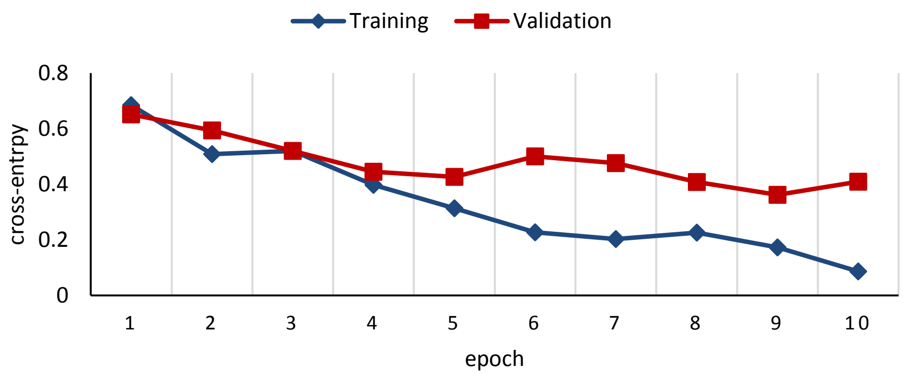
(b)

Fig. 15 Cross-entropy plot for Python150k using BiLSTM

Figures 16(a) and (b) showing the accuracy and cross-entropy evaluated on the same dataset for our model IABLSTM. The training accuracy uplifted to 97%, and validation improved up to 85%. The model training loss comes down to 8%, and validation loss goes down to 40%.



(a)



(b)

Fig. 16 Change detection Model for Python150k using IABLSTM (a) accuracy (b) cross-entropy

The accuracy analysis of both the model is shown in Figure 17.

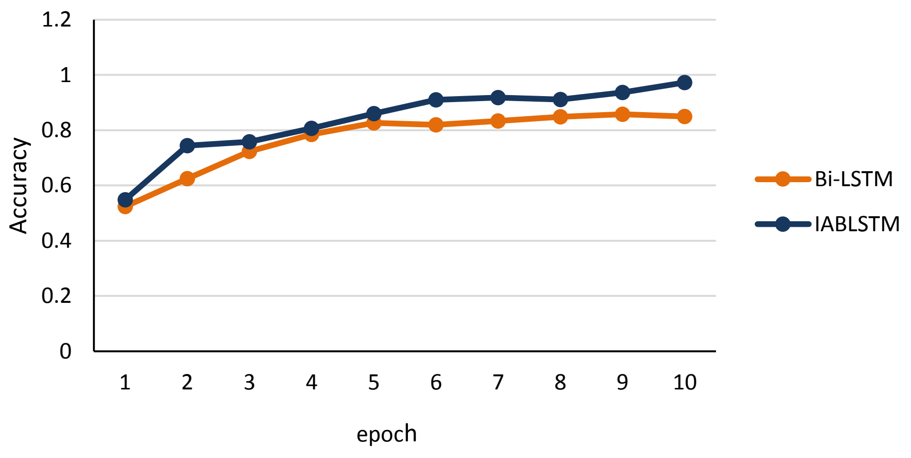


Fig. 17 Epochs wise accuracy analysis of both models on Python 150k

5.6 Hyperparameter

We specified numerous hyperparameter for the experimental purpose to improve our results in this study. To avoid overfitting, we proposed a dropout ratio of 0.25 for our proposed model. Adam optimizer is used in the LSTM network. The learning rate is critical for the training of neural networks since it may be used to modify the model's learning speed. The learning of the network becomes slower or faster as the value of the learning rate decreases or increases. This work determines the learning rate (*l*) 0.002, and during training, the network weights are updated with the value of *l*. We take input length 100, the number of epoch 20 and the number of hidden units 200.

6 Results and Discussion

Table 4. gives a comparative analysis among the various neural network based LSTM, BiLSTM, IABLSTM models. We applied all these models on considered example code, EleVehicl and Python 150k dataset.

Table 4. Accuracy and cross-entropy comparison on different NN models on example source code

	EleVehicl		Python 150k	
Model	Cross-entropy	Accuracy	Cross-entropy	Accuracy
LSTM	0.6772	0.5434	0.6810	0.5682
BiLSTM	0.5459	0.6597	0.3459	0.8497
IABLSTM	0.2693	0.8056	0.0859	0.9721

We have compared cross-entropies of all the models by evaluating them on different data and shown in Figure 18.

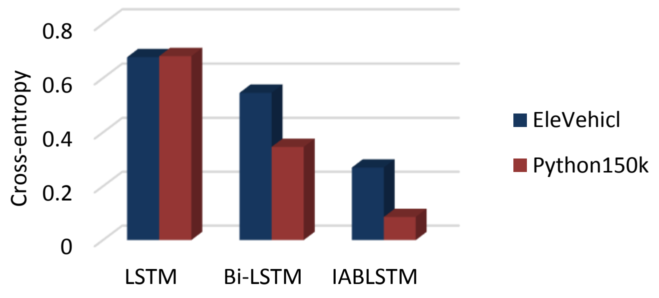


Fig. 18 Comparison on cross-entropies of various models for codes

Figure 19 displays the chart for training accuracy observed on various models, which we mentioned in the paper.

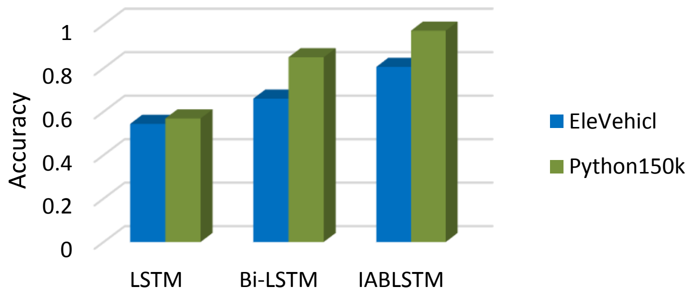


Fig. 19 Comparison on accuracies of various models for example codes

The figure showing the comparative analysis of BiLSTM and IABLSTM that stated using BiLSTM with self-attention improves the accuracy up to 11% to detect the code change compared to simple BiLSTM.

Our IABLSTM model outperforms other state-of-art that uses AST. Table 5 enlisted the results in terms of accuracy. The comparative analysis is performed based on the objective achieved, and the method opted by the authors.

Table 5. Comparison on Accuracy

Objective	Methods	Results(Accuracy/ Precision)
Change impact prediction	LCIP	74%
Code change extraction	CC2Vec	90.7%
Defect prediction	AST+LSTM	90%
Code clone detection	AST+GMMN	96%
Learning program properties	AST path+Word2Vec(JavaScript)	69.1%
Code clone detection and classification	ASTNN	**98.2%**
Code clone detection	AST+ Attention BiLSTM	96.8%
Code change detection	Path2Vec + IABLSTM	**97.2%**

The methods included in given Table 5. have considered AST as source code parsing in most of the works. We also generated AST and processed it through Path2Vec, achieving 28% better performance than the Word2Vec approach.

7 Threats to validity

There are some threats to validity stated here. The Python150k dataset taken for experimental purposes is easily downloadable but not easy to handle completely due to its length. The dataset can be breakdown into small sizes for its usefulness. Another threat we faced is system configuration constraints while evaluating object-oriented based parallel programming code. The exact amount of change is hard to find, although we have achieved accuracy up to the mark.

8 Conclusion and Future Work

In this work, we have presented an approach for source code change detection using deep learning based BiLSTM with self-attention IABLSTM. BiLSTM with attention is used to acquire the semantic as well as syntactic information together of the input code in the encoding process. The results improved up to 85% in terms of accuracy to find the actual impact set compared to other models. The implemented model gives 11% more accuracy than the BiLSTM model. We generated the syntax tree and traversed the tree to extract the said information using a novel approach of parallel programming for change detection. Then the path is transformed into the vector using a vectorization approach and applied distance metrics to analyze all changes. We could see that our model outperforms existing neural network based models. Here we suggested the model that is experimented on small scale software. In the future, the work can be extended to detect the change and its impact on large-scale industry level software with backend and frontend software modules.

Conflicts of Interest

The authors declared no conflict of interest.

References

Ahmad, S., Ghani, A. A. A., Sani, F. M.: Dependence flow graph for analysis of aspect oriented programs. International Journal of Software Engineering & Applications, 5(6), 125 (2014)

Alon, U., Zilberstein, M., Levy, O., Yahav, E.: A general path-based representation for predicting program properties. ACM SIGPLAN Notices. 53(4), 404-419 (2018)

Angerer, F., Grimmer, A., Prähofer, H., Grünbacher, P.: Change impact analysis for maintenance and evolution of variable software systems. Automated Software Engineering. 26(2), 417-461 (2019)

Baah, G. K., Podgurski, A., Harrold, M. J.: The probabilistic program dependence graph and its application to fault diagnosis. IEEE Transactions on Software Engineering. 36(4), 528-545 (2010)

Badri, L., Badri, M., St-Yves, D.: Supporting predictive change impact analysis: a control call graph based technique. In Proceedings of Asia-Pacific Software Engineering Conference, IEEE. 9 (2005)

Bohner, S. A.: Impact analysis in the software change process: a year 2000 perspective. In icsm. 96, 42-51 (1996)

Büch, L., Andrzejak, A.: Learning-based recursive aggregation of abstract syntax trees for code clone detection. In Proceedings of International Conference on Software Analysis, Evolution and Reengineering. 95-104 (2019)

Gethers, M., Kagdi, H., Dit, B., Poshyvanyk, D.: An adaptive approach to impact analysis from change requests to source code. In Proceedings of IEEE/ACM International Conference on Automated Software Engineering. 540-543 (2011)

Hameed, Z., Garcia-Zapirain, B.: Sentiment classification using a single-layered BiLSTM model. IEEE Access. 8, 73992-74001 (2020)

Hoang, T., Kang, H. J., Lo, D., Lawall, J.: CC2Vec: Distributed representations of code changes. In Proceedings of ACM/IEEE International Conference on Software Engineering. 518-529 (2020)

Kitsu, E., Omori, T., Maruyama, K.: Detecting program changes from edit history of source code. In Proceedings of Asia-Pacific Software Engineering Conference, IEEE. 1, 299-306 (2013)

Liang, H., Yu, Y., Jiang, L., Xie, Z. Seml: A semantic LSTM model for software defect prediction. IEEE Access, 7, 83812-83824 (2019)

Meng, Y., Liu, L.: A Deep Learning Approach for a Source Code Detection Model Using Self-Attention. Complexity. (2020)

Molderez, T., Stevens, R., De Roover, C.: Mining change histories for unknown systematic edits. In Proceedings of International Conference on Mining Software Repositories, IEEE. 248-256 (2017)

Rahman, M., Watanobe, Y., Nakamura, K.: A Neural Network Based Intelligent Support Model for Program Code Completion. Scientific Programming. (2020)

Thomas, S. W., Adams, B., Hassan, A. E., Blostein, D.: Validating the use of topic models for software evolution. In Proceeding of IEEE working conference on source code analysis and manipulation. 55-64 (2010)

Tiwang, R., Oladunni, T., Xu, W.: A Deep Learning Model for Source Code Generation. In SoutheastCon, IEEE. 1-7 (2019)

Le TH, Chen H, Babar MA.: Deep learning for source code modeling and generation: Models, applications, and challenges. ACM Computing Surveys (CSUR). 53(3),1-38 (2020)

Zhang J, Wang X, Zhang H, Sun H, Wang K, Liu X.: A novel neural source code representation based on abstract syntax tree. In Proceedings of IEEE/ACM International Conference on Software Engineering (ICSE). 783-794 (2019)

Wang W, Li G, Ma B, Xia X, Jin Z.: Detecting code clones with graph neural network and flow-augmented abstract syntax tree. In Proceedings of International Conference on Software Analysis, Evolution and Reengineering, IEEE. 261-271 (2020)

White M, Tufano M, Vendome C, Poshyvanyk D.: Deep learning code fragments for code clone detection. In Proceedings of the International Conference on Automated Software Engineering (ASE), IEEE. 87-98 (2016)

Eid S, Makady S, Ismail M.: Detecting software performance problems using source code analysis techniques. Egyptian Informatics Journal. 21(4), 219-29 (2020)

Goknil A, Kurtev I, Berg KV.: A rule-based change impact analysis approach in software architecture for requirements changes. arXiv preprint arXiv. 1608.02757 (2016)

Musco V, Carette A, Monperrus M, Preux P.: A learning algorithm for change impact prediction. In Proceedings of the International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering, IEEE. 8-14 (2016)

Siow JK, Gao C, Fan L, Chen S, Liu Y. Core: Automating review recommendation for code changes. In Proceedings of the International Conference on Software Analysis, Evolution and Reengineering, IEEE. 284-295 (2020)

Dam HK, Pham T, Ng SW, Tran T, Grundy J, Ghose A, Kim T, Kim CJ.: A deep tree-based model for software defect prediction. arXiv preprint arXiv. 1802.00921 (2018)

https://www.sri.inf.ethz.ch/py150

Intelligent Automation & Soft Computing
DOI:10.32604/iasc.2022.022335
Article

Crow Search Algorithm with Improved Objective Function for Test Case Generation and Optimization

Meena Sharma and Babita Pathik*

Institute of Engineering & Technology, Devi Ahilya Vishwavidyalaya, Indore, 452001, India
*Corresponding Author: Babita Pathik. Email: babitapathik@gmail.com
Received: 04 August 2021; Accepted: 07 September 2021

Abstract: Test case generation and optimization is the foremost requirement of software evolution and test automation. In this paper, a bio-inspired Crow Search Algorithm (CSA) is suggested with an improved objective function to fulfill this requirement. CSA is a nature-inspired optimization method. The improved objective function combines branch distance and predicate distance to cover the critical path on the control flow graph. CSA is a search-based technique that uses heuristic information for automation testing, and CSA optimizers minimize test cases generated by satisfying the objective function. This paper focuses on generating test cases for all paths, including critical paths. The control flow graph covers the information flow among all the classes, functions, and conditional statements and provides test paths. The number of test cases examined through graph path coverage analysis. The minimum number of test paths is counted through complexity metrics using the cyclomatic complexity of the constructed graph. The proposed method is evaluated as mathematical optimization functions to validate their effectiveness in locating optimal solutions. The python codes are considered for evaluation and revealed that our approach is time-efficient and outperforms various optimization algorithms. The proposed approach achieved 100% path coverage, and the algorithm executes and gives optimum results in approximately 0.2745 seconds.

Keywords: Test case generation; Crow Search Algorithm; improved objective function; control flow graph; branch distance; predicate distance

1 Introduction

The generation or selection of test cases is critical for software testing to ensure its eminence as a product. Software testing is a process of ensuring that the actual result matches the desired result. It also ensures that the software is free from any kind of bugs or defects. There are various testing methods, such as manual testing or automation testing. Manual testing consumes 40%–70% of the time and expense associated with software development. The testing team performs software testing by examining the structure of the code deeply. It passes through each line of code to generate a test case suite and test data. Software testing determines the critical errors in the software products optimally so that error detection should be maximum. Simultaneously testing cost and time should be minimal [1,2]. Testing required some effort in terms of time and cost. Estimation is the process of approximating the needful value for business and industry purposes, even if there is any type of inconsistent data. Test estimation is a maintenance activity that predicts how long such a task would take to complete.

Numerous approaches for selecting test cases have been suggested to address the time constraint inherent in conventional testing. For developing test cases automatically, Search-Based Testing (SBT) is considered. There have been meta-heuristic techniques effectively utilized in the development of test data, including Genetic Algorithms (GA) [3,4], Ant Colony Optimization(ACO) [5,6], Particle Swarm Optimization (PSO) [7,8], Artificial Bee Colony (ABC) [9,10], hybrid Genetic Algorithm [11], Bat Algorithm [12,13]. Fitness functions (FF) drive a metaheuristic algorithm search process. The FF is measured through a mathematical equation that evaluates the feasibility of every solution in the search space by assigning a value. In SBT, FFs are primarily defined in terms of coverage criteria. A metric that determines how fine the generated test cases exercise the software under test is named as coverage criteria [14]. Path coverage, statement coverage, decision coverage, and branch coverage are all frequently used criteria [15]. For covering the target path, many researchers [10,16] employ a FF based on branch distance. Branch distance is estimated by examining the branching node's conditional statement and sometimes missed target [17]. The branch distance value alone is insufficient to produce the desired path. The branch distance is paired with heuristic information of approximation level to improve the searching. It indicates the distance between the target node and the node currently visited [16]. When there is a low probability of covering a path, this combined FF fails [18].

In this work, a CSA based approach is paired with an enhanced combination function to generate test cases for the critical path. The innovative aspect of this paper's search strategy is introducing an improved objective function (IOF). IOF is a FF that combines branch distance with predicate distance. The objective behind considering CSA is the behavior of crow flock that resembles an optimization process [19]. Crows keep their extra food in explicit locations (hiding spots) in their environs and retrieve it once required. Crows are hungry birds as they flock together in search of better food supplies. From an optimization perspective, the crows represent searchers or agents. The environs represent searching space, and each location in the environs represents a feasible solution. The objective function is represented by food source quality, and the global solution to the problem is the finest food source in the environment.

We focus on finding the optimal test case suite of software. The selection of test cases is one of the significant parts of the test estimation. The aim is the critical path coverage so that test case selection will be effective, and any typical test case should not be skipped. The total test case selected is observed through Cyclomatic Complexity(CC) [20] by tracing paths on the CFG. In the proposed work, CSA is considered for generating test cases automatically and optimizing the test suite by applying FF. CSA with IOF is a novel approach that we considered in the paper. The main contributions of the paper are:

- Generate CFG for program code and trace the graph for complete path coverage without dropping any critical path.
- Compute the CC to approximate the number of test cases.
- Apply CSA search technique with path or branch based IOF.

Other content of the article organized as Section 2 summarizes the contributions done by researchers in this area. Section 3 elaborates on the background of the method opted in this paper. Section 4 comprises the proposed method, whereas Section 5 describes experimental setup and evaluation, Section 6 containing results and discussion. Section 7 concludes the work and mentions future works.

2 Related Work

Generation of test data automatically using SBT is a motivating research area. The criteria for code-level testing is path coverage. This section focuses primarily on coverage of the branch or path code using

AI-based heuristic search algorithms, using path/branch-based FFs. Some nature-inspired population based algorithms used by several researchers for test case generation are also mentioned here. The concepts and contributions are pithy here.

A hybrid and simple genetic algorithm are proposed by Garg D. and Garg P. [18] for path testing. An ExLB FF is suggested for path coverage. Although HGA provides better coverage through ExLB, it does not cover the target path. A genetic algorithm (GA) based algorithm is used for test data generation by Pachauri et al. [21]. The FF combines branch and path information for computation. The results of experiments reveal significantly better coverage percentages during the search, still difficult to get the targeted path. Babamir FS et al. [4] proposed a GA based testing technique to automate test data generation using different parameters for structural-oriented program structure. The FF proposed by the authors tries to cover program paths maximum possible way but does not achieve the targeted path. Jia YH et al. [7] developed a PSO based optimization algorithm for automatic test data generation. The criteria for software testing is condition-decision coverage that covers all conditional statements. This approach also does not cover the target path. PSO algorithm-based test case generation is suggested by Huang M et al. [8]. To improve the performance of PSO, they merged the group self-activity feedback (SAF) operator and Gauss mutation (G) changing inertia weight. That improved approach is efficient in test case generation for multi-path. Khan SA et al. [22] suggested Particle Swarm Optimization (PSO) technique for generating test case data for integration testing. Dahiya SS et al. [10] presented an automatic test generation for structural software using an artificial bee colony (ABC) based novel search technique. The FF generates test data with branch distance-based objective function. The technique is not suitable for large inputs and where constraints have many equality constraints. Sheoran S et al. [9] presented a novel approach using the ABC algorithm for data flow testing search. The approach prioritizes the definition-use paths that are not definition-clear paths. Aghdam ZK and Arasteh B. [23] presented an ABC algorithm for automatic test data generation. The experiment was evaluated with a FF that considers branch coverage criteria to optimize the solutions.

All the approaches discussed above cover a single path at a time that is a time-consuming process. Following are some approaches presented to address this gap by searching for multiple paths.

The Dynamic Multiple-Objective Sorting Algorithm is offered by Panichella A et al. [21] to handle many objectives. The FF combines approach level and branch distance in GA for branch coverage. Lv XW et al. [16] proposed the PSO approach with metamorphic relationships for generating test cases for multiple path coverage. The aim was to cover multiple paths efficiently with fewer iterations.

The attention of these mentioned works is on multiple path coverage, but they did not cover the critical path. In our proposed technique, we focus on covering many paths automatically along with the critical path in less time consumption.

3 Background

A population based metaheuristic CSA is utilized for searching all crucial paths for test case generation. The common heuristics, *e.g.*, branch distance, approximation level, have been considered. A graphical representation of source code called Control Flow Graph [24] is generated for path coverage. CFG is giving a moderate and sometimes exact prediction of test cases. The two important heuristic parameters are considered here: Branch distance and Predicate distance.

3.1 Path Coverage

Testing with path coverage generates test cases by executing each path at least once in order to identify defects included within the path. First, we determine the total linear independent branches/paths by measuring the CC [15,18]. CC is a software metric measured to analyze the code's probable error. The complexity metric was first introduced by Thomas J. McCabe [20]. CFG, an intermediate graph of lines of code, is constructed for the calculation of CC. Then, using the McCabe formula, CC is determined from the source code's CFG. CFG is the combination of nodes and edges that denote instructions and flow of control or data, respectively.

The mathematical equation to calculate the CC of a syntax drawn from the graph V(G) is:

$$V(G):\ M\ =\ E\ -\ N\ +\ 2 \tag{1}$$

Here M = CC of graph G

E = # edges in the graph

N = # nodes in the graph

The calculation of complexity is as follows:

$$V(G):\ M\ =\ P+1 \tag{2}$$

Here P = no. of predicate node, the node with the conditional statement

CC is a metric of statistics about independent paths equivalent to the number of test case suites. Using Eqs. (1) or (2), we can find the CC, and both equations give the same amount of test case suite.

In this work, we consider two case studies for test cases by covering the multiple paths. The first one is a python program for Triangle Identification Problem [16,18] and generates test cases with 100% path coverage. The second example is a small python program to Identify Armstrong Numbers.

3.1.1 Triangle Identification Problem (TIP)

This python program, given in Fig. 1a, identifies the type of triangle and classifies the angle. Triangle classification is done based on three sides of the triangle. This code segment has a conditional statement to decide whether a triangle is equilateral, isosceles, scalene, or it is not a triangle according to sides given to the program as input. It also classifies the angle of a triangle as acute or not given input. For the given program, a CFG is generated, as depicted in Fig. 1b.

(a)

```
Input(a,b,c)
1.  if a > 0 and b > 0 and c > 0:
2.        if a == b and b == c:       //Triangle
3.             print("Equilateral")
4.        elif a == b or b == c or c==a:
5.             print("Isosceles")
                 else:
6.             print("Scalene")
7.        if a + b > c:                //Angle
8.              print("and acute")
    else:
9.       print("Can't form a triangle ")
    STOP
```

(b)

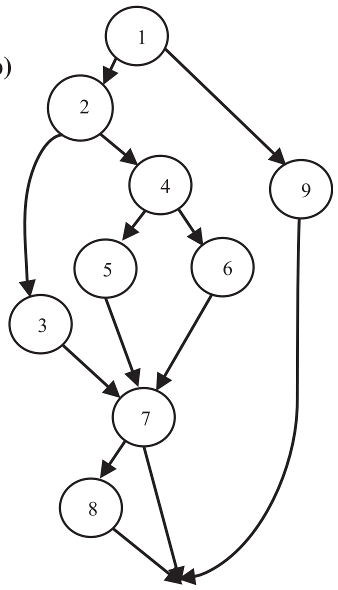


Figure 1: TIP (a) Source code in python (b) CFG

Each node of the graph presents a complete statement, and each edge of the graph carries the data and control flow between two statements.

The linearly independent paths for the TIP program is obtained from its CFG as:

Path_1: 1 – 2 – 3 – 7 – 8 – S

Path_2: 1 – 2 – 4 – 5 – 7 – S

Path_3: 1 – 2 – 4 – 5 – 7 – 8 – S

Path_4: 1 – 2 – 4 – 6 – 7 – S

Path_5: 1 – 2 – 4 – 6 – 7 – 8 – S

Path_6: 1 – 9 – S

The outcomes of all these paths are given below:

Path_1: Equilateral and acute

Path_2: Isosceles

Path_3: Isosceles and acute

Path_4: Scalene

Path_5: Scalene and acute

Path_6: Can't form a triangle

The total number of the independent path can be confirmed with CC calculated by using Eq. (1).

$M = 13 - 9 + 2 = 6$

where $E = 13$ $and\ N = 9$

The input to the program is given as three sides of triangles a, b, c as test data. If the value of a, b, c is less than or equal to zero, then a triangle cannot be formed. The probability is very low or zero for that test data generation. The probability of generating test data for these values is zero. Although the probability of the aimed crucial path is always less than that of other alternatives, but it must be higher than 0. The path with 0 probability is an infeasible solution. The probability of forming an equilateral triangle is on all equal test data. All the paths have higher probability except Path_1, which has the lowest probability than others according to probability measures. In the example given in Fig. 1, the Path_1 that results in "Equilateral and acute" is reflected as a critical path.

3.1.2 Identify Armstrong Number (IAN)

The part of the source code for Identification of an Armstrong number is shown in Fig. 2a. Input to this algorithm is a number to check whether it is an Armstrong number or not. In this program, a modulus, power, and division calculation are implemented within the for a loop. Whereas in TIP, only decision and selection statements are included.

The CFG for IAN is shown in Fig. 2b. The total number of the linearly independent path traced through path coverage is three for testing the IAN program. CC of the code can be calculated by equation no (1). According to the graph shown in Fig. 2b, Path 2 (dashed line) has a lower probability of covering. The other paths, i.e., Path 1 and Path 3, respond earlier than Path 2 because they have a higher probability of traversing if the input range is taken between [0,152]. Path 2 is considered a critical path.

(a)

```
Input(no)
 1. s = 0
 2. temp = no
 3. while temp > 0:
 4.     dig = temp % 10
              s += dig ** 3
              temp //= 10
 5. if no == s:
 6.    print("Armstrong number")
    else:
 7.    print("Not Armstrong number")
    STOP
```

(b)

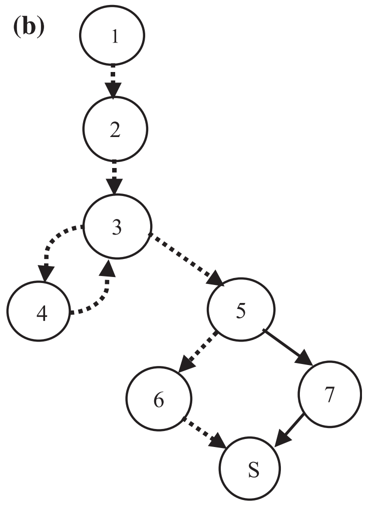


Figure 2: IAN (a) Source code in python (b) CFG

3.2 Crow Search Algorithm

Crows are a widespread genus of birds that are the most intelligent fowls among the creatures on the planet. They are capable of memorizing faces, using tools, communicating in complicated ways, and concealing and retrieving food. There are numerous examples of crow's intelligence. These creatures have shown self-alertness and possess the ability to create tools. Additionally, they are capable of recalling the location of their meal up to several months later.

CSA is a unique population-based metaheuristic introduced by Askarzadeh A, 2016 [19], for solving engineering optimization issues. The primary source of inspiration of CSA is the behavior of crows, the memorization of hiding places used to store excess food, the following of one another during thefts, and the protection of their caches being stolen.

The fundamental principles of the algorithm include the organization of crows into flocks. These ideas resulted in the construction of a novel algorithm that is significantly different from existing algorithms that take their primary inspiration from the natural behavior of birds, such as Bird Swarm Algorithm (BSA), Bird Mating Optimizer (BMO), Chicken Swarm Optimization (CSO), Cuckoo Search (CS), and Peacock Algorithm (PA). The following are the CSA principles:

1. Crows live in large families (flocks).
2. Crows can remember and recognize the hiding place of food.
3. They follow each other to thieve their food.
4. Crows guard their stocks against theft by some probability.

The parameters for the algorithm are mentioned here: N is the flock size (number of crows), d is the number of dimensions in the search space. Itr^{max} is the maximum number of iterations, $\{c^{min}, c^{max}\}$ denotes the range of possible crow position. At time Itr in search space, the position of crow i denoted as $c^i(Itr)$ where $i = 1, 2, 3 \ldots \ldots N$, $Itr = 1, 2, 3, \ldots \ldots Itr^{max}$ and $c^i(Itr) = \left[c_1^i, c_2^i, \ldots \ldots . . c_d^i\right]$.

The position of the hiding place of crow i at iteration Itr is given as $m^i(Itr)$. Now suppose the crow j wishes to visit its hiding spot, $m^j(Itr)$, during iteration iter. Now crow i wants to follow crow j and tracks crow $j's$ hiding location in this Itr. The output of CSA is the i^{th} item from memory m for which the value of $OF(m^i)$ is either minimum or maximum in the minimization or maximization cases.

Following is the matrix that describes the N crows searching in d-dimensional search space and positioned randomly. Each of the members of the flock represents a feasible solution to the problem, while d symbolizes the number of choice variables.

$$Crows = \begin{bmatrix} C_1^1 & C_2^1 & \cdots & C_d^1 \\ C_1^2 & C_2^2 & \cdots & C_d^2 \\ \vdots & \vdots & \vdots & \vdots \\ C_1^N & C_2^N & \cdots & C_d^N \end{bmatrix} \tag{3}$$

The value of memory m^i is initialized with the value of c^i. Initially $Memory(m) = Crows(c)$.

$$Memory = \begin{bmatrix} m_1^1 & m_2^1 & \cdots & m_d^1 \\ m_1^2 & m_2^2 & \cdots & m_d^2 \\ \vdots & \vdots & \vdots & \vdots \\ m_1^N & m_2^N & \cdots & m_d^N \end{bmatrix} \tag{4}$$

3.3 Objective Function

The objective function OF(x) evaluates by using the fitness of c^i. The input parameter x is the vector of decision variables. CSA technique researchers use the number of dimensions d in search space to evaluate fitness. Branch distance is determined for conditional nodes using test data. The value decides how close or distant the test data must be in order to satisfy the condition (true/false) [25]. In this research paper, we suggest an improved objective function with two heuristic values.

4 Proposed Method

In this paper, we proposed a CSA based objective function that is modified for the fitness of the flocks. We analyze path coverage to check the criticality of a branch that should not be skipped. We apply CSA with the suggested FF to generate test cases. Fig. 3 showing the overall flow of the proposed method.

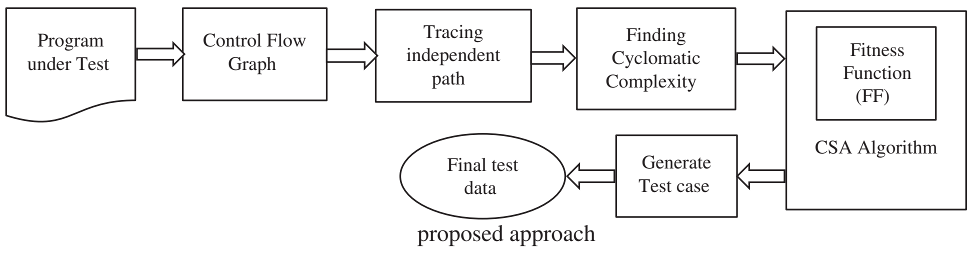


Figure 3: Flow diagram of the proposed approach

4.1 Test Case Selection with CSA

Each crow denotes one test path in the CSA stated in ALGORITHM 1 as given in Fig. 4. The next path is selected by updating the value by the statement with the equation below:

$$c_j^i = c_j^i + r^i \times fl \times \left(m_j^k - c_j^i\right) \quad \text{if } r^i \geq AP \tag{5}$$

$$c_j^i = r^j \times \left(c^{max} - c^{min}\right) + c^{min} \quad \text{if } r^i < AP \tag{6}$$

where c_j^i is the new position of crow and r^i is random value ranges [0,1], and *fl* is flight length that is an adjustable parameter. Awareness probability (AP) is also an adjustable parameter of this population based algorithm. These are the two main parameters of CSA.

```
ALGORITHM 1: Crow Search Algorithm for Test Case Generation
Input:  Source code, N, $Iter^{max}$, $c^{max}$, $c^{min}$
Output: Test case for critical path
Initialize flock of Crow (test cases) N
for i = [1.. N]:
     initialize crow $c^i$ in the d-dimensional search space (inputs)
     initialize memory m of each crow(test data)
     evaluate fitness value of $c^i$
end for
t = 0
while t < $Iter^{max}$
     for i = [1.. N]:
          select a crow k (path) randomly to follow from {1, N}
          initialize awareness probability AP
          choice random value $r^i$ range [0,1]
         if $r^i \geq AP$
             for j = [1.. d]:
                  $c_j^i = c_j^i + r^i \times fl \times (m_j^k - c_j^i)$
             end_for
         else
             for j = [1..d]:
                  choice random value $r^i$ range [0,1]
                  $c_j^i = r^j \times (c^{max} - c^{min}) + c^{min}$
             end_for
         end_if
   end_for
   for i = [1.. N]:
         calculate the new position of $c^i$ (next path)
         calculate the new value of memory $m^i$
         if IOF ($c^i$) < IOF($m^i$ ):
              $c^i$ updated by $m^i$
         end_if
   end_for
    t = t + 1
end_while
```

Figure 4: Algorithm for test case generation

The memory for crow *j* indicate by m_j^k, where *j* varies from {1,…,d} and *k* varies from {1,…,N}. The first circumstance in which the crow c^i follows another crow c^j from the flock with the primary goal of discovering that crow's memory m^j and the second instance corresponds to the circumstance in which the new position in the d-dimensional search space is initialized randomly. In our proposed method, dimension d is initialized with the maximum number of nodes in any path, called path length. c^i is validated for each path.

All of the values of the d-dimensional vector fall within the interval [c^{min}, c^{max}], then c^i is the feasible solution. The selection of the paths are updated to reflect values from the interval [c^{min}, c^{max}] as, if $c_j^i > c^{max}$, then $c_j^i = c^{max}$, and if $c_j^i < c^{min}$, then $c_j^i = c^{min}$.

In this work we initialize interval [c^{min} , c^{max}] = [1,8] means c^{min} = 1 and c^{max} = 8.

4.2 Branch Distance Calculation

The main purpose is to develop test cases for paths that have a very low probability of being covered during automated testing. Automatic test case generation becomes extremely challenging in this situation, as the test data is sparse and covers a wide range.

It is critical to direct the search process in order to obtain that kind of test data. In our proposed searching technique, we consider Branch Distance (BD) and Predicate Distance(PrD) for improved objective function (IOF). Fig. 5 showing all predicate nodes and brach distance.

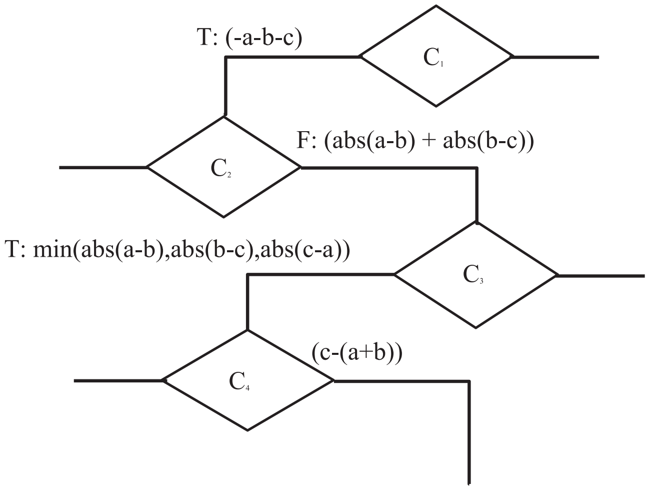


Figure 5: Branch distance for the target of four conditional statements in TIP

The BD determines the degree to which the input deviates from the predicate [16]. It specifies whether the test data is close or far to satisfy the conditional statement of the code [25].

Here we consider TIP code for the explanation of the proposed technique. The conditional nodes in this example are node1, node2, node4, and node6. Suppose predicate node for a path is node1 and denoted as $Pr_{1,}$ then Branch Distance is shown as:

$$BD(Pr_1) = BD(a > b) = abs(b - a) \tag{7}$$

Tab. 1 has the description of all predicates and their BD values. This BD is normalized [26] to map the value of function within the interval [0,1] through the following formula for input x:

$$NBD(x) = 1 - (1.0001)^{-x} \tag{8}$$

where NBD is Normalized Branch Distance. This is the formula for Branch Distance Fitness (BDF).

4.3 Predicate Distance (PrD)

In our objective function, the second fitness parameter we consider is predicate distance (PrD). PrD is the difference of nodes from the predicate node between the target(critical) path and other traversed path and denoted as:

$$PrD_i = diff\_node(P_i\ ,\ P_t) \tag{9}$$

where P_i is i^{th} traversed path and P_t is the targeted path.

Table 1: Fitness function evaluation of various predicates suggested by Tracey N [27]

Predicates	BD Evaluation
x > y	if y − x < 0, then 0 else (y − x) + K
x ≥ y	if y − x ≤ 0, then 0 else (y − x) + K
x < y	if x − y < 0, then 0 else (x − y) + K
x ≤ y	if x − y ≤ 0, then 0 else (x − y) + K
x = y	if abs(x − y) = 0, then 0 else abs(x − y) + K
x and y	BD(x) + BD(y)
x or y	min [BD(x), BD(y)]

4.4 *Improved Objective Function*

The path (c^i) is evaluated by using IOF and if the value of IOF(c^i) is less than the value of IOF(m^i) then m^i is updated to the value of c^i. Entered into the next iteration for repeating the evaluation. The proposed improved objective function is given for the traversed path P_i on test cases t and predicate node Pr_j as:

$$IOF(t) = \sum_{i,j=1}^{N} BD_{ij} + PrD_j \tag{10}$$

The value of IOF(m^i) should be minimal.

5 Experimental Setup and Evaluation

5.1 *Predicate Distance Calculation*

A matrix is generated from the CFG depicted in Fig. 1b, and it contains all of the test paths. This matrix is supposed to initialize the crow positions. The row denotes the paths for the crow, whereas the column denotes the nodes as crow position.

$$\text{Paths} = \begin{bmatrix} 1 & 2 & 3 & 7 & 8 & 0 & 0 \\ 1 & 2 & 4 & 5 & 7 & 0 & 0 \\ 1 & 2 & 4 & 5 & 7 & 8 & 0 \\ 1 & 2 & 4 & 6 & 7 & 0 & 0 \\ 1 & 2 & 4 & 6 & 7 & 8 & 0 \\ 1 & 9 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

All these paths are equalized in length by appending zero at the trailing position. We replace the letter ‘S’ with zero (0) to simplify the calculation of path distances. The final node ‘stop’ is the same in all the paths, which does not affect any computation. The nodes that exist on a path of the graph (CFG) are matched against the target path (critical). Predicate Distance is the number of unmatched nodes on these two paths. The critical path is Path_4, as discussed in section 3.1.1.

Now we calculate PrD of target path with every traverse path as:

Target path: P_t = [1 2 3 7 8 0 0] = P_4

P_t: 1 2 3 7 8

PrD(P_1 , P_t) = 0

$PrD(P_2 , P_t) = 2$

$PrD(P_3, P_t) = 1$

$PrD(P_4 , P_t) = 2$

$PrD(P_5 , P_t) = 1$

$PrD(P_6 , P_t) = 4$

These values are going to be used to calculate fitness values through the CSA objective function.

5.2 CSA implementation for Generating Test Cases

The CSA is implemented on TIP with IOF function is described here:

Step 1: Initialize the crow population (number of test data)

$N = 5$

Dimension of search space is initialized with $d = 3$

Maximum Number of iterations $Itr = 20$

We initialize matrix of test data as the initial crow position as:

$$\text{Crow} = \begin{bmatrix} 2 & 2 & 5 \\ 4 & 9 & 3 \\ 3 & 7 & 5 \\ 2 & 4 & 7 \\ 1 & 3 & 3 \end{bmatrix}$$

Initially, the position of each crow is equivalent to the value of the memory.

So $c_1 = m_1 = [2\ 2\ 5]$, $c_2 = m_2 = [4\ 9\ 3]$, $c_3 = m_3 = [3\ 7\ 5]$, $c_4 = m_4 = [2\ 4\ 7]$, $c_5 = m_5 = [1\ 3\ 3]$.

Fitness value evaluation:

Next, each crow's position is evaluated through the improved objective function $IOF(c^i)$. The fitness value is initialized by using Eq. (9). Test case c_1 [2 2 5] moves to the Path_2 and changes its position towards 'Isosceles.' BD is evaluated by using rule suggested by Tracey N et al.

$BD(c_1) = BD_1 + BD_2 + BD_3 + BD_4$

BD_1: $(0-2)+(0-2)+(0-5) = -9<0 = 0$

BD_2: $|(2-2)|+ |(2-5)| = 0+3 = 3+0.1 = 3.1$

BD_3: $\min(|(2-2)|, |(2-5)|, |(5-2)|) = \min(0, 3.1, 3.1) = 0$

BD_4: $(c-(a+b)) = (5-(2+2)) = 1+0.1 = 1.1$

$BD(c_1) = 0+3.1+0+1.1 = 4.2$

$NBD(c_1) = 1 – (1.001)^{-BD} = 1 – (1.001)^{-4.2} = 0.0041$

Fitness evaluation $IOF(c_1) = BD + PrD = 0.0041+2 = 2.0041$ by Eq. (10)

Test case(c_2) = [4 9 3] and Test case(c_3) = [3 7 5] move to Path_5 and leads to same position 'Scalene and acute'

$BD(c_2) = BD_1 + BD_2 + BD_3 + BD_4 = 0+11.2+1+0 = 12.2$

$NBD(c_2) = 1- (1.001)^{-12.2} = 0.0121$

Fitness evaluation $IOF(c_2) = BD + PrD = 0.0101 + 1 = 1.0121$

$BD(c_3) = BD_1 + BD_2 + BD_3 + BD_4 = 0+9.2+2+0 = 11.2$

$NBD(c_3) = 1\text{- } (1.001)^{-11.2} = 0.0111$

Fitness evaluation $IOF(c_3) = BD + PrD = 0.0101 + 1 = 1.\ 0111$

Test case(c_4) = [2 4 7] is covering the Path_4 for 'Scalene'

$BD(c_4) = BD_1 + BD_2 + BD_3 + BD_4 = 0{+}2.1{+}0{+}1.1 = 3.2$

$NBD(c_4) = 1\text{- } (1.001)^{-3.2} = 0.0031$

Fitness evaluation $IOF(c_4) = BD + PrD = 0.0031{+}2 = 2.0031$

Likewise, for Test case(c_5) = [1 3 3] to 'Isosceles and acute' by moving on Path_3

$BD(c_5) = BD_1 + BD_2 + BD_3 + BD_4 = 0{+}2.1{+}0{+}0 = 2.1$

$NBD(c_5) = BDF = 1\text{- } (1.001)^{-2.1} = 0.0020$

$IOF = 0.0020{+}1 = 1.0020$

Tab. 2 consists of fitness values calculated above using BDF and IOF for all the randomly selected test cases and shown in the matrix as crow's initial positions.

Table 2: Fitness value as per BDF and IOF for all test data

Test Case	A	B	C	BDF	IOF
C_1	2	2	5	0.0041	2.0041
C_2	4	9	3	0.0121	1.0121
C_3	3	7	5	0.0111	1.0111
C_4	2	4	7	0.0031	2.0031
C_5	1	3	3	0.0020	**1.0020**

The minimum fitness value observed is $IOF(c_5) = 2.0020$

Step 2: We initialised $AP = 0.5, fl = 0.7$ r = random number ranges [0,1]

Next, the new position of crows to follow the path is computed by using Eqs. (5) and (6), given below:

$$\text{Crow} = \begin{bmatrix} 4 & 6 & 4 \\ 4 & 2 & 8 \\ 3 & 5 & 7 \\ 6 & 6 & 2 \\ 4 & 1 & 5 \end{bmatrix} = \text{New position}$$

Now, again we evaluate fitness value for the first iteration. The value for the IOF for all new test cases given above is shown in Tab. 3. The fitness value is be replaced by step mentioned in the algorithm as: '*if IOF* $(c^i) < IOF(m^i)$' and test cases also updated accordingly.

Table 3: Updated fitness value after 1st iteration on new test cases

Test Case	A	B	C	IOF
C_1	4	6	4	1.0041
C_2	4	9	3	1.0121
C_3	3	5	7	1.0061
C_4	6	6	2	1.0040
C_5	1	3	3	**1.0020**

In Tab. 3, the fitness value of C_1, C_3, and C_4 are lower than previous memory and only replaced in a new position. Repeat the same step for the second iteration, find a new crow position, and evaluate fitness value. The comparison is made with IOF(c^i) and IOF(m^i). The position of the crow is updated according to CSA.

As shown in Tab. 4, the fitness value for test cases C_2 and C_4 have been updated. Again the process is repeated for the third iteration to find the new position of crow and their fitness value. The position as test cases and respective fitness values are updated and mentioned in Tab. 5. The CSA with IOF can achieve the target in few iterations. Although test on large numbers requires more iterations. From Tab. 5, we found a test case (C_2) with [3 3 3] that leads to the critical path 'Equilateral and acute' triangle with minimum fitness value.

Table 4: Updated fitness with changes in C_2 and C_4

Test Case	A	B	C	IOF
C_1	4	6	4	1.0041
C_2	5	4	7	1.0051
C_3	3	5	7	1.0061
C_4	2	3	4	1.0022
C_5	1	3	3	**1.0020**

Table 5: Updated fitness with the minimum value

Test Case	A	B	C	IOF
C_1	4	6	4	1.0041
C_2	**3**	**3**	**3**	**0.0000**
C_3	3	5	7	1.0061
C_4	2	3	4	1.0040
C_5	1	3	3	1.0020

The proposed CSA based approach with improved objective function has its advantage over other metaheuristic algorithms in that the IOF covers almost every path by generating various test cases, including the critical path. The test case for the crucial path needs only a minimum number of iterations and reduced runtime.

6 Results and Discussion

The parameter setting plays a key role in improving the performance of any optimization algorithm. Tab. 6 depicts the parameter setting of CSA for TIP and IAN. The values are based on some research work [3,16,18], general sources, and the nature of the programs. The system configuration is Windows 10, 8GB RAM, Intel Core i7,4690T central processing unit, 2.50 GHz, 64-bit OS, x64 based processor. CSA is implemented in an Anaconda environment.

A comparative analysis is done with the other metaheuristic algorithm such as PSO and APSO. These optimization algorithms are applied with BDF and Combined Fitness Function (CFF). The authors have considered the triangle classification problem and area calculation for empirical evaluation.

Table 6: Parameters for CSA

Parameters	TIP	IAN
Population Size	1000, 25000	50000
Dimension of search space	3	1
Awareness Probability(AP)	0.5	0.5
Flight Length(FL)	0.7	0.7
Maximum number of iterations	100	50
C^{max} (maximum test data)	9	600000
C^{min} (minimum test data)	1	100

We also have considered the same problem for comparison purposes. To the best of our knowledge CSA algorithm with IOF is yet not applied for test case generation and optimization.

So here, we compared our results with the algorithm that considered the TIP problem with the population N =1000 and iterations $Iter$ = 100 as parameter setting.

According to results shown in Tab. 7, the CSA with IOF giving test data for the target path in the 3rd iteration but within an average execution time of 0.2745 sec.

Table 7: TIP comparison with various population

Algorithm	Iterations to achieve target path	Execution time in s (avg)
PSO with BDF	3	1.3535
APSO with BDF	3	1.5002
PSO with CFF	3	1.3560
APSO with CFF	2	0.4406
CSA with IOF	3	0.2745

The results produced through the existing FFs and the suggested FF differ significantly in terms of how many iterations and the average time spent on implementation. CSA with ICF produces better results for considered TIP case study including IAN since this algorithm implements an efficient search strategy.

Fig. 6. has the details about the number of test cases generated for each test path for TIP in 100 iterations on 1000 population (test data) as given in Tab. 8.

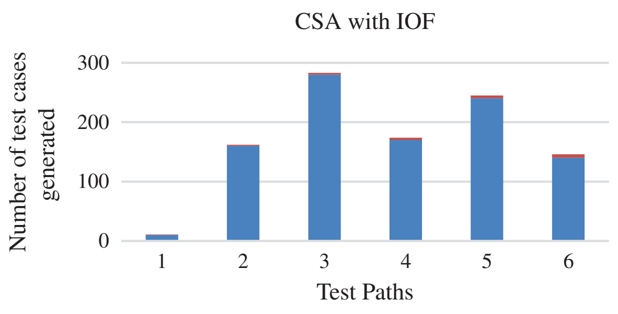


Figure 6: Test case generated with CSA

Table 8: Number of generated test cases for all paths

Test Paths	Path Description	Number of Test cases
Path_1	Equilateral and acute	10
Path_2	Isosceles	160
Path_3	Isosceles and acute	280
Path_4	Scalene	170
Path_5	Scalene and acute	240
Path_6	Can't form a triangle	140

7 Conclusions and Future Work

In this paper, the main objective was to cover the target path with a minimum time span. We adopt CSA based optimization technique by improvising its objective function. The fitness value is evaluated by using branch distance and predicate distance in IOF that guide the search algorithms to develop various test cases to cover a maximum number of paths automatically. The key idea was finding a test case for critical path minimum time and fewer iterations. A population-based metaheuristic algorithm is used. A heuristic path distance function IOF searched the target path efficiently. To carry out the experiments, we considered two case studies, TIP and IAN. We have achieved almost 100% complete path coverage for the case studies we considered using the suggested FF of CSA. The importance of the FF is that we reach our aimed path in fewer iterations and very little time span compared to the other existing heuristic algorithm with their branch distances FFs. We combine the predicate distance with branch distance as its heuristic distance to give the search technique a suitable direction.

Our future work is to cover additional test paths concurrently in evolutionary software systems. We will target to cover critical paths for regression testing and automate test case generation.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] A. Z. Karimi and B. Arasteh, "An efficient method to generate test data for software structural testing using artificial bee colony optimization algorithm," *Int. Journal of Software Engineering and Knowledge Engineering*, vol. 27, no. 06, pp. 951–966, 2017.

[2] P. R. Srivastava, "Optimization of software testing using genetic algorithm," in *Proc. Information Systems, Technology and Management*, Berlin, Heidelberg, Springer, pp. 350–351, 2009.

[3] A. Pachauri and G. Mishra, "A path and branch based approach to fitness computation for program test data generation using genetic algorithm," in *Proc. IEEE Int. Conf. on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE)*, Greater Noida, India, pp. 49–55, 2015.

[4] F. S. Babamir, A. Hatamizadeh, S. M. Babamir, M. Dabbaghian and A. Norouzi, "Application of genetic algorithm in automatic software testing," in *Proc. Int. Conf. on Networked Digital Technologies*, Berlin, Heidelberg, Springer, pp. 545–552, 2010.

[5] S. Sankar and V. Chandra, "An ant colony optimization algorithm based automated generation of software test cases," in *Proc. Int. Conf. on Swarm Intelligence*, Cham, Springer, pp. 231–239, 2020.

[6] S. Biswas, M. S. Kaiser and S. A. Mamun, "Applying ant colony optimization in software testing to generate prioritized optimal path and test data," in *Proc. ICEEICT*, IEEE, pp. 1–6, 2015.

[7] Y. H. Jia, W. N. Chen, J. Zhang and J. J. Li, "Generating software test data by particle swarm optimization," in *Proc Asia-Pacific Conf. on Simulated Evolution and Learning*, Springer, pp. 37–47, 2014.

[8] M. Huang, C. Zhang and X. Liang, "Software test cases generation based on improved particle swarm optimization," in *Proc Int. Conf. on Information Technology and Electronic Commerce*, IEEE, pp. 52–55, 2014.

[9] S. Sheoran, N. Mittal and A. Gelbukh, "Artificial bee colony algorithm in data flow testing for optimal test suite generation," *Int. Journal of System Assurance Engineering and Management*, vol. 11, no. 2, pp. 340–349, 2020.

[10] S. S. Dahiya, J. K. Chhabra and S. Kumar, "Application of artificial bee colony algorithm to software testing," in *Proc. Australian Software Engineering Conf.*, IEEE, pp. 149–154, 2010.

[11] A. Alhroob, W. Alzyadat, A. T. Imam and G. M. Jaradat, "The genetic algorithm and binary search technique in the program path coverage for improving software testing using big data," *Intelligent Automation & Soft Computing*, vol. 26, no. 4, pp. 725–733, 2020.

[12] P. R. Srivastava, A. Bidwai, A. Khan, K. Rathore, R. Sharma *et al.,* "An empirical study of test effort estimation based on bat algorithm," *Int. Journal of Bio-Inspired Computation*, vol. 6, no. 1, pp. 57–70, 2014.

[13] M. M. Öztürk, "A bat-inspired algorithm for prioritizing test cases," *Vietnam Journal of Computer Science*, vol. 5, no. 1, pp. 45–57, 2018.

[14] M. Shahid, S. Ibrahim and M. N. Mahrin, "A study on test coverage in software testing," *Advanced Informatics School (AIS), Universiti Teknologi Malaysia, International Campus, Jalan Semarak*. Kuala Lumpur, Malaysia, 2011.

[15] R. Mall, "Fundamentals of software engineering," in *PHI Learning Pvt. Ltd.*, Delhi, India, 2018.

[16] X. W. Lv, S. Huang, Z. W. Hui and H. J. Ji, "Test cases generation for multiple paths based on PSO algorithm with metamorphic relations," *IET Software*, vol. 12, no. 4, pp. 306–317, 2018.

[17] A. P. Gursaran, "Program test data generation branch coverage with genetic algorithm: Comparative evaluation of a maximization and minimization approach," *Int. Journal of Software Engineering & Applications*, vol. 3, no. 1, pp. 207–218, 2012.

[18] D. Garg and P. Garg, "Basis path testing using SGA & HGA with ExLB fitness function," *Procedia Computer Science*, vol. 70, pp. 593–602, 2015.

[19] A. Askarzadeh, "A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm," *Computers & Structures*, vol. 169, no. 2, pp. 1–12, 2016.

[20] T. J. McCabe, "A complexity measure," *IEEE Transactions on Software Engineering*, vol. SE-2, no. 4, pp. 308–320, 1976.

[21] A. Panichella, F. M. Kifetew and P. Tonella, "Automated test case generation as a many-objective optimisation problem with dynamic selection of the targets," *IEEE Transactions on Software Engineering*, vol. 44, no. 2, pp. 122–158, 2018.

[22] S. A. Khan and A. Nadeem, "Automated test data generation for coupling based integration testing of object oriented programs using particle swarm optimization (PSO)," in *Genetic and Evolutionary Computing*. Cham: Springer, pp. 115–124, 2014.

[23] Z. K. Aghdam and B. Arasteh, "An efficient method to generate test data for software structural testing using artificial bee colony optimization algorithm," *Int. Journal of Software Engineering and Knowledge Engineering*, vol. 27, no. 06, pp. 951–966, 2017.

[24] S. Laokok and T. Suwannasart, "An approach for test case generation from a static call graph for object-oriented programming," in *Proc. Int. Multi Conf. of Engineers and Computer Scientists*, Hong Kong, vol. 1, 2017.

[25] Y. Chen, Y. Zhong, T. Shi and J. Liu, "Comparison of two fitness functions for GA-based path-oriented test data generation," *Proc. IEEE Int. Conf. on Natural Computation*, vol. 4, pp. 177–181, 2009.

[26] J. Wegener, A. Baresel and H. Sthamer, "Evolutionary test environment for automatic structural testing," *Information and Software Technology*, vol. 43, no. 14, pp. 841–854, 2001.

[27] N. Tracey, J. Clark, K. Mander and J. McDermid, "An automated framework for structural test-data generation," in *Proc. IEEE Int. Conf. on Automated Software Engineering (Cat. No. 98EX239)*, Honolulu, HI, USA, pp. 285–288, 1998.

Automated Software Engineering

Source Code Change Analysis with Deep learning based programming model

--Manuscript Draft--

Manuscript Number:	AUSE-D-21-00066R1
Full Title:	Source Code Change Analysis with Deep learning based programming model
Article Type:	S.I. : Software Engineering for Parallel Programming
Keywords:	Change Impact Analysis, Abstract syntax tree, Path2Vec, Deep learning, word embedding, distance metrics, attention, un-parsing.
Corresponding Author:	Babita Pathik, Phd Scholar DAVV: Devi Ahilya Vishwavidyalaya Indore, INDIA
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	DAVV: Devi Ahilya Vishwavidyalaya
Corresponding Author's Secondary Institution:	
First Author:	Babita Pathik, Phd Scholar
First Author Secondary Information:	
Order of Authors:	Babita Pathik, Phd Scholar
	Meena Sharma, PhD
Order of Authors Secondary Information:	
Funding Information:	

Reviewer #1:

1. **In section 4, paragraph 3, the authors explained the flow of their process in four parts. Instead of writing steps as 'one', 'two' etc., they should give a number or can write step 1, step 2, etc., for clarity.**

   Response: *Done*

2. **In figure 4, two parts (a) and (b) are there, but the authors did not consider them separately in the paragraph. It is suggested to mention them in the text.**

   Response: *Done*

3. **ALGORITHM 1 and ALGORITHM 2 caption is missing. Give proper caption based on the task of these algorithms.**

   Response: *Captions are added*

4. **Some references are not in the prescribed format. The formatting of all the references should be uniformed.**

   Response: *Reference formatting is Done*

5. **The author should add some latest works of literature in the paragraphs of the introduction section, especially from 2019-20.**

   Response: *Latest research papers are added.*

6. **Sub-heading 5.2 'Results' should be renamed according to the experimental evaluation like the other paragraph's heading. That looks appealing.**

   Response: *Done*

7. **I recommend the authors to proofread the manuscript with a native English speaker.**

   Response: *Done*

8. **Discuss the comparison of the accuracy of the relevant studies**

   Response: *A comparative table is added.*

Review #2:

1. **Add some more references in support of the work in the literature review section. Briefly explain these references in that section that will show the novelty of your work.**

   Response: *Latest references are added along with the limitations of existing work in tabular form.*

2. **The meaning of variables is not clear. Readers will be confused. The authors should explain them properly.**

   Response: *Detailed explanation is added.*

3. **Write the caption and number separately for figure 11, as it seems the caption is missing.**
   Response: *Done*

4. **Authors have used abbreviations in some places and their full form somewhere. Once the abbreviation is declared, use it throughout the paper where it is required.**

   Response: *Correction is done.*

5. **Discriminate the use of LSTM and Bi-LSTM for the proposed approach.**

   Response: *Done*

6. **In figure 3, a very small code segment has been taken; why? Give reason for that.**

   Response: *The proposed model is also applicable to large program code, but we portray results on small code segments due to the large space requirement for graphical representation of ASTs.*

7. **The conclusion part should include result-specific data or focus on the outcome of work done.**

   Response: *The results are included in the conclusion part.*

Review #3:

1. **The drawbacks of conventional techniques should be described clearly. The authors should emphasize the difference with other methods to clarify the position of this work further. The author can prepare a table that will help to understand.**

   Response: *Information is added in a tabular form. Refer Table 1.*

2. **The attention mechanism should be explained in more detail, supporting your work. A separate sub-section can be added in the main section.**

   Response: A *separate section is added explaining the Attention mechanism (section. 4.6.1)*

3. **What are the values for tuning Hyperparameter? Specify the values for the parameters that are mentioned in the last paragraph of section 5?**

   Response: *Hyperparameter values are mentioned in the new paragraph.*

4. **In figure 6, the description of the node's type is missing. It will better to use a table in place of the figure to describe the nodes. For example, ClassDef → ?????.**

   Response: *The description of Figure 6 is shown in the form of a table. (refer Table 2)*

5. **In section 3, subsection 3.1 abbreviation EIS is not described before. Although before using the abbreviation, its description must be written.**

   Response: *Done*

6. **The term IABLSTM is missing from the abstract or title. If it is significant in this paper, it should be added in the abstract part. The authors should write its expanded form also for more clarity.**

   Response: *Done*

7. **Authors are advised to present a discussion on the accuracy of similar studies.**

   Response: *A detailed description is added in the form of a table. (refer Table 5)*

Reviewer #4:

1. **In this manuscript, the contributor has not emphasized how the Deep Learning Framework is utilized to handle industrial manufacturing. A deep learning framework is provided in diagram 1, but its incorporation with the testing and its relevant information is missing.**

   Response: *Thanks for your suggestion. Deep Learning Framework is fit for industrial manufacturing for a reason dealing and manufacturing substantial data. This is useful where these data are used efficiently. As far as our paper is concerned, the results of the framework can be considered for regression testing, but we have not extended it in this article. We have focused on finding code changes and their impacts.*

2. **How the two purposes achieve it with the loss function is completely loaded with the equations. Therefore, it is not easy to follow. Applying the operational flow of approach Path2Vec provided in a diagram appears well, but its implementation in this manuscript and its functional description are not justified.**

   Response: *Done*

3. **Moreover, the usage of the algorithm along with the data by several processes they are general information storage, implementation of the user interface, privacy design, and support design and its necessary information, are not presented well.**

   Response: *The proposed algorithm is specifically covering the methodology to achieve our objective of change detection. We have not covered the above mentioned area of software products in this article.*

4. **Besides, there is no clear description provided for how the Effective Inspection System for the smart manufacturing industry and its discussions are not considered in this manuscript; therefore, it deficits in studying the central theme of this work.**

   Response: *The idea presented in this article will facilitate the software maintenance process in the information technology industry. Software engineers and developers also take help for regression testing.*

5. **In the result analysis, how the Receiver Path2Vec is used to evaluate the rate and its necessary information is not provided. Furthermore, how the blob, edge, corner, and spots are depicted in diagram 8 are not elucidated.**

   Response: *As two additional diagrams are added as per review, so Diagram 8 has become diagram 10. It is updated with the necessary information. In this presented work, IABLSTM is evaluated as a whole on accuracy rather than Path2Vec separately. Rate is not considered as evaluation.*

6. **The conclusion does not summarize how the images are captured to analyze the manufacturing system's deficits efficiently.**

   Response: *This review seems to be irrelevant with respect to the presented work as we have neither captured any images and nor analyzed any manufacturing system.*

Review #6

1. **To support the novelty of the presented work, a clear comparative study is required for a crisp conclusion. Need to refer to more recent papers to fulfill the said task.**

   Response: *Recent papers are added (2019, 2020). A table is added showing the limitations of various existing methods.*

2. **Hyper-parameter tuning is specifically not mentioned. A detailed description is equired for more clarity.**

   Response: *Detailed description about Hyper-parameter tuning is added.*

3. **A comparative analysis based on the performance of the proposed model must be provided by the authors.**

   Response: *A table is added for comparative analysis of presented work with existing methods based on performance.*

4. **In fig.2 LSTM network is missing from the overall working flow, even though it is a part of the proposed approach, that should be included in the given figure.**

   Response: *Figure 2 is modified with said content.*

5. **The description of variables or the meaning is not given properly. The description gives a better understanding and authors must focus on this point.**

   Response: *All the considered variables are described for understanding.*

6. **Authors have given Figure 1 for the LSTM network. However, the given figure seems to be a standard figure. Authors are required to give an explanation about the figure in the interest of the readers of the journal. As many parameters are given in the figure, authors are required to give a short explanation about them.**

   Response: *The explanation of Figure 1 with all parameters and variables is done*.

7. **In the text authors have mentioned "Figure 1 depicts the overall architecture of the LSTM network". However, this representation could be better in reference to the author's work.**

   Response: *We have replaced the text with a better statement.*

8. **Authors have used multiple abbreviations. Either authors should prepare the table of abbreviations or each abbreviation must be written in full form at its first instance.**

   Response: *All the abbreviations have been elaborated before their use*.

9. **Figure 2 has a caption as "Figure 2. Overall working flow of our approach IABLSTM". I recommend that this caption can be more comprehensive to enlighten the details.**

   Response: *We have updated the caption in detail in Figure 2.*

10. **There are divisions of Figure 4 (a) and Figure 4 (b) and a separate caption is given. However, I recommend that there should be only one caption for Figure 4 where both the captions can be referred using (a) and (b) and similarly figures can be referred through (a) and (b). This is similar to the Figure 3 used in the manuscript.**

    Response: *Only one caption is written for Figure 4(a) and (b)*

11. **Equation 10 is used in the text. But what about other equations? Other equations should also be used in the text.**

    Response: *All the equations (1-13) are used and cited in the text.*

12. **Authors are required to discuss what is BOW, TF, or TF-IDF and how these are used?**

    Response: *We have not evaluated the considered program and dataset on above said technique. We have applied the concept of Word2Vec to proposed a new concept of Path2Vec in this article. We have mentioned the background of all of these terminologies in section 3, 2nd paragraph.*

13. **The authors are required to discuss the available relevant methods.**

    Response: *Done in the form of tables. (refer Table 1)*

14. **The authors are required to compare their results with the current state-of-the-art methods.**

    Response: *Done in the form of tables. (refer Table 5)*

15. **Include more relevant work in the related work section.**

    Response: *Done*

16. **Overall recommendation is to improve the language to increase the readability of the paper.**

    Response: *Done*

Review #7

1. **The difficulties faced in the existing system based on object-oriented based parallel programming source code have not depicted along with their constraints.**

    Response: *Done (refer section 7)***.**

2. **When compared to Path2Vec, how much accuracy is achieved by the conventional models despite of generating abstract syntax tree?**

    Response: *Done (refer Table 5).*

3. What are the characteristics of change impact analysis (CIA) software involved in identifying the changes?

    Response: *Static Code analysis is performed on object-oriented method as granularity level.*

4. **If the software change is implemented after the maintenance, then how do the change impacts the backend and frontend source code and what are the drawbacks that occur due to the impact made on source code?**

    Response: *As we have proposed a methodology for change detection if it occurred and applied it to small code segments. We have also applied our approach to the AST dataset to verify the correctness of the model proposed, but we have not considered or coordinated with backend and frontend source code. Source code change impact analysis is part of the maintenance process, and it definitely affects the related modules (backend and frontend). Added this point in section 8 as future work.*

5. **lack of information concerning the structure of source code tree illustration utilized by the Abstract syntax tree (AST).**

   Response: *Done*

6. **While representing about the contribution of this paper related to the change detection in source code Bi-LSTM is suggested but in abstract section only Path2Vec approach is described what about the Bi-LSTM method?**

   Response: *Done*

7. **The author did not state whether the proposed Word2vec word embedding technique is either a supervised or unsupervised model.**

   Response: *Done*

   Although word embeddings are considered unsupervised, they are trained using a fictitious supervised learning problem.

8. **There is no sufficient information provided for the existing system LSTM and MLP architecture for processing abstract syntax tree.**

   Response: *Done*

9. **How does the tokens sequence is generated for the input data and how does it maps into vector through embedding layer?**

   Response: *Done*

10. **Comparative data between several word embedding techniques BOW, TF-IDF, Word2Vec have not elucidated in this paper.**

    Response: *We have not evaluated above said technique on considered data. We have applied the concept of Word2Vec to proposed a new concept of Path2Vec in this article.*

11. **Working procedure of Bi-LSTM neural network approach is not explained well and seems to be difficult to understand without testimonials. The function of softmax activation layer and its benefits is not represented in this paper.**

    Response: *Explained with separate section.*

Source Code Change Analysis with Deep learning based programming model

Babita Pathik · Meena Sharma

Abstract Analyzing the change in source code is a very crucial activity for object-oriented parallel programming software. This paper suggested an Impact analysis method with Attention BiLSTM (IABLSTM) for detecting the changes and their affected part in the object-oriented software system. Classical approaches based on control flow graph, program dependence analysis, latent dirichlet allocation, and data mining have been used for change impact analysis. A Path2Vec approach is presented in the paper, combining a deep learning technique with word embedding to analyze and identify the change. The paper considers two versions of a python program for experiment and generates the abstract syntax tree (AST). Then extract the path to produce a token sequence. Next, convert the token sequence into unique vectors by applying a word embedding layer. The BiLSTM network encodes the sequence into a vector representation. After that, compare the embedded output with the use of cosine distance metrics. We trained the neural network model with the embedded outcome. Then decode the resultant token sequence into a path of AST. Finally, convert the AST path back to code using the un-parsing technique. To strengthen the parallel programming based proposed model, we combined the attention mechanism to emphasize and detect the differences in the code. The model is detecting the change of code efficiently. The experimental results show that our proposed model's change detection accuracy increases significantly compared with other conventional models for change impact analysis. The proposed method can also be applied for impact analysis on object-oriented based parallel programming. The empirical evaluation shows that the model outperforms change detection with approximately 85% validation accuracy.

Keywords: Change Impact Analysis, Abstract syntax tree, Path2Vec, Deep learning, word embedding, distance metrics, attention, un-parsing.

Babita Pathik (✉)
IT, Institute of Engineering & Technology, DAVV, India
e-mail: babitapathik@gmail.com

Meena Sharma
Department of Computer Engineering, Institute of Engineering & Technology, DAVV, India
e-mail: msharma@ietdavv.edu.in

1 Introduction

Due to the continuous growth of computer based applications and usage, the maintenance of the software has become a crucial task. For reducing the maintenance cost and effort, it is necessary to find the changes and their impact on any part of the code very efficiently. Software change impact analysis enables testers to reduce the maintenance cost by identifying changes and their impact on the code. Once the change is implemented in code, it can affect anywhere any part of the code. This effect is commonly known as the ripple effect. The change in software may cause many side effects. Sometimes it may cause errors too. The object-oriented program comprises influential behavior. The object-oriented paradigms have programming concepts, e.g., polymorphism, inheritance, abstraction and encapsulation. The change analysis techniques take these concrete features of object-oriented programs into account and generate potentially impacted classes, class methods or class fields. In order to help testers and developers, it is reasonably necessary to analyze the changes and their impact on code efficiently and accurately. It may reduce testing costs up to some extent. Change Impact Analysis(CIA) of software is a process of finding changes and their potential impacts on the part of the software system (Bohner et al. 1996).

This paper focuses on finding all the changes in the revised version of the software by evaluating both versions using a parallel programming model. The presented work will facilitate the software maintenance process in the information technology and software industries. The model framework is useful for software engineers to perform regression testing. The granularity of a program may be file level, code level, function level, change level. We choose the code level of the program for an experiment in this paper. The CIA technique suggested by various researchers for static code analysis gives the approximate result. Metrics like McCabe and CK are not sufficient to analyze the basic structure of the code. Traditional methods like software edit history (Kitsu et al. 2013), repositories mining (Moldrez et al. 2017), Control call graph (Badri et al. 2005), Program Dependence Graph (PDG) (Baah et al. 2010), Aspect-Oriented Dependence Flow Graph (Ahmad et al. 2014) considered syntax structure and its relationship among function and classes like elements. But these methods did not consider semantic information instead.

There are some more techniques used by various researchers, such as LDA (Thomas et al. 2010), Latent Semantic Indexing (LSI) (Gethers M. et al. 2011). These improved the performance of code change analysis up to some extent. A deep learning model with a framework of encoder-decoder is applicable in source code generation and its modeling, according to Le TH et al. (2020). Meng and Liu (2020) present a BiLSTM network with a self-attention layer to detect source code.

The syntactic and semantic information are the features that contain such structural and semantic information that may advance the performance of impact analysis. An AST of every program has detailed semantic information and conceives its syntactic structure (Alon et al., 2018, Wang W et al., 2020). A neural network based on AST has been developed (Zhang J. et al., 2019). The change and change impact can be searched and analyzed more accurately with the help of the AST of the program code. The semantic information helps to find the difference between two different versions of the code. Some change does not affect other code. Those set of change is termed as the actual impact set (AIS). That can be traceable by semantic information. Extract all the paths from the generated AST for further process. The paths are extracted as a whole then each of which is separated by splitting them. Each path is converted in vector representation.

Following are the significant contributions in this paper:

(1) We suggest a BiLSTM based change detection that learns useful features related to the source code's semantic and syntactic information. BiLSTM with attention is used for training and named IABLSTM.
(2) We do parsing on a program to evaluate the syntax tree and follow the path of the tree in a depth-first traversal manner. Split the path to generating the tokens.
(3) We apply Path2Vec that uses the Word2vec word embedding technique to convert code’s ASTs into high dimensional real-valued vectors taken as input to the LSTM based model for training.

The rest of the paper is organized as section two briefly summarizes the work achieved by various researchers in CIA and software code analysis. The background of our work is mentioned in section three. The fourth section describes the proposed methodology. The fifth section contains the experimental setup with results, and finally, the sixth section concludes the paper with expected future work.

2 Literature Review

The motivation behind this work is that if changes and their consequences can be detected efficiently, the testing and maintenance costs can be reduced proportionally. It is always a thrust area of research due to the industry's need. Here we brief the work presented by the author, which we have referred to in our work. Table 1 briefs the approaches and threats studied from the above articles. Angerer et al. (2019) presented a CIA approach, which determines the source code's impacted element. They used to control flow and data flow analysis for their approach. Goknil A et al. (2016) utilize semantics of requirements relations, change requests and traces between requirements and architecture to improve CIA in software architecture. Musco V et al. (2016) present LCIP, a learning system that uses historical data to forecast future impacts. The artifacts investigated for CIA are object-oriented software methods. A multi-level word and character embedding were adapted to record the semantics of code modifications and reviews. The embeddings are trained using a suggested attentional deep learning model (Siow JK et al., 2019). Tiwang et al. (2019) suggested a deep learning model for source code generation and completion. AST is processed for structure evaluation of source code. They utilize LSTM, deep learning, and MLP architectures to generate the model.

A learning-based approach is presented for detecting code clones. Code analysis enables the automatic connection at lexical and syntactic levels of patterns mined with a system based on deep learning (White M et al., 2016). Using source code analysis techniques, Eid S et al. (2020) proposes a novel approach to automatically identify probable code changes that result in performance degradation between system versions. A deep learning method is combined with word embedding in a framework for predicting the program's defect (Liang et al. 2019). Token sequence extracted from the generated AST. These tokens are mapped with a real-valued vector using a mapping table. They provided unsupervised training using the word embedding model and LSTM network using vector sequences and labels.

Table 1. Proposed approach and limitations

Author	Proposed Approach	Limitations
Goknil A. et al. (2016)	To identify architectural aspects for the change impacts in architecture requirements, employ the formal semantic of requirement relations and traces between Requirement & Architecture.	Returns the impact on new requirement only if an existing requirement relating to a new requirement exists.
Angerer et al. (2019)	Configuration-aware CIA and interprocedural approach using conditional system dependence graph	The assessment is based on a less customizable code base; some alternative functionalities are not included. It could not be applied to full code.
Musco V et al. (2016)	CIA for object-oriented methods artifacts presented by considering Class-Hierarchy-Analysis call graph	Valid only for Java software, even only for the studied projects.
Siow JK et al.(2019)	Developed a multi-level word and character embedding technique to express the semantics of code modifications and reviews.	Negative data may exist in the training set, while actual positive data is available in the test set. The model is learning some sections of the test set in the training phase.
Eid S et al. (2020)	Identify probable code changes for test cases with a genetic algorithm-based performance tool.	The approach has one basic problem: it cannot identify newly added, instead of updated, source code performance regressions.

Dam HK et al. (2018) designed a prediction model capable of automatically learning and utilizing attributes for representing and predicting defects in source code. The approach is based on a sophisticated deep learning, tree-structured LSTM that corresponds directly to the source code's AST representation. LSTM with an attention mechanism based intelligent model proposed by Rahman et al. (2020) for source code completion. The model classifies the erroneous source code and clean code. The goal of the work is to detect the error in code line by line. Meng and Liu (2020) presented a unique model for detecting source code with a self-attention layer based on a BiLSTM network. The model encodes the series of statement vectors using the model, which is a well-known deep learning framework. The model maintains both the syntactic and semantic properties of the source code during the encoding process.

From the briefing of the literature survey, we can conclude that syntactic and semantic features help source code analysis. The method proposed here is generalized and applicable in any source code except parsing of the code. In this work, we get an AST to extract both the features information. Our model, unlike others, is using BiLSTM with self-attention for detecting the change code.

3 Background

3.1 Change Impact Analysis

Change Impact Analysis is a crucial activity in the software development process due to software evolution. There are so many changes that emerge in software during maintenance. The conception of the CIA is to recognize the changes and their side effects. The impacted part needs to be analyzed effectively to reduce maintenance costs. For large scale software, the CIA is a rigorous task. Assessment of Estimated Impact Set (EIS) is the CIA's primary goal, and it must be as close to AIS. Several researchers suggest various methods to detect impacted parts in source code, including the control call graph and other CIA techniques.

3.2 Abstract Syntax Tree

Abstract Syntax Tree (Büch et al. 2019) is a mode of representing the source code in graphical notations. AST is used by compilers, which reads the code by parsing it and generate the object binaries. It is a tree that represents the abstract syntactic structure of a selected source code. AST completely restores the structural information of the given source code. Each node of the tree corresponds to the essential elements of the code. It efficiently characterizes the programs with any source code and is widely adopted in the software engineering field. AST holds syntactic and semantic information in an exemplary manner and is frequently used by researchers and IT industries. The purpose is to extract hidden information from the code. In this paper, we traverse the tree to process the path.

3.3 Word Embedding

Word embedding is a broadly accepted technique for text in the field of natural language processing. (Hoang et al. 2020). It is a vector representation of words of given documents. Word embedding is mainly a feature extraction technique used in text processing (Hameed et al. 2020). A vector of real value represents each word of the document. We use embedding to encode the token sequence and map it into a vector. Then these vectors are required as input for our model. There is various embedding technique we have gone through, such as BOW, TF-IDF, Word2Vec. We add an embedding layer to our network to extract features.

The BOW model is often employed in the categorization of documents, and each word is used as a training feature for a classifier. BOW doesn't work very well if statements have the same meaning but only with different terms. TF-IDF is a statistic that reflects the importance of a word in a corpus of documents. Word2Vec model is used for vector representations of words known as word embedding. This is done in a preprocessing phase through which the acquired vectors are put into a NN model for predictions and some other task. This model utilizes the semantics of words. We extend the working of the Word2Vec model for our method accordingly.

The purpose of this training is to drag the semantic difference between paths. Embedding layer added as part of a neural network model. The layer is usually built for dealing with natural language processing tasks, specifically classification, language modeling. The document should be preprocessed for encoding. The vector size is specified with a particular dimension such as 50, 100, or 300 and becomes part of the model. One-hot encoded word is mapped to the word vectors through mapping. The network takes the encoded input sequentially.

The widely used embedding method is Word2Vec, an efficient statistical method (Meng and Liu, 2020). Refer the program to as a text document. Skip-gram and Continuous Bag of words (CBOW) are the two architectures on which Word2Vec works. The word w_p is predicted in CBOW, if wp-2, wp-1, wp+1, wp+2 are given context or words. In this work, a token is a path sequence that is a combination of nodes. We adopt the Word2Vec word embedding technique to encode the targeted token. The basic reason for selecting this technique is to retain the order of the words intact. It is the way we can maintain the sequence of the token

3.4 LSTM Network

LSTM is a gated version of RNN (Liang et al., 2019). It is well suitable to process sequential data. LSTM clenches mainly three gates as input gate (i^t), forget gate (f^t), and output gate (o^t). The working of the cell state is transmitting relevant data through a sequence forming a chain. The cell state remembers the material that comes out of the previous time step during the processing in sequences. Data is collected or erased into the cell state via gates as far as the cell state comes into the movement. The neural network-integrated gates determine which cell state data is enabled.

The first gate is the "forget gate," which decides which information should store and discard. The current state input and information from the previously hidden state pass by the sigmoid function. The function returns the value between 0 and 1. If the value it returns is nearer to 0 means inhibiting the information, and if the value is closer to 1, keep and pass all information. We get the value by equation (2).

We've got the input gate to modify the cell state by equation (1). To further control the network, transfer the current input and hidden state output through the *tanh* function, which ranges the output values from -1 to 1. Then the sigmoid result is multiplied with the *tanh* output. To calculate cell state, we received sufficient information. First, the pointwise multiplication is held between the cell state and forgot vector. If values compound it near 0, it drops values in the cell state. Next, take the input gate's output, and pointwise addition is applied, which upgrades the cell state with new values found appropriate by the considered neural network. The state is our new cell state and evaluated using equations (4) and (5).

The output gate is appropriate to determine the next hidden state using equation (3). For predicting the value, the hidden state is having information about previous inputs. First, new input and the previously hidden state transfer into the sigmoid function. Then transfer the recently changed cell state to the *tanh* function. The sigmoid output is multiplied with *tanh* output to determine hidden state data with equation (6). The hidden cell and new cell state are then passing to the subsequent stage. Fig. 1 depicts an internal architecture of the LSTM cell.

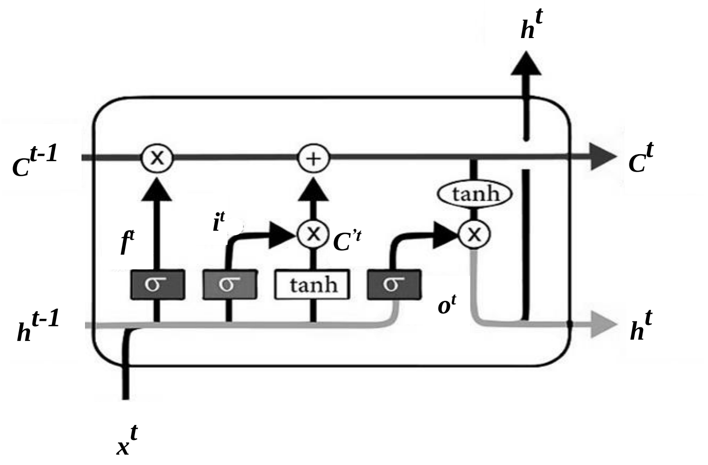


Fig. 1 Internal working of an LSTM cell

The equations for the gates in LSTM are:

Input gate	$i^t = \sigma\,(w^i[h^{t-1}, x^t] + b^i)$	(1)
Forget gate	$f^t = \sigma\,(w^f[h^{t-1}, x^t] + b^f)$	(2)
Output gate	$o^t = \sigma\,(w^o[h^{t-1}, x^t] + b^o)$	(3)

σ = Sigmoid function
w^i/ w^f / w^o = Weight for the input/forget/ output gate neuron
h^{t-1} = Output of the previous hidden state is at time *t-1*
x^t = Current state input, i.e., at time-stamp *t*
b^i/ b^f /b^o = Biases for the input/forget/ output gates

New Candidate for cell state at time *t* :	$C'^t = tanh\,(w^c[h^{t-1}, x^t] + b^c)$	(4)
Cell state at time *t* :	$C^t = f^t \times C^{t-1} + i^t * C'^t$	(5)
Final output at *t* :	$h^t = o^t \times tanh(C^t)$	(6)

C^{t-1} = Cell state at time *t-1*
w^c = weight for a new candidate
b^c = Bias for a new candidate

With cell state and the gates mentioned above, gratuitous information is automatically dropped by LSTM cell as the time step grows. LSTMs are widely used in software engineering problems (Liang *et al.*, 2019, Meng and Liu, 2020) and natural language processing (Hameed and Garcia, 2020), where semantic relation is essential. In the paper, we train the model using BiLSTM to learn source code.

3.5 BiLSTM

BiLSTM is an advanced form of RNN. These can significantly increase model performance when used to solve sequence classification issues. The model consists of two LSTMs: one that takes the input in one direction and the other in the opposite direction (forward and backward direction). BiLSTM significantly improves the quantity of data presented to the network, which benefits the algorithm's context.

The BiLSTM is designed to achieve the objective of long term dependence at the moment around t. Source code comprises contextual information that is important to spot potential issues. Each program has its own context-sensitive syntax and semantics. The emergence of a different code segment is normally relevant to both preceding or succeeding code. In the implementation of BiLSTM, bidirectional processing runs given inputs in two directions; forward direction leads from past to future, and backward is from future to past. We use this model two combined hidden states; one can preserve information from both the past and future at any point in time.

3.6 Attention Mechanism

We can obtain hidden features of all time nodes in a series from the output of the BiLSTM network. We insert an attention layer after the Bi-LSTM layer in order to improve the influence of crucial nodes. When the attention mechanism is applied, critical nodes that are important for the sequence are aggregated together to produce a sequence vector.

4 Proposed Approach

This section of the paper describes the overall working of our proposed approach. Multiple version of source code is taken for experiment purpose to analyze the change. Our proposed IABLSTM is a system that automatically traces syntactic features and semantic information from the source code through parsing to extract key features to find change code.

Very first, we apply preprocessing on input source codes which involve the removal of comment lines. Our target is to train the model with a different version of the same program with or without error and, after that, successfully use it to detect changes by giving another version of the same program. Fig. 2 demonstrates the proposed method of IABLSTM.

The LSTM based approach has two working parts. The first one is the representative part, and the second is the discriminative part. In the first part, we represent the code in dimensionality vector, which is achieved with the following steps:

Step 1: Both versions' selected python source code is presented into ASTs by utilizing available AST tools. These ASTs generate at the statement level.

Step 2: We perform pre-order traversal on the tree to extract path sequence with a granularity of statement. Now we split the sequence path up to the leaf nodes.

Step 3: According to the deep-first traversal technique, each split path sequence is converted into a real-valued vector with an LSTM encoder's help. All path fragment is transformed into an ordered set of path vectors.

Step 4: The proposed model generates a final characteristic vector from a path vectors sequence through a bidirectional LSTM (BiLSTM) encoder with the self-attention layer. A set of code vectors calculated by applying cosine distance using this model is further loaded into the designated model. The model predicts the probability of change for the pair of codes.

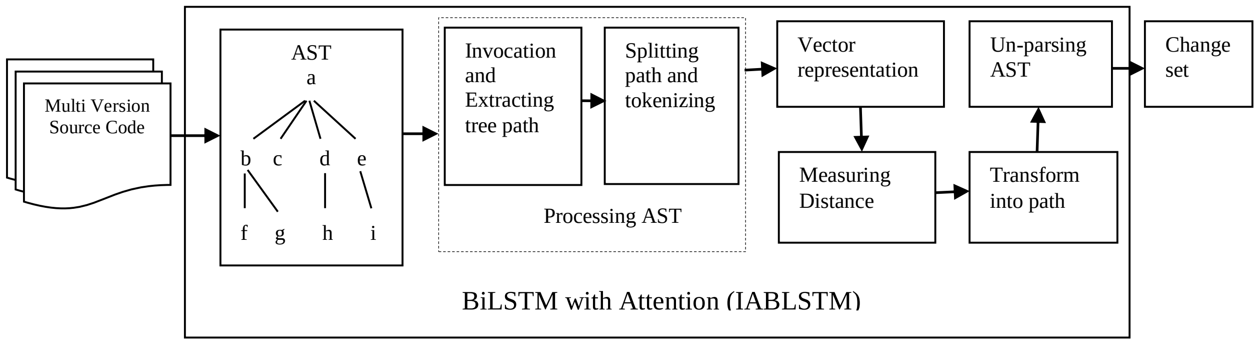

Multi Version Source Code
AST
a
b c d e
f g h i
Invocation and Extracting tree path
Splitting path and tokenizing
Processing AST
Vector representation
Measuring Distance
Transform into path
Un-parsing AST
Change set
BiLSTM with Attention (IABLSTM)

Fig. 2 A process diagram for the proposed IABLSTM approach with AST.

4.1 Code Preprocessing

Before training the model, we filtered raw source codes by removing unnecessary items. First, all irrelevant elements have been eliminated from the code, such as new lines, comments (#), and tabs (\t). Then, every remaining code element like keywords, numbers, functions, variables, classes, and characters have been translated into sequences of terms. The filtered code was parsed through the parser.

4.2 Source Code Parsing

AST is a syntactical structure representation of a code as a tree. It is an intermediate representation of a program during the construction of a compiler. The construction of AST is a part of parsing that is a syntactic analyzer.

```
class EleVehicl(Vehicl):
    def __init__(self, make, model, type):
        self.chrg_lvl = 0
```

(a)

```
class EleVehicl(Vehicl):
    def __init__(self, make, model, type):
        self.chrg_lvl = 0
    def charg(self):
        self.chrg_lvl = 100
        print('charged')
```

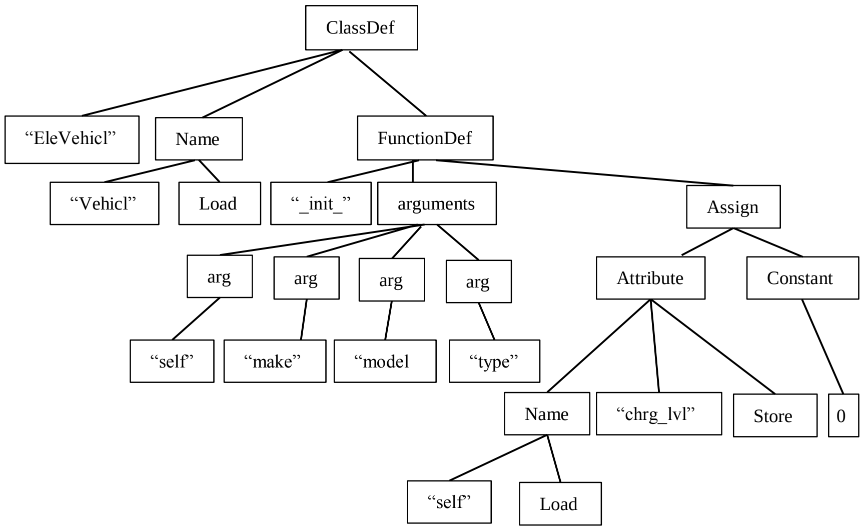
(b)

Fig. 3 Python programs for charging vehicle (a) first version (b) revised version

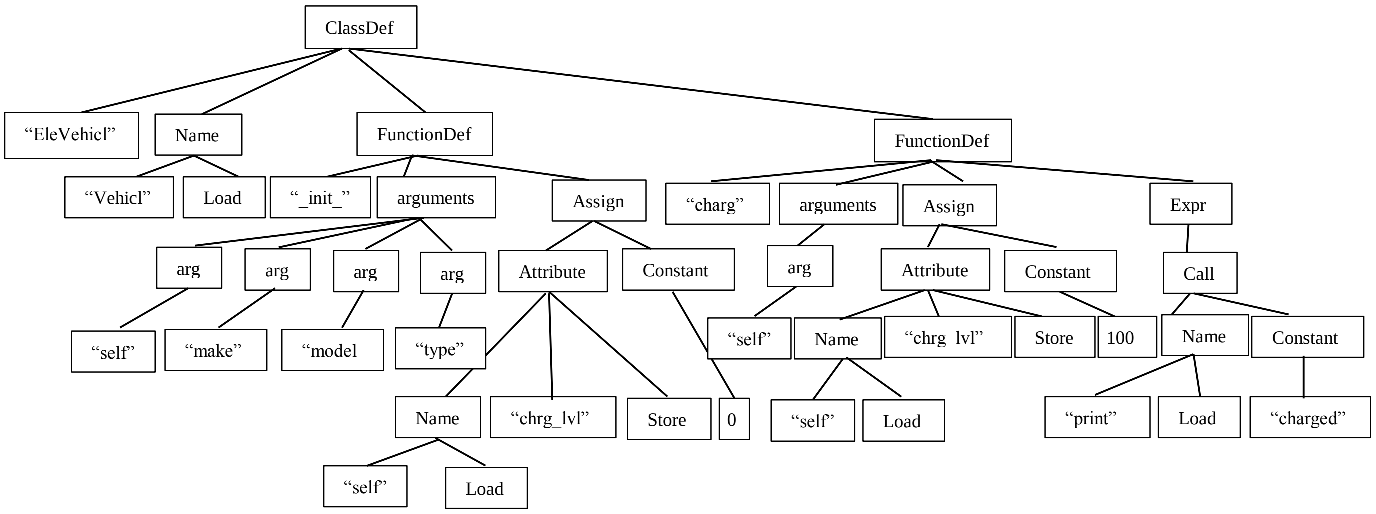
We generate AST for a different version of a program to extract the various existing path. The semantic analyzer utilizes the information provided by the AST. Every path ends with a leaf node. The leaf node or terminal node contains tokens of the program code.

There are two small python codes given in Fig. 3(a) and 3(b) for the first and second versions, respectively. Fig. 4(a) and 4(b) depict the AST of code given in Fig. 3(a) and 3(b), respectively. The proposed model is also applicable to large program code, but we portray results on small code segments due to the large space requirement for graphical representation of ASTs.

In Fig. 4(a) and 4(b), the nodes of the tree comprise keywords, characters. These fundamental elements of the source code are tokens. The parent-child node pair can also be extracted from AST. A parent-child relationship is valuable when it is required to trace back for analysis. There are various types of nodes that emerged in AST. Each node shows either a code component or a specific characteristic of code. Every tree starts with a module as a root node.



(a)



(b)

Fig. 4 AST for (a) first version (b) revised version

Fig. 5 enlisted the character hold by the nodes of a tree. The list may vary, here we include some of them.

4.3 Processing of AST

Path extracted from AST is considered for further processing. Visit every node of the tree through pre-order. We traverse the nodes of the tree in the depth-first traversal technique. After traversing, the path is extracted as shown in Figures 6 and 7 for source codes shown in figures 3(a) and 3(b), respectively. It shows the output as a complete string. To convert this string into words, we need to break it down to generate tokens. The paths are separated by splitting with the terminal node, which is having variable path length. These separated paths are tokens that construct a sequence vector. All the nodes, including the terminal node of each truncated path, show the program code's characteristics or code elements. Algorithm 1 in Figure 8 explains the generating and parsing of the source code.

ClassDef	Module	Alias	Store	Store
NameConstant	comprehension	ListComp	Return	Num
FunctionDef	Assign	Slice	Constant	Tuple
Name	Call	For statement	Mult	NotEq
Arguments	str	While statement	AugAssign	Eq
Print	Expr	If statement	List	Mod
BinOp	True	Gt	Attribute	LtE
Load	Subscript	GtE	Sub	Add
Compare	arg	Lt	Div	Index

Fig. 5 Types of nodes in AST

4.4 Path2Vec

All the separate paths transform into a vector representation. Paths are tokens and in the form of strings that cannot be used as input directly. So, first of all, the tokens are converted into the token id. Then the token ids are transformed into a vector. Vectors are of varying length of n number of tokens. The token ids are unique integer values. Vectorization is achieved through the encoding technique by using word embedding.

```
["ClassDef(name='EleVehicl', bases=[Name(id='Vehicl', ctx=Load())],"
 "keywords=[], body=[FunctionDef(name='__init__',"
 "args=arguments(args=[arg(arg='self', annotation=None), arg(arg='make',"
 "annotation=None), arg(arg='model', annotation=None), arg(arg='type',"
 'annotation=None)], vararg=None, kwonlyargs=[], kw_defaults=[],   kwarg=None,'
 "defaults=[]), body=[Assign(targets=[Attribute(value=Name(id='self',"
 "ctx=Load()), attr='charg_level', ctx=Store())], value=Num(n=0))],"
 'decorator_list=[], returns=None)], decorator_list=[])']
```

Fig. 6 Extracted path for code in Fig. 3(a)

```
["ClassDef(name='EleVehicl', bases=[Name(id='Vehicl', ctx=Load())],"
 "keywords=[], body=[FunctionDef(name='__init__',"
 "args=arguments(args=[arg(arg='self', annotation=None), arg(arg='make',"
 "annotation=None), arg(arg='model', annotation=None), arg(arg='type',"
 'annotation=None)], vararg=None, kwonlyargs=[], kw_defaults=[], kwarg=None, '
 "defaults=[]), body=[Assign(targets=[Attribute(value=Name(id='self',"
 "ctx=Load()), attr='charg_level', ctx=Store())], value=Num(n=0))],"
 "decorator_list=[], returns=None), FunctionDef(name='charg',"
 "args=arguments(args=[arg(arg='self', annotation=None)], vararg=None, "
 'kwonlyargs=[], kw_defaults=[], kwarg=None, defaults=[]),'
 "body=[Assign(targets=[Attribute(value=Name(id='self', ctx=Load()),"
 "attr='charg_level', ctx=Store())], value=Num(n=100)),"
 "Expr(value=Call(func=Name(id='print', ctx=Load()), args = [Str (s = 'charged')],"
 'keywords=[]))], decorator_list=[], returns=None)], decorator_list=[])']
```

Fig. 7 Extracted path for code in Fig. 3(b)

The description of the tree component as nodes are given in Table 2.

Table 2. Description about tree components

Nodes	Description
ClassDef	Class definition
args	arguments
FunctionDef	Function definition
vararg	Variable argument
attr	attribute
kwonlyargs	lists of arg nodes
kw_defaults	Default keywords
Expr	Expression
Str	String
kwarg	passed arguments by keyword
ctx	Store an assignment
Fun	function

```
Algorithm 1:
Input: Two Version Source Codes $S = \{S_i, S_{i+1}\}$
        Set of featuring nodes $F = \{f_1, f_2, f_3, ...f_n\}$
Output: Path List $P = \{P_i, P_{i+1}\}$
        Vector $V = \{V_i, V_{i+1}\}$ for $\{S_i, S_{i+1}\}$
for all source files do
  for $i = 1$ to $n$ do
    $AST_i$ = Generating $AST$ from $S_i$
    Visiting $ASTs$ each node $F$ by depth first manner
    Accumulate $P_i = \{f_{i1}, f_{i2}, f_{i3}, ...f_{in}\}$
  end for
end for
for each $P$ do
  Splitting $P_i$ as Tokens $\longrightarrow \{P_{i1}, P_{i2}, P_{i3}, \ldots, P_{in}\}$
  Adding each Token to $V$
  return $V$
end for
```

Fig. 8 Algorithm 1 for Parsing the files and vector representation of AST

The token id is represented in a real-valued vector through the Word2Vec embedding technique. Dealing with the paths and converting them into a vector is termed as Path2Vec. For example, '*ctx=Load()*' is a token mapped to some integer value and emerged in an array of vector V_p. In this way, the vector sequence is generated as $v_1, v_2, v_3, v_4, \ldots v_n$ and given as input to the network. V_p and V_c are the vectors of previous code and current code, respectively. The hidden state outcome to form a new state as:

$$h^t = LSTM(V_p),\ t = \{1..n\} \tag{7}$$

$$h^t = LSTM(V_c),\ t = \{1..n\} \tag{8}$$

V_p = Vector for previous code
V_c = Vector for current code
h^t = hidden state outcome for both code

4.5 Change Detection

We apply the cosine distance metric for searching the changes that occurred in the revised version of the code. The cosine distance metric formula is the dot product of two attributes. It is the measurement of the cosine angle of two vectors. The outcome of the equation is a normalized value. Resultant values of the metric laid between 0 to 1. The idea comes from determining the angle between the two objects. Researchers extensively use the cosine distance metric to search for the similarities between two components. In this paper, we use the metric to observe the dissimilar part of a program to detect the imposed changes. Cosine similarity is widespread because it is efficiently evaluated on vectors, significantly on sparse vectors. It determines how much two vectors are similar or dissimilar irrespective of their size. The vectors are path sequence embedded vectors in our context. Cosine similarity is much helpful for cases when there are duplicate data matters. Analyzing text similarity is an NLP-based application of the metric. The formula for cosine metric is given as equation (9).

$$\text{dis}(V, V') = \cos(\theta) = \frac{V \cdot V'}{\|V\|\|V'\|} = \frac{\sum_{i=1}^{n} V_i\ V_i'}{\sqrt{\sum_{i=1}^{n} V_i^2}\sqrt{\sum_{i=1}^{n} V_i'^2}} \tag{9}$$

where $v \cdot v' = \sum_1^n v_i \cdot v_i' = v_1v_1' + v_2v_2' + v_3v_3' \ldots + v_nv_n'$ = two vector's dot product.

In equation (9), dis (V, V') is the difference between two vectors. Here V and V' are path vectors of the program's previous version and the current version.

dis (V, V') = 1 if V = V'

dis (V, V') = 0 or < 1 if V ≠ V'

Algorithm 2 in Figure. 9 describes the pseudo convention for vectorization and difference measurement of ASTs. The P_i and P_{i+1} are the two path tokens of some length *m* we get from Algorithm 1. These vectors held the unique ID of the tokens such as:

$P_i = \{t_i^1, t_i^2, t_i^3, \ldots \ldots \ldots t_i^m\}$

$P_{i+1} = \{t_{i+1}^1, t_{i+1}^2, t_{i+1}^3, \ldots \ldots \ldots t_{i+1}^m\}$

t_i^m = mth token of ith path

t_{i+1}^m = mth token of $(i+1)$th path

4.6 Neural Network Model for Difference Measurement

After completing the embedding and tokenization process, we trained our proposed models and other associated, cutting-edge models with the past versions of source codes for *Vehicle* problems. The next move is to review the model's output on the code variant detection task at the end of the training process. How correctly do the differences are identified, and changes are predicted?

Our proposed model is trained with BiLSTM. Supervised learning gives better performance than unsupervised learning.

4.6.1 Attention with BiLSTM

The attention mechanism with BiLSTM enhances the hidden feature of nodes in sequence. Figure 10 depicts the process of attention mechanism. The intention of incorporating the attention mechanism into BiLSTM is to strengthen our model and predict long sequences of source codes. So an attention layer is embedded with the BiLSTM layer. The attention layer aggregates all sequences and from a sequence vector. Merging all the hidden layer output and giving the attention function improves the performance of the model. The BiLSTM is used here to generate a sequence of annotations (h_1, h_2, ….., h_n) for each input sentence. All the vectors h_1, h_2, .., etc., used in work are mainly the interconnection of forward and backward hidden states in the network as given in equation (10).

$$h_j = \left[\overrightarrow{h}_j^T ; \overleftarrow{h}_j^T\right]^T \tag{10}$$

h_j = annotation sequence

```
Algorithm 2:
Input: ASTs' path vectors $P_i, P_{i+1}$, the fixed length of each vector is $m$;
Output: distance vector (diff_list), sent;
Initialize a list V, dictionary tokenID;
for i=1 to n do
   for j = 1 to $len(P_i)$ do
      if $split(P_i)$ is TRUE
         $tokenID = int\_value(p_i^j)$ ;
         $append(tokenID)$;
      end if
   end for
end for
//Creating a list of tokens for both program separately on the basis of term frequency:
$P_{vect1}$ and $P_{vect2}$
for i = 1 to $len(P_{vect1}[1 .. len(P_i)])$ & $len(P_{vect2}[1..len(P_{i+1})])$ do
     for j = 1 to i do
       for k= $len(j)$ to $len(max\_length)$ do   //max_length is for adjusting the max dimension
               $P_{vect1}$ [k] = 0;
          end for
     end for
end for
for i = 1 to $len(P_i)$ & $len(P_{i+1})$ do
     $cosine(P_{vect1}, P_{vect2})$
     if $cosine == 1$
          $continue$
     else
          $append(diff\_list[P_{vect2}])$
     end if
end for
for i = 1 to $len(diff\_list)$ do
     $out = invert\_decode(i)$
     $sent = unparse(out)$
end for
return diff_list, sent
```

Fig. 9 Algorithm 2 for vectorization and difference measurement

We input the annotation h_j to the multilayer perceptron to generate a hidden representation a_t by the equation (11) and depicted in Figure 10.

$$a_t = tanh\,(W_n\, h_j + \; b_n) \tag{11}$$

Here W_n is new weight and b_n is a new bias.

Furthermore, we measured cross-entropy at the softmax layer for every epoch to evaluate the model loss function. The difference between the actual and predicted results is referred to as cross-entropy. The softmax function is used to normalized the weight at attention. Softmax is typically used as the final layer of neural networks. The softmax function's performance range is between 0 and 1.

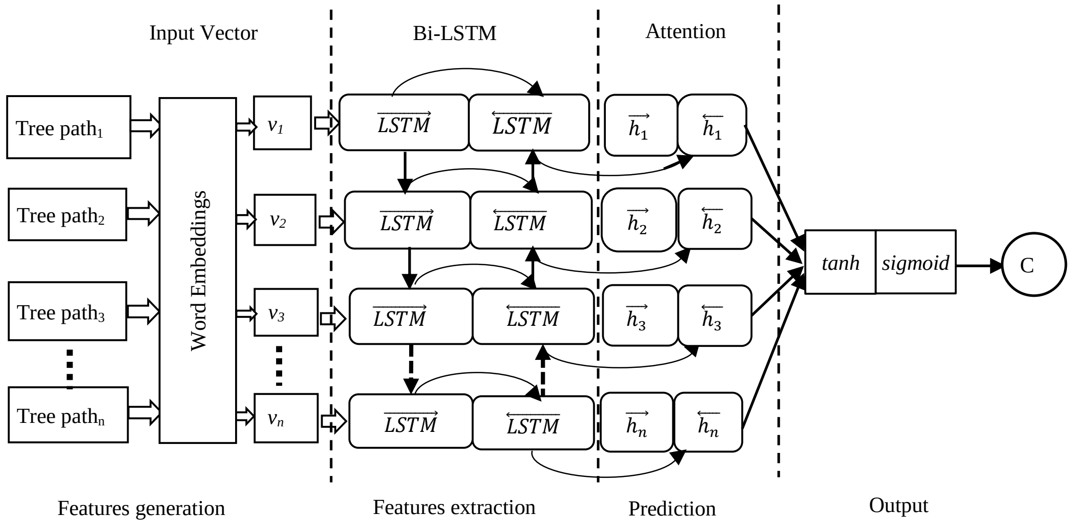


Fig. 10 BiLSTM Model with self-attention

The input to the softmax layer is context vector $x \; = [\, x_1, x_2, x_3, \ldots \ldots .. x_n]$ multiplied with the output a_t and returned normalized weights $p \;=\; [\, p_1, p_2, p_3, \ldots \ldots . p_n]$, that can be defined as in equation(12) :

$$p_i = \frac{exp(a_t x_i)}{\sum_{j=1}^{k} exp(a_t x_j)} \tag{12}$$

Finally, we construct the sequence vector by summing all the nodes with corresponding weights using equation (13). The context vector at the node level is randomly initialized and updated during training.

$$s_i \;=\; \textstyle\sum_j p_i\, h_j \tag{13}$$

s_i = weighted sum

5 Experimental Setup and Results

5.1 Dataset and System Selection

There are barely any datasets available for code change detection, which gives specifications related to our concern, and it's quite challenging to identify change sets. The empirical setup is done on 150k Python Dataset [sri.inf.ethz.ch/py150]. We select an AST file python100k_train for around 2GB in size and JSON format for training and python100k_test for testing. The file contains AST of 100,000 python programs.

These python programs are gathered from the GitHub repository. The programs those AST have a maximum of 30,000 nodes are included in these repositories. According to the system's capacity, we further divide this big-sized JSON file into small chunks with the help of a JSON splitter. Tensorflow with Keras using Python development environment on google cloud. Windows 10, 8GB RAM, Intel Core i7,4690T central processing unit, 2.50GHz, 64-bit OS, x64 based processor.

5.2 AST Encoding

ASTs are capable of storing both the structural as well as semantic information about a program module. Because the vector is composed of path tokens, it cannot be used directly as an input to IABLSTM. As a result, we create a dictionary for mapping tokens to integers. If there are m tokens and each token relates to a distinct integer, the mapping range is 1 to m. To begin, we count the tokens' frequency and then arrange them according to their frequency. Next, we create an indexed dictionary for the ordered tokens, with the most often occurring tokens at the top. Afterward, in the mapping stage, we equalize the length of these numeric vectors. A length of the vector should be chosen with a size less than the required length is denoted by 0 because 0 has no meaning when tokens are mapped starting at 1. If the length of a vector exceeds the required length, the additional component is truncated. Since the tokens with a greater frequency are translated to a smaller integer, the tokens with a lower frequency are mapped to the largest integer. As a result, we find the index of the largest integer in the vector and remove it one by one until the vector length equals the set length. Figure 11 describes the processing of the AST.

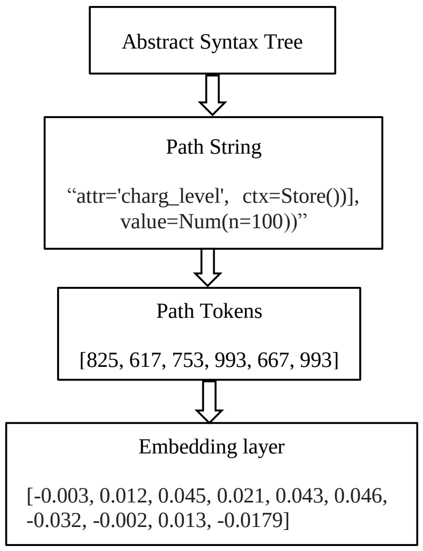


Fig. 11 Processing of AST with embedding layer

Finally, we apply word embedding for high-dimensional vector representation of each token using a trainable word dictionary embedded in the network. Although word embeddings are considered unsupervised, they are trained using a fictitious supervised learning problem.

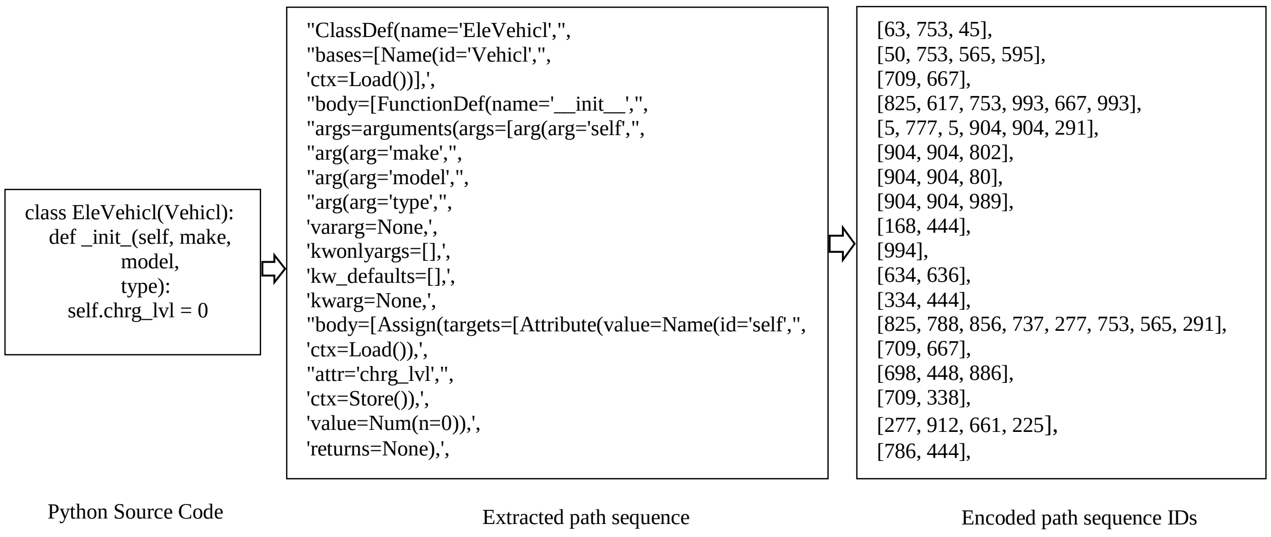
5.3 Path Extraction and Vectorization

The experiment is initially performed on a small dataset with various python codes with multiple entries, e.g., the category of program, type of file, file name, and program's status. The program's status has entries regarding the change or no change and the program itself. We first fetched the program its status according to the program category and file type. In the next step, we preprocessed all these programs and removed the comment lines. Next, we generated an AST and then got a path with pre-order traversal from it. Table 3 briefs the various kinds of tokens extracted from the tree's path and their corresponding integer value. We have enlisted some of them in the table. The table with the tokens column contains various parts of the tree we got by traversing the pre-ordered manner and tokenizing it. Another column of the table includes the token id of these tokens.

Figure 12 portrays an example code for this paper, then we extracted the abstract tree and got the sequence path. Next, we tokenize the path sequence and generate tokens, and then we converted these tokens into identifiers and got token IDs. Every token itself has various tokens.

Table 3 Tokens and its corresponding ID

Tokens	Integer value
ctx=Load()	[709, 667]
body=[FunctionDef(name='__init__')]	[825, 617, 753, 993, 667, 993]
ctx=Store()	[709, 338]
keywords=[]	[958]
annotation=None	[693, 444]
vararg=None	[168, 444]
decorator_list=[]	[206, 370]
kwonlyargs=[]	[994]
returns=None	[786, 444]
kw_defaults=[]	[634, 636]
kwarg=None	[334, 444]
defaults=[]	[636]



```
class EleVehicl(Vehicl):
    def _init_(self, make,
            model,
            type):
    self.chrg_lvl = 0
```

```
"ClassDef(name='EleVehicl',",
"bases=[Name(id='Vehicl',",
'ctx=Load())],',
"body=[FunctionDef(name='__init__',",
"args=arguments(args=[arg(arg='self',",
"arg(arg='make',",
"arg(arg='model',",
"arg(arg='type',",
'vararg=None,',
'kwonlyargs=[],',
'kw_defaults=[],',
'kwarg=None,',
"body=[Assign(targets=[Attribute(value=Name(id='self',",
'ctx=Load()),',
"attr='chrg_lvl',",
'ctx=Store()),',
'value=Num(n=0)),',
'returns=None),',
```

```
[63, 753, 45],
[50, 753, 565, 595],
[709, 667],
[825, 617, 753, 993, 667, 993],
[5, 777, 5, 904, 904, 291],
[904, 904, 802],
[904, 904, 80],
[904, 904, 989],
[168, 444],
[994],
[634, 636],
[334, 444],
[825, 788, 856, 737, 277, 753, 565, 291],
[709, 667],
[698, 448, 886],
[709, 338],
[277, 912, 661, 225],
[786, 444],
```

Fig. 12 Example python code and Tokenization and Encoding Process

5.4 Distance Measurement

We calculated the cosine distance and found the dissimilar part of the code. The cosine metrics give better results on a sequence of the vector. Cosine metrics take the total length of the vectors. These vectors are prepared from BOW, TF, or TF-IDF. Through Figure 13(b), the probable changes are marked in a different color that has been integrated into the code given in 13(a). This python code is for showing the charging level of a vehicle. We have taken the code just for demonstration purposes. In the first code, there is a class defined for an electric vehicle with an initialization function. The revised version has one new *charg* function added to it, and the rest of the code is as same as the previous one. The function sets the charging level to 100. The code from keyword '*def*' to the symbol '*)*' is newly included syntax into the same existing program. The model identified these codes with the highest probability.

5.5 Training and Validation

We evaluated both the model BiLSTM and IABLSTM on the chosen python program of charging the vehicle's battery and Python 150k dataset. The model accuracy and cross-entropy are portrayed with the help of the pyplot library.

```
class EleVehicl(Vehicl):
    def __init__(self, make, model, type):
        self.chrg_lvl = 0
```

(a)

```
class EleVehicl(Vehicl):
    def __init__(self, make, model, type):
        self.chrg_lvl = 0
    def charg(self):
        self.chrg_lvl = 100
        print('charged')
```

(b)

Fig. 13 Previous version (a) current version with change part (b)

The following figures are showing the results of empirical evaluation sequentially. The accuracy and cross-entropy are evaluated on the selected AST dataset for the BiLSTM model, as depicted in Figure 14 and Figure 15, respectively. The training accuracy goes up to 85%, and validation accuracy has grown up to 75%. Likewise, training cross-entropy is down to 35%, whereas validation cross-entropy goes down to 56%.

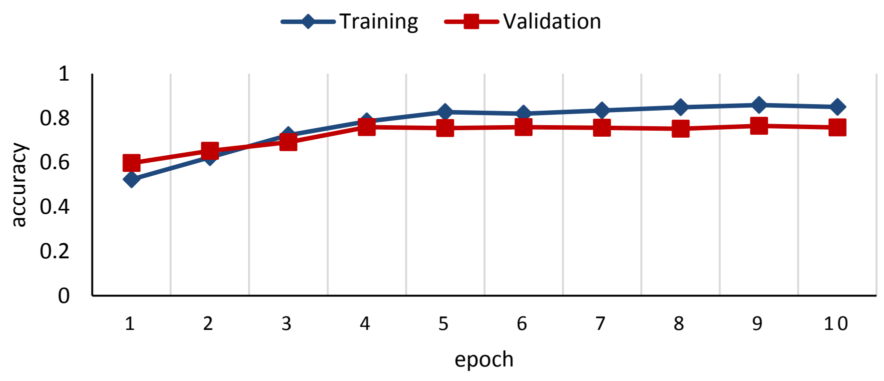
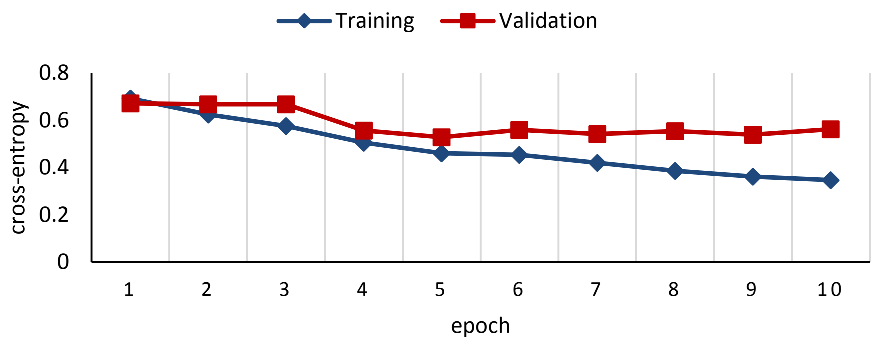


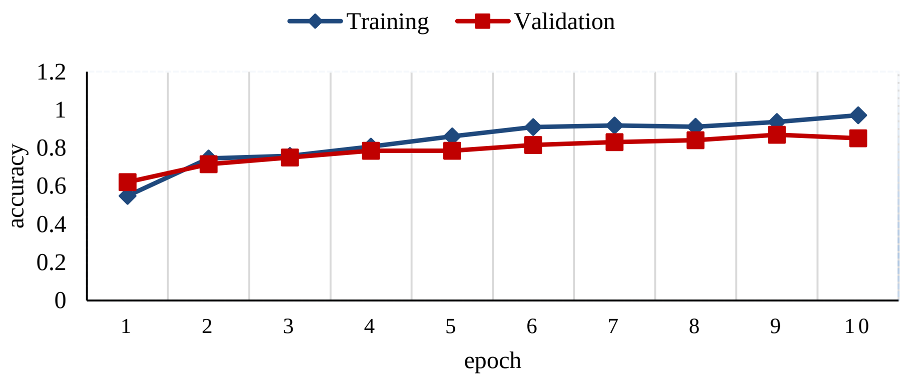
Fig. 14 Accuracy plot for Python150k using BiLSTM



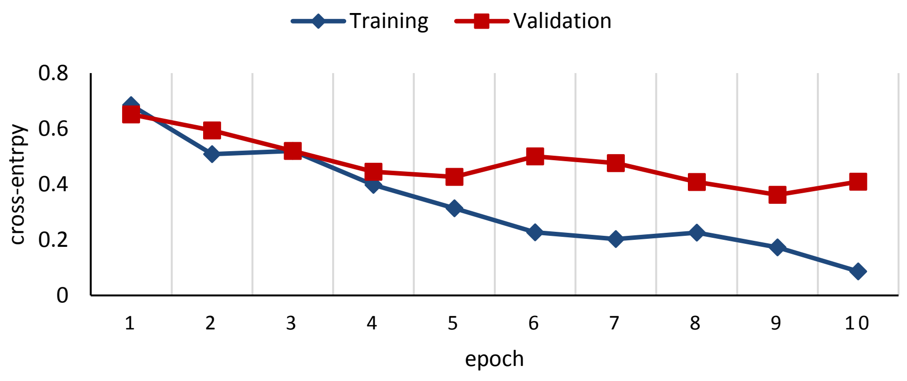
(b)

Fig. 15 Cross-entropy plot for Python150k using BiLSTM

Figures 16(a) and (b) showing the accuracy and cross-entropy evaluated on the same dataset for our model IABLSTM. The training accuracy uplifted to 97%, and validation improved up to 85%. The model training loss comes down to 8%, and validation loss goes down to 40%.



(a)



(b)

Fig. 16 Change detection Model for Python150k using IABLSTM (a) accuracy (b) cross-entropy

The accuracy analysis of both the model is shown in Figure 17.

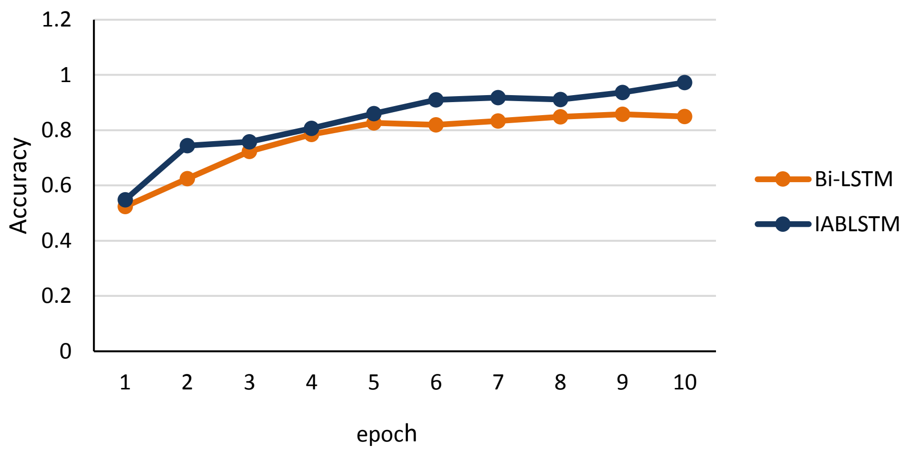


Fig. 17 Epochs wise accuracy analysis of both models on Python 150k

5.6 Hyperparameter

We specified numerous hyperparameter for the experimental purpose to improve our results in this study. To avoid overfitting, we proposed a dropout ratio of 0.25 for our proposed model. Adam optimizer is used in the LSTM network. The learning rate is critical for the training of neural networks since it may be used to modify the model's learning speed. The learning of the network becomes slower or faster as the value of the learning rate decreases or increases. This work determines the learning rate (*l*) 0.002, and during training, the network weights are updated with the value of *l*. We take input length 100, the number of epoch 20 and the number of hidden units 200.

6 Results and Discussion

Table 4. gives a comparative analysis among the various neural network based LSTM, BiLSTM, IABLSTM models. We applied all these models on considered example code, EleVehicl and Python 150k dataset.

Table 4. Accuracy and cross-entropy comparison on different NN models on example source code

	EleVehicl		Python 150k	
Model	Cross-entropy	Accuracy	Cross-entropy	Accuracy
LSTM	0.6772	0.5434	0.6810	0.5682
BiLSTM	0.5459	0.6597	0.3459	0.8497
IABLSTM	0.2693	0.8056	0.0859	0.9721

We have compared cross-entropies of all the models by evaluating them on different data and shown in Figure 18.

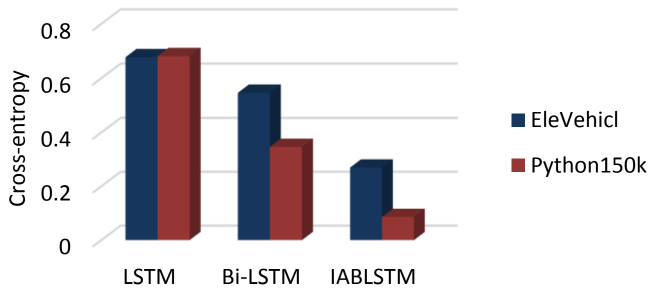


Fig. 18 Comparison on cross-entropies of various models for codes

Figure 19 displays the chart for training accuracy observed on various models, which we mentioned in the paper.

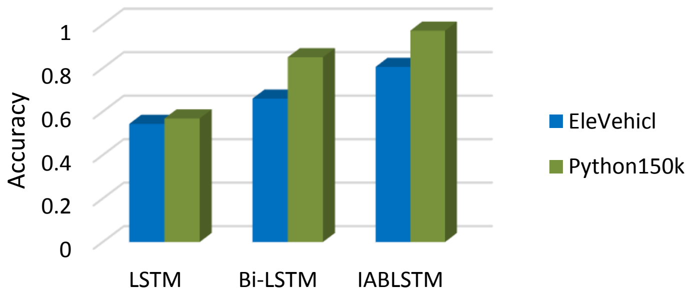


Fig. 19 Comparison on accuracies of various models for example codes

The figure showing the comparative analysis of BiLSTM and IABLSTM that stated using BiLSTM with self-attention improves the accuracy up to 11% to detect the code change compared to simple BiLSTM.

Our IABLSTM model outperforms other state-of-art that uses AST. Table 5 enlisted the results in terms of accuracy. The comparative analysis is performed based on the objective achieved, and the method opted by the authors.

Table 5. Comparison on Accuracy

Objective	Methods	Results(Accuracy/ Precision)
Change impact prediction	LCIP	74%
Code change extraction	CC2Vec	90.7%
Defect prediction	AST+LSTM	90%
Code clone detection	AST+GMMN	96%
Learning program properties	AST path+Word2Vec(JavaScript)	69.1%
Code clone detection and classification	ASTNN	**98.2%**
Code clone detection	AST+ Attention BiLSTM	96.8%
Code change detection	Path2Vec + IABLSTM	**97.2%**

The methods included in given Table 5. have considered AST as source code parsing in most of the works. We also generated AST and processed it through Path2Vec, achieving 28% better performance than the Word2Vec approach.

7 Threats to validity

There are some threats to validity stated here. The Python150k dataset taken for experimental purposes is easily downloadable but not easy to handle completely due to its length. The dataset can be breakdown into small sizes for its usefulness. Another threat we faced is system configuration constraints while evaluating object-oriented based parallel programming code. The exact amount of change is hard to find, although we have achieved accuracy up to the mark.

8 Conclusion and Future Work

In this work, we have presented an approach for source code change detection using deep learning based BiLSTM with self-attention IABLSTM. BiLSTM with attention is used to acquire the semantic as well as syntactic information together of the input code in the encoding process. The results improved up to 85% in terms of accuracy to find the actual impact set compared to other models. The implemented model gives 11% more accuracy than the BiLSTM model. We generated the syntax tree and traversed the tree to extract the said information using a novel approach of parallel programming for change detection. Then the path is transformed into the vector using a vectorization approach and applied distance metrics to analyze all changes. We could see that our model outperforms existing neural network based models. Here we suggested the model that is experimented on small scale software. In the future, the work can be extended to detect the change and its impact on large-scale industry level software with backend and frontend software modules.

Conflicts of Interest

The authors declared no conflict of interest.

References

Ahmad, S., Ghani, A. A. A., Sani, F. M.: Dependence flow graph for analysis of aspect oriented programs. International Journal of Software Engineering & Applications, 5(6), 125 (2014)

Alon, U., Zilberstein, M., Levy, O., Yahav, E.: A general path-based representation for predicting program properties. ACM SIGPLAN Notices. 53(4), 404-419 (2018)

Angerer, F., Grimmer, A., Prähofer, H., Grünbacher, P.: Change impact analysis for maintenance and evolution of variable software systems. Automated Software Engineering. 26(2), 417-461 (2019)

Baah, G. K., Podgurski, A., Harrold, M. J.: The probabilistic program dependence graph and its application to fault diagnosis. IEEE Transactions on Software Engineering. 36(4), 528-545 (2010)

Badri, L., Badri, M., St-Yves, D.: Supporting predictive change impact analysis: a control call graph based technique. In Proceedings of Asia-Pacific Software Engineering Conference, IEEE. 9 (2005)

Bohner, S. A.: Impact analysis in the software change process: a year 2000 perspective. In icsm. 96, 42-51 (1996)

Büch, L., Andrzejak, A.: Learning-based recursive aggregation of abstract syntax trees for code clone detection. In Proceedings of International Conference on Software Analysis, Evolution and Reengineering. 95-104 (2019)

Gethers, M., Kagdi, H., Dit, B., Poshyvanyk, D.: An adaptive approach to impact analysis from change requests to source code. In Proceedings of IEEE/ACM International Conference on Automated Software Engineering. 540-543 (2011)

Hameed, Z., Garcia-Zapirain, B.: Sentiment classification using a single-layered BiLSTM model. IEEE Access. 8, 73992-74001 (2020)

Hoang, T., Kang, H. J., Lo, D., Lawall, J.: CC2Vec: Distributed representations of code changes. In Proceedings of ACM/IEEE International Conference on Software Engineering. 518-529 (2020)

Kitsu, E., Omori, T., Maruyama, K.: Detecting program changes from edit history of source code. In Proceedings of Asia-Pacific Software Engineering Conference, IEEE. 1, 299-306 (2013)

Liang, H., Yu, Y., Jiang, L., Xie, Z. Seml: A semantic LSTM model for software defect prediction. IEEE Access, 7, 83812-83824 (2019)

Meng, Y., Liu, L.: A Deep Learning Approach for a Source Code Detection Model Using Self-Attention. Complexity. (2020)

Molderez, T., Stevens, R., De Roover, C.: Mining change histories for unknown systematic edits. In Proceedings of International Conference on Mining Software Repositories, IEEE. 248-256 (2017)

Rahman, M., Watanobe, Y., Nakamura, K.: A Neural Network Based Intelligent Support Model for Program Code Completion. Scientific Programming. (2020)

Thomas, S. W., Adams, B., Hassan, A. E., Blostein, D.: Validating the use of topic models for software evolution. In Proceeding of IEEE working conference on source code analysis and manipulation. 55-64 (2010)

Tiwang, R., Oladunni, T., Xu, W.: A Deep Learning Model for Source Code Generation. In SoutheastCon, IEEE. 1-7 (2019)

Le TH, Chen H, Babar MA.: Deep learning for source code modeling and generation: Models, applications, and challenges. ACM Computing Surveys (CSUR). 53(3),1-38 (2020)

Zhang J, Wang X, Zhang H, Sun H, Wang K, Liu X.: A novel neural source code representation based on abstract syntax tree. In Proceedings of IEEE/ACM International Conference on Software Engineering (ICSE). 783-794 (2019)

Wang W, Li G, Ma B, Xia X, Jin Z.: Detecting code clones with graph neural network and flow-augmented abstract syntax tree. In Proceedings of International Conference on Software Analysis, Evolution and Reengineering, IEEE. 261-271 (2020)

White M, Tufano M, Vendome C, Poshyvanyk D.: Deep learning code fragments for code clone detection. In Proceedings of the International Conference on Automated Software Engineering (ASE), IEEE. 87-98 (2016)

Eid S, Makady S, Ismail M.: Detecting software performance problems using source code analysis techniques. Egyptian Informatics Journal. 21(4), 219-29 (2020)

Goknil A, Kurtev I, Berg KV.: A rule-based change impact analysis approach in software architecture for requirements changes. arXiv preprint arXiv. 1608.02757 (2016)

Musco V, Carette A, Monperrus M, Preux P.: A learning algorithm for change impact prediction. In Proceedings of the International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering, IEEE. 8-14 (2016)

Siow JK, Gao C, Fan L, Chen S, Liu Y. Core: Automating review recommendation for code changes. In Proceedings of the International Conference on Software Analysis, Evolution and Reengineering, IEEE. 284-295 (2020)

Dam HK, Pham T, Ng SW, Tran T, Grundy J, Ghose A, Kim T, Kim CJ.: A deep tree-based model for software defect prediction. arXiv preprint arXiv. 1802.00921 (2018)

https://www.sri.inf.ethz.ch/py150

Int. j. inf. tecnol.
https://doi.org/10.1007/s41870-022-00966-1

ORIGINAL RESEARCH

Sentiment analysis based on aspect and context fusion using attention encoder with LSTM

Jitendra Soni[1] · Kirti Mathur[2]

Received: 29 December 2021 / Accepted: 8 April 2022

Abstract Sentiment analysis is a type of natural language processing approach that identifies the emotional tone hidden within a body of text by using machine learning techniques. It would be beneficial to consider the aspects and contexts that are hidden behind the text in order to identify the hidden tone. We propose a hybrid model in this research, based on the fusion of Long short term memory (LSTM) and Encoder with attention for sentiment analysis. We've taken into account a variety of factors and different contexts and aspects when analyzing a tweet's word or phrase. Encoder attention mechanism is used to calculate aspect relevance whereas to obtain the related context, we employ Paragraph2vec to facilitate the process of determining contextual meaning. Paragraph2vec and encoder output features were combined and fed into the LSTM for classification. We carried out experiments using the Twitter Sentiment140 dataset. The results of fusion model is being compared with the LSTM, Bi-directional LSTM and Bidirectional Encoder Representations from Transformers (BERT) model and the results of the tests show that our method outperforms the baseline models that are currently available.

Keywords Sentiment analysis · Natural language processing · Long short term memory · Social media

✉ Jitendra Soni
jsoni@ietdavv.edu.in

1 Institute of Engineering and Technology, Devi Ahilya University, Indore, India

2 International Institute of Professional Studies, Devi Ahilya University, Indore, India

1 Introduction

There has been seen a rise in the popularity of sentiment analysis in the recent years, in the processing of social media data from online communities, blogs, micro-blogging platforms, and other collaborative media [1]. There are numerous applications of sentiment analysis, including social media monitoring [2], monitoring of opinion [3], stock market prediction [4], and so on and so forth. Based on the existing research till now sentiment analysis is divided into three broad level document [5], sentence [6] and aspect [7]. Document level analysis require to process the entire document to express some opinion as positive or negative, whereas sentence level uses individual sentences for classification or to draw some opinion.

According to the aforementioned sentiment analysis methods, it is observed that sentiment analysis involving aspect is more precise than other sentiment analysis methods like sentence level or document level. Aspect based require to extract and summarise what people have to say about things, such as entities or aspects of entities [8]. Consider the sentence, "Ravi is brilliant because he is arrogant" this statement shows both positive and negative nature of Ravi. In this case, the words brilliant and arrogant are being considered as aspects that, if ignored, will result in a text prediction error. Likewise the sentence "Vishnu likes watching WWE as it is vibrant and provoking" the sentence has both positive and negative words, likes is a positive word at the same time vibrant and provoking can be considered as negative words. Therefore for classifying a sentence considering both aspect and context will definitely add a flavour. As a result, considering both the aspect and context of a sentence will lead to a more accurate analysis of the sentiments. For sentiment analysis at the aspect level, LSTM and attention are becoming increasingly popular. When it comes to an attention model, it makes sense that a context word closer to the aspect is more important than one further away. This intuition holds true for context word location information as well.

It is common for deep neural network models to be used for sentiment analysis because of their ability to generate low-dimensional word embeddings and better sentence representations [9, 10]. LSTM uses a sequential approach and performs the same operation on each context word, making it difficult to determine the significance of a factor of the various words in a sentence. One solution to this problem is to manually or through some other method explicitly capture the contextual importance of words. Another issue with existing baseline models is that they treat all instances of a target as equally important and simply compute a mean vector over such instances, even if a target contains multiple instances.

Context and aspect-based methods are combined to address the aforementioned concerns, for sentiment analysis wherein the fused features were fed to the Attention Encoder with LSTM model for classification of sentiments.

The following is a summary of the paper's major contributions:

(1) Extraction of features based on context using Paragraph2Vec.
(2) Extraction of features based on aspect using encoder attention.
(3) Fusion of Aspect and Context of Sentences embeddings to add more meaning to the features.
(4) Proposal of Hybrid Attention Encoder with LSTM for final classification based on the fusion.

The remainder of the paper is organised in the following manner: Sect. 2 presents the literature survey. Section 3, defines the methodology, the architecture of Attention Encoder with LSTM, dataset, preprocessing, context retrieval, aspect retrieval, fusion and the LSTM model for classification of sentiments. In Sect. 4 We present the results of our experiments and compare them with the baseline models. Finally Sect. 6 concludes the paper by pointing to possible future work that can be undertaken.

2 Literature survey

In this section, we will be highlighting the work done by various researchers in the field of sentiment analysis. Earlier, sentiment analysis was done with the use of conventional methods, such as sentence modelling and linear classifiers. Sentiment analysis was carried out using a bag of words (BoW) [11], however the bag of words approach lacks the semantics and ordering of words information. N-gram [12] is another method which somehow overcome the drawback of BoW but suffers from data sparsity problem [13]. Later, sentiment classification in sentences was made possible by the introduction of machine learning algorithms like Random forest, Decision Tree, and Support Vector Machine. [14–16], but these algorithms cannot extract features on their own, as a result, deep neural networks, which are capable of making decisions on their own, are increasingly being used. Recently, Recurrent Neural Networks (RNN) have been used in a variety of Natural Language Processing (NLP) applications such as voice recognition, text analysis and language translation. Long short term memory (LSTM), an RNN variant, was very popular, and many researchers used it for various sequence to sequence modelling tasks.

One of the most influential category of sentiment analysis is aspect-level sentiment analysis. The authors Chen et al. [17] uses the combination of LSTM and convolutional neural network (CNN) for identifying sentiments and their work is predicated on the assumption that aspect and opinion are in the same clause. However they have given the context but have not touched upon the aspect. Similarly authors Jain et al. [18] have also proposed a hybrid model based on LSTM and CNN for sentiment analysis. Other work for the analysing aspect in sentiments is shown by the authors, Zhang et al. [19] where they developed two aspect-level gated neural networks for sentiment analysis: first one is use for modelling semantic information from tweet text and the other for modelling the interaction between the target and its environment. On a similar note Tang et al. [9] have modeled sentence representation using LSTM with target dependency. With the introduction of attention model by Vaswani et al. [20] the research has shifted towards identifying the attention of words for NLP tasks. An LSTM model based on attention was proposed by the authors Wang et al. [21] which focuses on various parts of sentences, depending on the context of the sentence to which it is being applied. As per the work done by Ma et al. [22], a method using common sense knowledge was demonstrated for resolving the targeted aspect level sentiment analysis.

The method of learning text representation by using explicit memory has started emerging in recent years [23]. Using eye-tracking data to learn attention information, as recently proposed by the authors Chen et al. [24], the accuracy of attention-based sentiment analysis can be improved. It was proposed by Zhu and Qian [25] to use an auxiliary memory network with a deep learning algorithm in conjunction with an aspect and term learning algorithm to simultaneously learn the characteristics of both aspects and terms.

Attention mechanism today have gained popularity in detecting sentiments that focuses on aspect. The authors Naseem et al. [26] that encodes the representation from the transformer and uses deep intelligent contextual embedding for sentiment analysis. A more recent work by authors dowalgar et al. [27] uses transformer with meta embedding for mixed sentiment analysis on social media. In 2021 the authors He et al. [28] propose a sentiment analysis model that incorporates time squeeze fusion and unimodal reinforced Transformer. The authors Bacco et al. [29] have introduced explanability in the sentiment analysis using transformers.

3 Methodology Attention encoder with LSTM

In this section, we describe the dataset used, the preprocessing performed on the dataset, context retrieval using Paragraph2vec, and aspect retrieval using the Attcoder model. Finally, the fusion of aspect and context is described, which will be fed into the LSTM for sentiment classification. The overall architecture of Paragraph2vec Attention Encoder with LSTM is shown in Fig. 1

3.1 Dataset

The Sentiment140 dataset, which contains 1.6 million Tweets, was used for the experiment. The dataset was obtained from UCI's Machine Learning repository [30].

The dataset includes the following six fields:

(1) target: The label of tweet or its polarity i.e negative,neutral and positive.
(2) ids: The identifier representing the tweet.
(3) date: The tweet's date
(4) flag: The query (lyx). This value is 0 if there is no query.
(5) user: User name who have posted the tweet.
(6) text: The content of the tweet.

3.2 Preprocessing

All 6 fields of the dataset is not required for classification of sentiments, also the text attribute of the dataset has to be preprocessed, as tweets have their own different symbolic language for communication which includes Html links, hash tags, mentions and special characters. So for the sentiment prediction these styled representation has to be removed. Preprocessing has been done using regular expression (re) package of python. The following preprocessing tasks were performed.

3.2.1 Removal of Hash Tags, mentions and special characters

By removing Hash Tags and mentions from a tweet, we aim to have a 'cleaner' tweet. Hash tags are the text that starts with # symbol like $\#([a-zA-Z0-9]1,50)$ so anything after # is meaningless for the sentiment prediction. Similarly mentions are the text that starts with @ symbol like @§+|@§+ again anything after @ symbol will not be contributing to the prediction of the sentiments therefore it has be removed in order to get cleaner text. Special characters includes the characters other than alphanumeric which has been excluded.

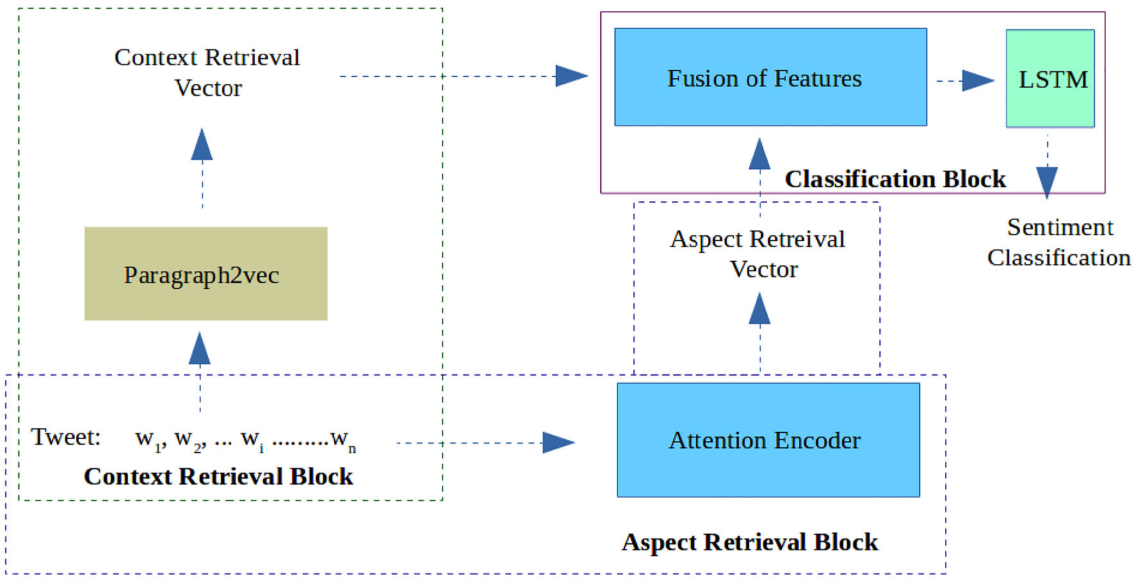


Fig. 1 Paragraph2vec Attention Encoder with LSTM Architecture

3.2.2 Removal of Html links and lower case the text

The second preprocessing tasks that has been implemented is the removal of Html links which starts with *https*. Some people tweets includes the links to some external web sources that has to be removed as such it again does not contribute to the prediction of sentiments. Once all pre-processing has been done finally the clean twitter text is converted to lowercase as it reduces the ambiguity of a sentence.

3.2.3 Removal of stop words

The clean text however contains some words like a, an, the which constitutes the stop words. Stopwords constitutes the words that add little meaning to a sentence and can be omitted without changing the meaning of the sentence. Python's NLTK package was used to remove stop words.

3.3 Context retrieval using Paragraph2vec

Once the preprocessing is accomplished the next task is to retrieve the context from it. The paragraph2vec is used to extract information context from the text. For varying-length texts, covering paragraphs, sentences and documents, the Paragraph Vector algorithm learns feature representations with fixed lengths from these representations. [31]. Paragraph2vec is basically implemented using two models a paragraph vector distributed bag of words (PV-DBW) and a paragraph vector distributed memory model (PV-DM). This study uses the distributed memory model to identify tweet context. Figure 2 shows the description of distributed memory model in which features are extracted for each word and then averaging is done for a specific tweet. In this study, each tweet representing a sentence or paragraph and every word in the sentence is assigned a unique vector. The paragraph and word vectors are averaged, and the Context retrieval vector is obtained by concatenating the two vectors together. The results of the embedding were used as features, which were then fused with the features obtained from the Encoder Attention model to produce a final result.

3.4 Aspect retrieval using encoder attention

The conceptual aspects or the primitives for the sentence is retrieved using the Encoder attention shown in Fig. 3. Here

Q: Vector(Linear layer output) related with what we encode(output of encoder layer).

K: Vector(Linear layer output) related with what we use as input to output.

V: Learned vector(Linear layer output) as a result of calculations, related with input. Q, K, V are the representation of the input sentence, after the embedding and positional encoding steps. The scaled Dot product attention is calculated using the formula below as implemented in Vaswani paper [20].

$$Attention(Q, K, V) = softmax(\frac{Q \times K^T}{\sqrt{d_k}}) \times V \quad (1)$$

The aspect and context words relationship is considered for representing the word vectors while applying encoder attention mechanism. Once the attention for individual words is calculated, it is concatenated for each sentence to represent the Aspect retrieval vector.

3.5 Fusion of aspect and context

The output of PV-DM i.e. Context Retrieval vector and the output of Encoder Attention i.e Aspect Retrieval were fused to obtain the final features embedding for each sentence representing tweets. The fusion is carried out by considering the average of the two features obtained.

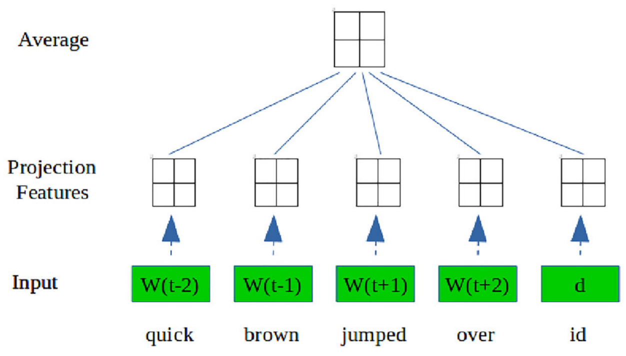


Fig. 2 Paragraph vector distributed memory (PV-DM)

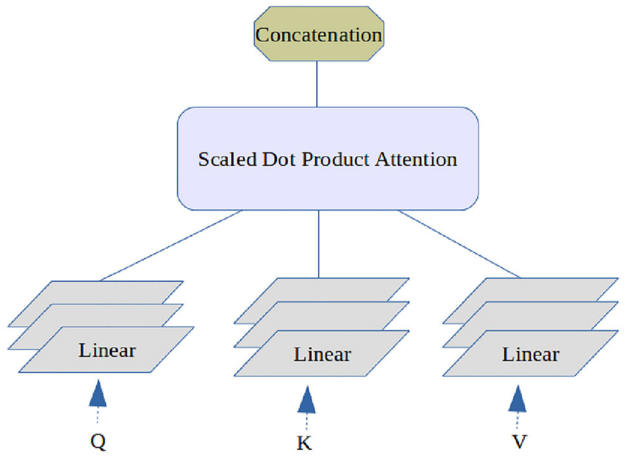


Fig. 3 Encoder attention

3.6 Long short term memory

For learning long term dependencies, LSTM is a good choice which is the category of Recurrent Neural Network (RNN). The meaning and overall polarity of a document are greatly influenced by the long-term dependencies that exist between them. LSTM are designed to address the problem of this problem of long-term dependency by incorporating a memory into the network. Hochreiter and Schmidhuber [32] were the ones who have introduced LSTM. Similar to how a regular RNN works, for each time step the LSTM architecture has a set of repeating modules. During each time step, it is controlled by a set of gates consisting of the input gate, the forget gate, and the output gate. Gates accept the old hidden state and the current time step as inputs. The current memory cell and the hidden state are both updated by these gates. Here are the LSTM transition functions:

$$i_t = (b_i + W_i[h_{t-1}, x_t]) \tag{2}$$

$$C_t = tanh(b_c + W_c[h_{t-1}, x_t]) \tag{3}$$

$$f_t = (W_f[h_{t-1}, x_t] + b_f) \tag{4}$$

$$O_t = (b_o + W_o[h_{t-1}, x_t]) \tag{5}$$

$$C_t = f_t \times C_{t-1} + i_t \times C_t \tag{6}$$

3.7 Architectural design of LSTM

The LSTM model's description is shown in Fig. 4 The model comprises of 2 LSTMs layers. The concatenation of the context feature obtained from Paragraph2vec and the aspect feature obtained from Encoder attention is depicted in the illustration. In Paragraph2vec, we define parameters specifying the minimum number of characters, the size of the embedding, and the size of the window. After done with the training process using Paragraph2vec, the sentence representations such as word vectors and information contexts are obtained. In encoder attention we obtain the attention of each word by applying the Eq. 1 which are then concatenated for each sentence or tweet to obtain the final representation. Then the averaging is done on top of concatenating the aspect and context vector. The number of hidden LSTM layers and dense layers are then initialised based on the classes present in the dataset, softmax activation function is applied to the representations. To represent positive, negative, and neutral sentiment, we're using softmax to transform the last layer's output values into probability values. A predicted class is the end result of the process (e.g sentiment).

4 Preliminary environment for experiment

Our experiments were carried out using the Twitter Sentiment140 dataset. We divided the dataset into two categories: balanced and imbalanced, so that the balanced dataset contains an equal number of records for each of the three sentiments (positive, negative, and neutral). Imbalanced includes records with unequal distribution of all three sentiments. While experimenting, the following settings were used:.

(1) Overfitting: Balanced and imbalanced category datasets were split into 90 : 10 to avoid overfitting. Because of the disparity in the number of positive, negative, and neutral tweets in the original data, the model was also trained with a more balanced data set, which included tweets that were equally positive, negative, and neutral.
(2) Embedding Dimension: The embedding dimension for each tweet selected is 100. As the maximum length of a tweet is close to 100, we believe it would be optimal.
(3) Hidden Layer LSTM : 32 Preliminary tests show that the best results are achieved when 32 layers are used for the hidden layers LSTM.
(4) Batch size : 64 Default value for batch size chosen is 64, meaning that after every processing of 64 sentences gradient is updated.
(5) Number of epochs : $10 - 100$ epochs A preliminary study is conducted to find the minimum number of epochs that would be chosen for training so we have started with 10 epochs and three maxlens, and gradually increasing to 100 with the intent of achieving more convergence.

5 Results and comparison

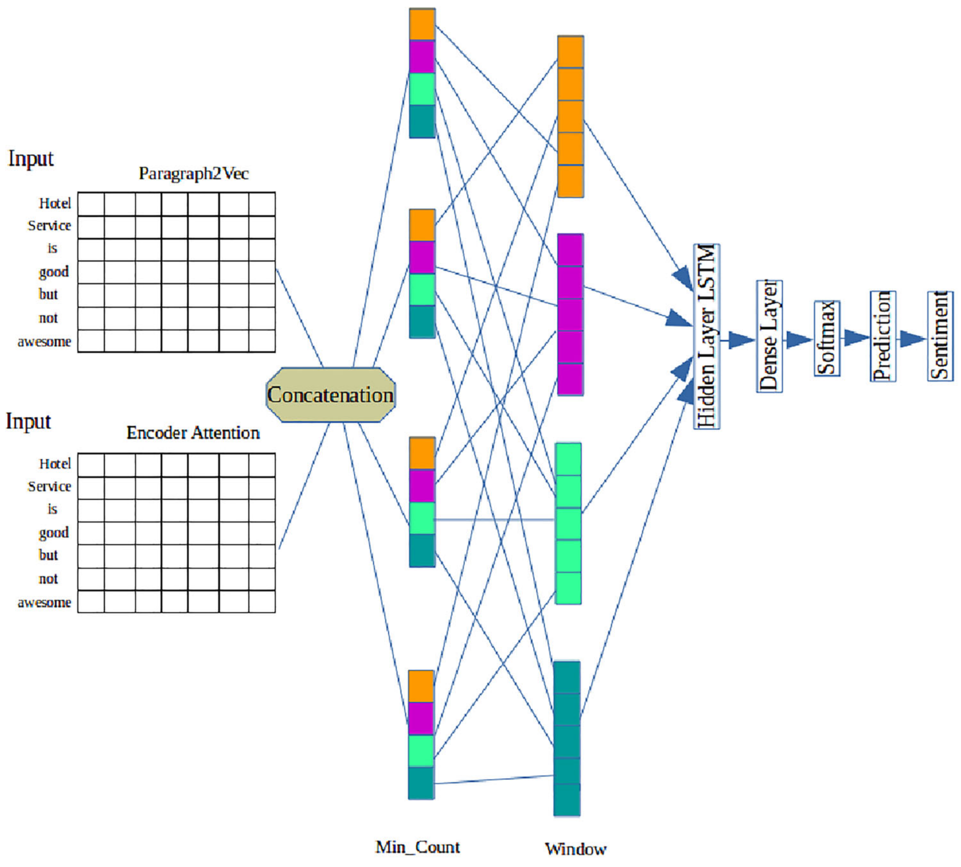
As a means of evaluating the classification results, the Confusion Matrix is used. A confusion matrix sums up classification prediction results. The study's primary objectives are to evaluate the proposed method and its ability to classify sentiments. To put it another way, as per the formula:

$$Acc := \frac{Tpos}{Tpos + Tneg + Fpos + Fneg} \tag{7}$$

$$Pr := \frac{Tpos}{Tpos + Fpos} \tag{8}$$

$$Rec := \frac{Tpos}{Tpos + Fneg} \tag{9}$$

Fig. 4 LSTM architecture



$$F1score := 2 \times \frac{Pr \times Rec}{Pr + Rec} \tag{10}$$

here Tpos represents True positive, Tneg represents True negative, Fpos represents False positive and Fneg represents False negative. Acc represents the accuracy, pr represents the precision and Rec represents the recall value.

5.1 Experiments on balanced dataset

Classification models are represented by a ROC curve (receiver operating characteristic curve) that shows how well they perform across all classification thresholds. The ROC curve for positive sentiments versus all other (negative and neutral) sentiments for the balanced dataset is depicted in Fig. 5. The value of the area under the curve was calculated to be 0.84.

5.2 Experiments on imbalanced dataset

The ROC curve for positive sentiments versus all other (negative and neutral) sentiments for the imbalanced dataset is depicted in Fig. 6. The value of the area under the curve was calculated to be 0.80. The curve's x axis represents the false positive rate, while the y axis represents the true positive rate.

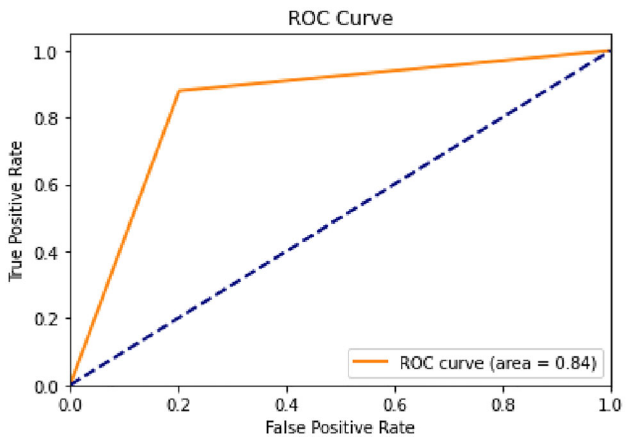


Fig. 5 ROC Curve of positive vs all (negative and neutral) sentiments for Balanced Category

5.3 Result comparisons with other baseline models

This section discusses the comparison results of the proposed Attcoder with other baseline models such as LSTM, Bi-directional LSTM and BERT for both balanced and imbalanced category of dataset. Table 1 shows the the comparison among all mentioned models for balanced category. In terms of classification metrics, Att-coder has outperformed all baseline models with precision of 0.86

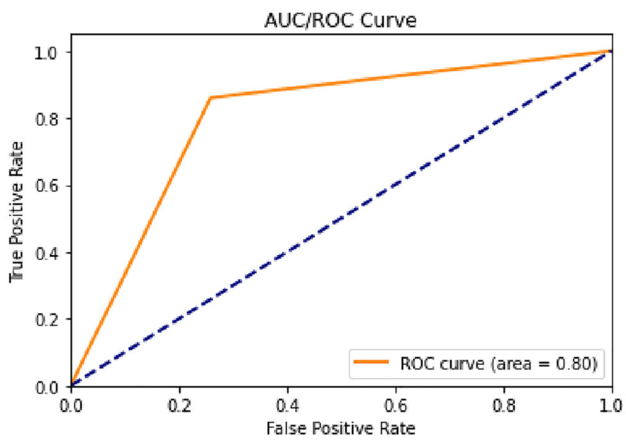


Fig. 6 ROC Curve of positive vs all (negative and neutral) sentiments for Imbalanced Category

Table 1 Comparison of classification metrics for Balanced category

Models	Precision	Recall	f1-score
LSTM (baseline)	0.75	0.75	0.75
Bi-directional LSTM	0.77	0.75	0.76
BERT	0.76	0.75	0.75
Att-Coder (Proposed)	**0.86**	**0.85**	**0.86**

Bold represents our model obtained accuracy which is better than the other listed models

Table 2 Comparison of classification metrics for Imbalanced category

Models	Precision	Recall	f1-score
LSTM (baseline)	0.65	0.62	0.63
Bi-directional LSTM	0.67	0.61	0.63
BERT	0.69	0.67	0.68
Att-Coder (Proposed)	**0.80**	**0.79**	**0.79**

Bold represents our model obtained accuracy which is better than the other listed models

percent, recall of 0.85 percent and f1-score of 0.86 percent. However the results of Bi-directional LSTM were promising among the other compared models.

Table 2 shows the the comparison among all mentioned models for imbalanced category. In terms of classification metrics, Att-coder has once again outperformed all baseline models with precision of 0.80 percent recall of 0.79 percent and f1-score of 0.79 percent respectively.

6 Conclusion

As part of this study, sentiment analysis specifically classification is accomplished. For extracting context feature we use Paragraph2vec and encoder attention for aspect feature extraction. LSTM is used for sentiment classification based on the fusion of an aspect vector and a context vector. Sentiment140 dataset is chosen for classification. The accuracy on the balanced category is reported to be 88.33 percent and for the imbalanced category comes out to be 79 percent. Comparing our proposed approach to the baseline models reveals that our proposed approach outperforms them all however there is still a lot of room for improvement, especially when it comes to the relationship between context and aspect. Context aids in the discovery of context-specific meaning, whereas the aspect reveals polarity. For sentiment analysis, considering both factors will add more meaning. One can plan for the future to handle vectors of aspects that include multiple words. Even more difficult is to deal with the task of recognising an explicit feature's implicit feature in an effective analysis method.

References

1. Cambria E, Das D, Bandyopadhyay S, Feraco A, et al. (2017) A practical guide to sentiment analysis. Springer
2. Zárate J. M, Santiago S. M (2019)"Sentiment analysis through machine learning for the support on decision-making in job interviews. In: International Conference on Human–Computer Interaction. Springer, pp. 202–213
3. Xiong S, Wang K, Ji D, Wang B (2018) A short text sentiment-topic model for product reviews. Neurocomputing 297:94–102
4. Groß-Klußmann A, König S, Ebner M (2019) Buzzwords build momentum: global financial twitter sentiment and the aggregate stock market. Expert Syst Appl 136:171–186
5. Lin C, He Y (2009) Joint sentiment/topic model for sentiment analysis. Proceedings of the 18th ACM conference on Information and knowledge management, pp. 375–384
6. Shoukry A, Rafea A (2012) Sentence-level arabic sentiment analysis. In: *2012 International Conference on Collaboration Technologies and Systems (CTS)*. IEEE, pp. 546–550
7. Schouten K, Frasincar F (2015) Survey on aspect-level sentiment analysis. IEEE Trans Knowl Data Eng 28(3):813–830
8. Zhang L, Wang S, Liu B (2018) Deep learning for sentiment analysis: a survey. Wiley Interdiscip Rev Data Min Knowl Discov 8(4):e1253
9. Tang D, Qin B, Feng X, Liu T, (2015) Effective lstms for target-dependent sentiment classification. arXiv preprint arXiv:1512.01100
10. Wang Y, Huang M, Zhu X, Zhao L (2016) Attention-based lstm for aspect-level sentiment classification. In: Proceedings of the 2016 conference on empirical methods in natural language processing, pp. 606–615
11. Da Silva NF, Hruschka ER, Hruschka ER Jr (2014) Tweet sentiment analysis with classifier ensembles. Decision Sup Syst 66:170–179
12. Joachims T (1998) Text categorization with support vector machines: learning with many relevant features. In: European conference on machine learning. Springer, pp. 137–142
13. Bengio Y, Ducharme R, Vincent P, Jauvin C (2003) A neural probabilistic language model. J Mach Learn Res 3: 1137–1155
14. Neethu M, Rajasree R (2013) Sentiment analysis in twitter using machine learning techniques. In: 2013 Fourth International

Conference on Computing, Communications and Networking Technologies (ICCCNT). IEEE, pp. 1–5
15. Jadav BM, Vaghela VB (2016) Sentiment analysis using support vector machine based on feature selection and semantic analysis. Int J Comput Appl 146(13)
16. Ajit P (2016) Prediction of employee turnover in organizations using machine learning algorithms. Algorithms 4(5):C5
17. Chen P, Xu B, Yang M, and Li S (2016) Clause sentiment identification based on convolutional neural network with context embedding. In: 2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD). IEEE, pp. 1532–1538
18. Jain PK, Saravanan V, Pamula R (2021) A hybrid cnn-lstm: a deep learning approach for consumer sentiment analysis using qualitative user-generated contents. Trans Asian Low Resou Lang Inform Process 20(5):1–15
19. Zhang M, Zhang Y, Vo D.-T (2016) Gated neural networks for targeted sentiment analysis. In: Thirtieth AAAI conference on artificial intelligence
20. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A. N, Ł. Kaiser, Polosukhin I (2017) Attention is all you need. In: Advances in neural information processing systems, pp. 5998–6008
21. Wang J, Li J, Li S, Kang Y, Zhang M, Si L, Zhou G (2018) Aspect sentiment classification with both word-level and clause-level attention networks. IJCAI 2018:4439–4445
22. Ma Y, Peng H, Cambria E (2018) Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive lstm. In: Thirty-second AAAI conference on artificial intelligence
23. Sukhbaatar S, Szlam A, Weston J, Fergus R (2015) End-to-end memory networks. arXiv preprint arXiv:1503.08895
24. Chen P, Sun Z, Bing L, Yang W, (2017) Recurrent attention network on memory for aspect sentiment analysis. In: Proceedings of the 2017 conference on empirical methods in natural language processing, pp. 452–461
25. Zhu P, Qian T (2018) Enhanced aspect level sentiment classification with auxiliary memory. In: Proceedings of the 27th International Conference on Computational Linguistics, pp. 1077–1087
26. Naseem U, Razzak I, Musial K, Imran M (2020) Transformer based deep intelligent contextual embedding for twitter sentiment analysis. Fut Gen Comput Syst 113:58–69
27. Dowlagar S, Mamidi R (2021) Cmsaone@ dravidian-codemix-fire2020: a meta embedding and transformer model for code-mixed sentiment analysis on social media text. arXiv preprint arXiv:2101.09004
28. He J, Mai S, Hu H (2021) A unimodal reinforced transformer with time squeeze fusion for multimodal sentiment analysis. IEEE Signal Process. Lett 28:992–996
29. Bacco L, Cimino A, Dell'Orletta F, Merone M (2021) Extractive summarization for explainable sentiment analysis using transformers
30. Dua D, Graff C (2017) UCI machine learning repository. [Online]. Available at: http://archive.ics.uci.edu/ml
31. Le Q, Mikolov T (2014) Distributed representations of sentences and documents. In: International conference on machine learning. PMLR, 2014, pp. 1188–1196
32. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Computat 9(8):1735–1780

Synthesis and study of ScN thin films

Susmita Chowdhury,[1] Rachana Gupta,[1] Parasmani Rajput,[2] Akhil Tayal,[3] Dheemahi Rao,[4, 5, 6] Reddy Sekhar,[7] Shashi Prakash,[1] Ramaseshan Rajagopalan,[7] S. N. Jha,[2] Bivas Saha,[4, 5, 6] and Mukul Gupta[8, *]

[1] *Applied Science Department, Institute of Engineering and Technology, DAVV, Indore, 452017, India*
[2] *Beamline Development and Application Section, Physics Group, Bhabha Atomic Research Centre, Mumbai 400085, India*
[3] *Deutsches Elektronen-Synchrotron DESY, Notkestrasse 85, D-22607 Hamburg, Germany*
[4] *Chemistry and Physics of Materials Unit, Jawaharlal Nehru Centre for Advanced Scientific Research, Bengaluru 560064,India*
[5] *International Centre for Materials Science, Jawaharlal Nehru Centre for Advanced Scientific Research,Bengaluru 560064, India*
[6] *School of Advanced Materials (SAMat), Jawaharlal Nehru Centre for Advanced Scientific Research, Bengaluru 560064,India*
[7] *Surface and Nanoscience Division, Materials Science Group, Indira Gandhi Centre for Atomic Research, HBNI, Kalpakkam - 603102, India*
[8] *UGC-DAE Consortium for Scientific Research, University Campus, Khandwa Road, Indore-452 001,India*

(Dated: April 18, 2022)

To contemplate an alternative approach for the minimization of diffusion at high temperature depositions, present findings impart viability of room-temperature deposited reactively sputtered ScN thin film samples. The adopted room temperature route endows precise control over the R_{N_2} flow for a methodical structural phase evolution from Sc→ScN and probe the correlated physical aspects of the highly textured ScN samples. In the nitrided regime i.e. at R_{N_2} = 2.5 - 100% flow, incorporation of unintentional oxygen defects were evidenced from surface sensitive soft x-ray absorption spectroscopy study, though less compared to their metal (R_{N_2} = 0%) and interstitial (R_{N_2} = 1.6%) counterparts, due to higher Gibb's free energy for Sc-O-N formation with no trace of ligand field splitting around the O K-edge spectra. To eradicate the sceptism of appearance of N K-edge (401.6 eV) and Sc L-edge (402.2 eV) absorption spectra adjacent to each other, the nascent Sc K-edge study has been adopted for the first time to validate complementary insight on the metrical parameters of the Sc-N system taken into consideration. Optical bandgaps of the polycrystalline ScN thin film samples were found to vary between 2.25 - 2.62 eV as obtained from the UV-Vis spectroscopy, whereas, the nano-indentation hardness and modulus of the as-deposited samples lie between 15 - 34 GPa and 152 - 476 GPa, respectively following a linearly increasing trend of resistance to plastic deformations. Besides, contrary to other early 3d transition metal nitrides (TiN, VN, CrN), a comprehensive comparison of noticeably large homogeneity range in Sc-N has been outlined to apprehend the minuscule lattice expansion over the large R_{N_2} realm.

I. INTRODUCTION

Early 3d transition metal nitrides (TMNs) e.g. ScN, TiN, VN and CrN even though crystallizes in cubic rocksalt-type B1 structure, but distinct band structure manifests heterogeneous electrical conductivity within the family of early TMNs [1, 2]. Among them, ScN as a semiconductor sought a profound research attention in recent times, whereas, the rest of the early 3d TMNs exhibit metallic nature and are substantially well explored since decades [2, 3]. Apart from ScN been a pre-eminent refractory compound (melting point exceeding ≈2873 K, corossion resistant, high hardness of ≈21 GPa) [3] exhibiting high thermoelectric *figure-of-merit* (0.3 at 800 K) [4] and assist as a template for the growth of low dislocation density GaN [5], ScN further possess immense functionalities in conjuction with other TMNs viz. $Sc_xGa_{1-x}N$ as light emitting diodes [6], $Al_{1-x}Sc_xN$ as MEMS magnetoelectric sensors [7, 8], epitaxial (Zr,W)N/ScN - metal/semiconductor superlattices as thermionic energy conversion devices [9] etc. Besides these intriguing aspects, the lowest enthalpy of formation (ΔH_f^0 = -19.79 eV) [10] Sc-O of Sc compared to other TMNs is the principal intricacy for synthesis of pure ScN, resultant being an unintentional n-type degenerate semiconductor [11, 12].

Hence for application based perspectives, to modulate the band structure engineering for superior device performances, so far, most of the studies adopted high vacuum ($\leq 10^{-8}$ Torr) depositions to ensure low defect concentrations with atomically smooth epitaxial growth of ScN thin film samples on variety of single crystal substrates (MgO, Al_2O_3, GaN, SiC etc.) [13–16], and few of them aided with process parameters are tabulated in Table I. As can be seen from Table I, conventional use of high substrate temperature ($T_s \geq 823$ K) has been an integral part during the synthesis of ScN thin film samples, possibly due to higher adatom mobility promoting enhanced crystalline defect free ScN growth [17]. In addition, intensive research attention have also been dedicated to get an insight on explicit defect contributions and microstructural growth behavior (e.g. dislocations, twin domains etc.) of ScN samples and when fabricated with other metals and/or TMNs as in metal/semiconductor superlattices, multilayers etc [1, 4, 18, 19].

In terms of defects, even though it is well known that

* Corresponding author:mgupta@csr.res.in

during the growth of ScN itself, finite incorporation of substitutional (O_N) and/or interstitial oxygen (O_i) is inherent regardless of T_s [16, 18, 20], but combined study of first principles density functional theory (DFT) with site occupancy disorder technique reveals that the electronic band structure of ScN remains unaltered despite of a shift of the Fermi energy level to the bottom of the conduction band [4]. Nonetheless, only recently, the primary contribution of oxygen incorporation has also been attributed to the surface oxidation [13, 17]. Howbeit, nitrogen vacancies (V_N) are known to form a defect energy level at ≈1.26 eV above the valence band maxima at Γ point of the Brillouin zone [21]. In this context, it is to be mentioned here that significance of high T_s depositions in supression of defects were found to be conflicting in literature [13, 17, 22, 23], yet have not been highlighted so far.

Furthermore, as regards to technological viability in electronics viz. CMOS integrated circuits, plastic substrates etc, high T_s synthesis is highly undesirable [24]. Moreover, the extent of diffusion across the metal-semiconductor superlattice and/or multilayer interfaces will be comparatively high at a high T_s regime [12], which could limit the device performances in practical applications. Additionally, interdiffusion across film-substrate interfaces are also pronounced at high T_s depositions [21, 25]. In view of this, contrary to high T_s depositions, we adopted a room temperature deposited reactive magnetron sputtering technique for the synthesis of ScN thin film samples, as ScN favors thermodynamical growth conditions even at 298 K (ΔH_f^0 = -3.29 eV). Such temperature regime also paves the way for precise control over variation of relative N_2 partial pressures (R_{N_2}) to probe the structural phase evolution from hexagonal close packed (hcp) Sc to rocksalt-type face centered cubic (fcc) ScN, which is still ambiguous.

In order to probe the electronic structure of ScN, so far, usually x-ray photoelectron spectroscopy (XPS) or soft x-ray absorption spectroscopy (SXAS) at N K-edge (401.6 eV) and Sc L-edge (402.2 eV) were considered [21, 26–28]. But, since the two absorption edges appear very close to each other and moreover, both XPS and SXAS are known to be surface sensitive techniques [29], an alternative powerful technique such as x-ray absorption fine structure (XAFS) can provide better insight on the metrical parameters at atomic scale level. With this motif, for the first time, XAFS was implemented on the Sc-N system complementary to SXAS to probe the K-edge of Sc in ScN. In spite of recent surge in investigation of various physical properties, further realization of variation of N were systematically demonstrated in terms of structural, electronic, optical and mechanical responses of room temperature deposited ScN thin film samples which are still missing in literature.

II. EXPERIMENTAL

Metallic Sc and a series of ScN thin film samples were deposited on amorphous quartz and single crystal Si (100) substrates at various R_{N_2} [= $P_{N_2}/(P_{N_2} + P_{Ar})$, where P_{N_2} and P_{Ar} are nitrogen and argon partial pressures, respectively] flow = 1.6, 2.5, 5, 10, 25, 50 and 100% in closed intervals using a direct current magnetron sputtering (dcMS) at ambient temperature (≈300 K). For thin film deposition, a Sc (99.95% pure) 3-inch target was sputtered in the presence of 5N purity Ar and/or N_2 gas flows. Prior to the deposition, the substrates were cleaned in an ultrasonic bath of acetone followed by methanol wiping with dry air blown and were loaded into the chamber. Subsequently, the sample holder was baked for 1 hour at 573 K and then cool down to room temperature to achieve a base-pressure of about 1×10^{-7} Torr or lower. During deposition, the working pressure was $\approx 3\times10^{-3}$ Torr and the substrate holder rotation was kept fixed at 60 rpm to get better uniformity of the samples.

For thickness calibration of the samples, x-ray reflectivity (XRR) measurements were performed using Cu-Kα x-rays on a Bruker D8 Discover system. Once the deposition rate was obtained from the fitting of the XRR data (not shown), typically 200 nm thick samples were prepared following the similar deposition procedure. The structural characterization of samples were carried out using x-ray diffraction (XRD) using a Bruker D8 Advance XRD system based on θ-2θ Bragg-Brentano geometry with Cu-Kα (1.54 Å) x-rays and detected using a fast 1D detector (Bruker LynxEye). To probe the local electronic structure, surface sensitive SXAS measurements were performed at N K-edge and Sc $L_{(III,II)}$-edges in total electron yield (TEY) mode at BL-01 beamline of Indus-2 synchrotron radiation source [37] at RRCAT, Indore, India. Complementary to SXAS, to get an elementary insight probing the deep core level in atomic scale regime, x-ray absorption fine structure (XAFS) measurements were performed in fluorescence mode at BL-09 beamline at RRCAT, Indore, India and also at P64 beamline of PETRA-III, DESY, Germany [38]. XAFS data taken at Sc K-edge from both beamlines were found to be similar and XANES data taken at BL-09 and EXAFS data taken at P64 has been included. The obtained data was processed in Athena software [39] with pre and post-edge normalization [40] and fitting of the Fourier Transform (FT) spectra were performed using a software code developed by Conradson et al [41]. The fitted R range was taken from 0 to 10 Å, while the used k-range was 3 to 8 Å^{-1}.

The optical absorption spectra of the ScN thin film samples were recorded by Perkin Elmer, Lambda-750 UV–Visible spectrophotometer with double beam monochromator in the wavelength range of 250 - 1000 nm at room temperature. The reflectance of the recorded data were converted to absorption spectra using Kubelka-Munk radiative transfer model, which is associated with

TABLE I. Growth techniques of ScN thin film samples deposited using various substrate temperature (T_s) and deposition power (P) on different substrates with corresponding lattice parameter (LP) and direct optical bandgap (E_g) values. Here, dcMS = Direct current magnetron sputtering and MBE = Molecular beam epitaxy, RMS = Reactive magnetron sputtering, $300^{\dagger}$ = amorphous at 300 K, GGA-PBE = Perdew-Burke-Ernzerhof GGA exchange correlation functional, HSE06 = Heyd-Scuseria-Ernzerhof Hybrid functional, FLAWP = Full-potential linearized augmented plane wave method, FLAWP-GGA = Full-potential linearized augmented plane wave method with generalized gradient approximation, LDA = Local density approximation, GGA+U = Generalized gradient approximation with Hubbard U correction.

Exp. Tech.	**Substrate**	**Process Parameters**	**LP (Å)**	**E_g (eV)**	**Ref.**
dcMS	MgO (001) & Si (001)	T_s = 1103 K, P =150 W	4.50	-	[4]
dcMS	c-plane Al_2O_3, MgO (111) & r-plane Al_2O_3	T_s = 973 - 1223 K, P = 125 W	4.504 - 4.512	-	[16]
dcMS	MgO (001)	T_s = 1123 & 1223 K, P = 60 - 300 W	4.50	2.18 - 2.7	[30]
dcMS	MgO (001)	T_s = 973 K	4.573	2.59	[26]
MBE	GaN (0001), SiC (0001) & AlN (0001)	T_s = 1023 K, P = 200 W	4.497	2.1	[15]
RMS	MgO (001)	T_s = 823 K, P = 25 - 127 W	4.52 - 4.54	2.1	[25]
dcMS	MgO (001)	T_s = 873 - 1073 K, P = 125 W	4.50	2.19 - 2.23	[17]
dcMS	c-plane Al_2O_3 & Si	T_s = $300^{\dagger}$ - 1023 K, P = 40 W	≈4.47 - 4.52,	2.2 - 3.1	[17]
MBE	MgO (001),	T_s = 1073 K,	-	2.15	[31]
dcMS	**Quartz & Si (100)**	**T_s = 300 K, P = 100 W**	**4.49 - 4.567**	**2.25 - 2.62**	**this work**
Theoretical					
GGA-PBE	-	-	4.519	2.02	[30]
& HSE06			4.499	-	
FLAPW &	-	-	4.42	-	[32]
FLAPW-GGA	-	-	4.50	-	
LDA	-	-	4.47	-	[33]
GGA+U	-	-	4.52	1.86	[34]

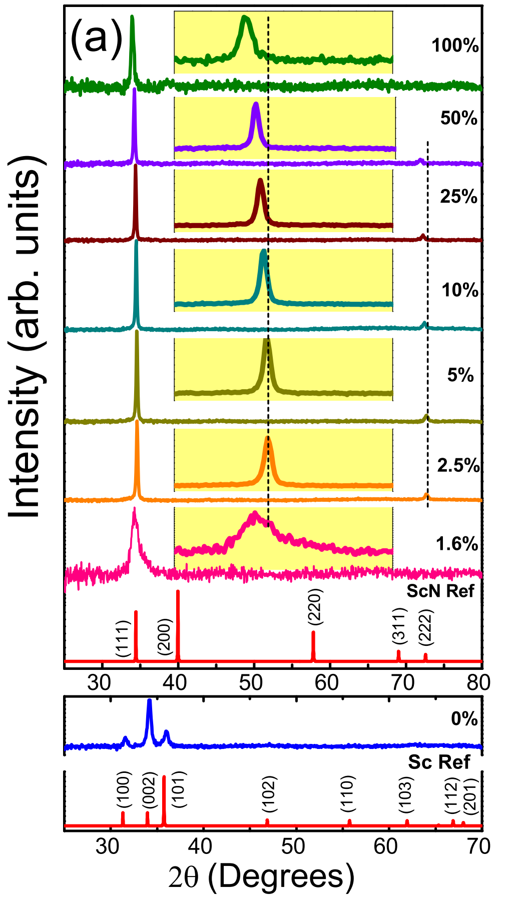
the absorption coefficient (α) [42] of the ScN thin film samples. To measure the hardness and elastic modulus of the samples, nanoindentation tests (Anton Paar, Switzerland) were performed using Berkovich diamond indenter tip with standard loading and unloading procedure based on Oliver and Pharr model [43]. In order to suppress the substrate effects, the measurements were performed on one/tenth of the total sample thickness [44].

III. RESULTS

A. X-Ray Diffraction

To take into account the phase formation of as-deposited samples, Figure 1(a) illustrates the XRD data of Sc and ScN thin film samples and are compared with bulk references [30, 45], whereas, Figure 1(b) demonstrates the obtained variations in the lattice parameters (LP) and crystallite size as a function of R_{N_2}. In addition, the highlighted region (in cyan) of Figure 1(b) depicts the experimentally obtained LP of ScN thin film samples in literature and the red dotted line is a guide to eye for the theoretical predicted value of bulk ScN [30]. As can be seen from Figure 1(a), the occurence of three prominent peaks for Sc thin film sample can be assigned to (100), (002) and (101) reflection planes of hcp Sc, whereas for ScN samples, three different growth stages can be witnessed with variation in R_{N_2} flow namely, (i) interstitial incorporation of N atoms within hcp Sc, (ii) formation of NaCl rocksalt type fcc-ScN, and (iii) gradual expansion of ScN lattice due to incorporation of N atoms in fcc-ScN.

Here, it is to be mentioned that an earlier report on deposition of polycrystalline ScN thin film samples on quartz substrates at T_s = 300 K using rf magnetron sputtering have reported amorphous growth and later on tuning of T_s to high temperature resulted in preferential grain growth either along (111) or (200) plane, albeit the XRD data for the as-deposited samples were not presented therein [46]. However, in the present work, at a very initial stage of R_{N_2} = 1.6%, the N atoms occupy the interstitial sites of hcp Sc manifesting an asymmetry and broadening in the reflection peak which suggests phase



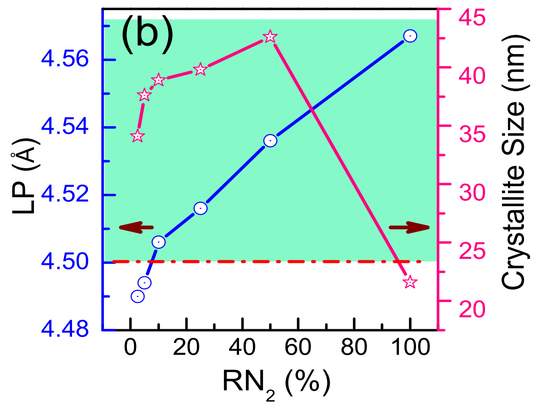


FIG. 1. XRD pattern(a) and obtained variations in the lattice parameters and crytallite size (b) as a function of R_{N_2} of pure Sc and ScN samples deposited at various R_{N_2}= 1.6, 2.5, 5, 10, 25, 50 and 100%.

co-existance of hcp Sc and fcc ScN at certain phase fractions (as could also be evidenced from our XAFS data) and an enhancement in the crystalline disorder. Later on, when R_{N_2} was increased to 2.5%, the sample exhibited a highly textured orientation with (111) and (222) reflection planes, resembling a rocksalt NaCl type structure even though the LP was 4.49 Å, slightly less than the theoretically predicted value of 4.501 Å [30]. With further increase in R_{N_2} flow (from 5 to 50%), the texturing of the samples remain unaltered, but due to gradual incorporation of N atoms within the crystal lattice, it starts to expand as shown in Figure 1(a) by the gradual peak shifts of both (111) (magnified view shown in the highlighted inset) and (222) reflection peaks towards the lower diffraction angle (shown by dotted lines) accompanied with an increase in the crystallite size (Figure 1(b)).

Such unidirectional grain growth can be attributed to the kinetics driven mechanism at ambient temperature deposition (≈300 K), due to trapping of the less mobile adatoms in the highest surface energy site i.e along the (111) reflection plane of ScN [47]. Besides, further increase in R_{N_2} at 100% flow causes lattice expansion in expense of reduced peak intensity of (111) reflection plane due to possible oversaturation of N, resulting in broadening of the (111) peak (as can be seen in highlighted region of Figure 1(a)) with reduced crystallite size (Figure 1(b)), and further absence of (222) grain growth suggests shattering of the long range periodicity. In view of growth evolution of Sc→ScN at various R_{N_2} flow with electronic properties, SXAS measurements were performed and are discussed in section III B.

B. Soft X-Ray Absorption Spectroscopy

To get an insight on the local electronic structure of Sc and ScN thin film samples with variation in R_{N_2} flow, SXAS spectra of Sc L-edge and N K-edge were recorded and are shown in Figure 2(a), whereas, the first order derivative of the absorption spectra with respect to the photon energy is described in Figure 2(b). Additionally, to probe the oxidation effect, Figure 2(c) demonstrates the O K-edge XANES spectra of the samples. Here, it is to be mentioned that for a pure Sc sample, two absorption edges namely L_{III} and L_{II} are expected due to spin-orbit splitting of the Sc 2p orbital into 2p3/2 and 2p1/2 states in the absence of any ligand (C, N, O etc.) around the vicinity of Sc, a consequence of the transition of core electron from Sc 2p3/2→Sc 3d (L_{III}) and Sc 2p1/2→Sc 3d (L_{II}) states [48]. However, in the present case, the prominent features of Sc sample in Figure 2(a) and 2(b) marked as '1', '2', '3' and '4' can be assigned to L_{III} (t_{2g}), L_{III} (e_g), L_{II} (t_{2g}) and L_{II} (e_g) edges respectively due to the presence of a finite amount of unintentional O ligand field, which led to hybridization of Sc 3d-O 2p orbitals resulting in further splitting of each L_{III} and L_{II} features into t_{2g} and e_g.

Subsequently, the effect of surface oxidation/oxidation during the deposition itself for Sc sample is even pronounced from the O K-edge, where the doublet appearing at around 532.4 and 534.4 eV (shown by the pink highlight) can be inferred to t_{2g} (O 2pπ+Sc3d) and e_g (O 2pσ+Sc3d) states due to the possible octahedral ligand field splitting (10Dq), wheras the broad feature 'D' arises due to hybridization of the O 2p with 4sp states of Sc [48, 49]. Similarly, for R_{N_2} = 1.6% sample, both

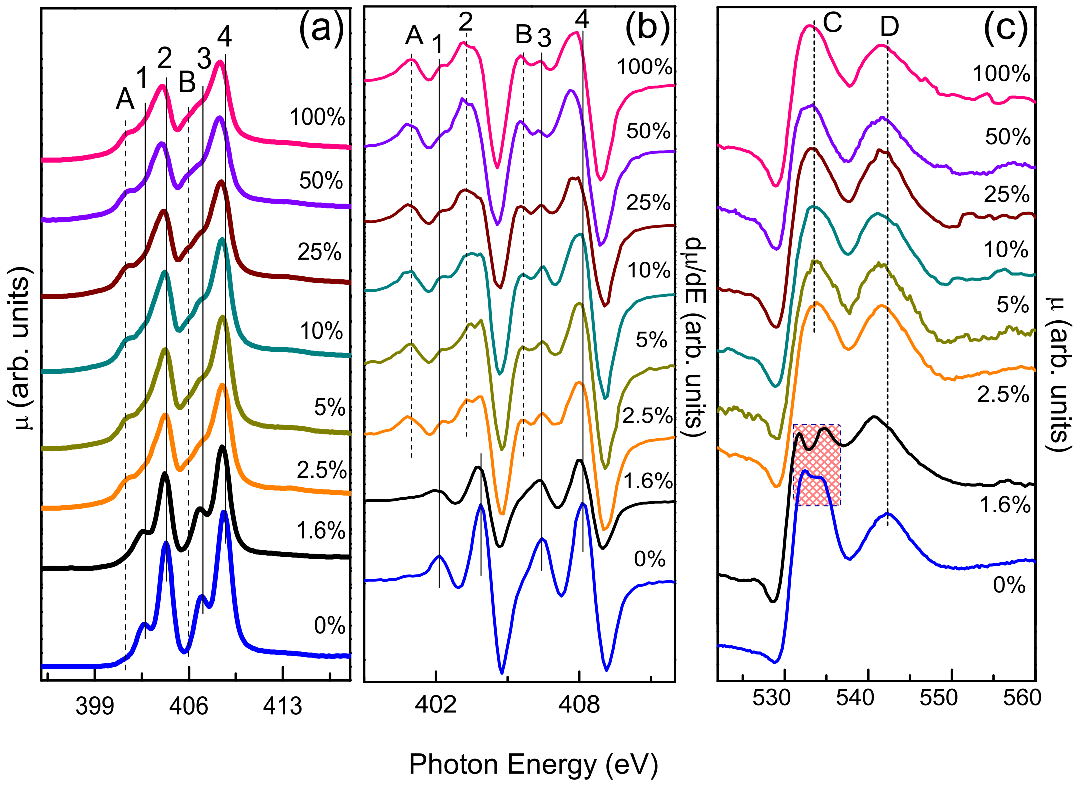


FIG. 2. Normalized SXAS spectra of Sc L_{III}, L_{II} and N K-edge (a), first order derivative of absorption spectra with respect to the photon energy (b) and O K-edge (c) of pure Sc and ScN samples deposited at various R_{N_2}= 1.6, 2.5, 5, 10, 25, 50 and 100%.

Sc L-edge and O K-edge features mimic the same trend as metallic Sc thin film sample, but an increase in 10Dq $\approx 2.9(\pm 0.3)$ eV can be observed, which might be due to the shrink in volume of interstitial ScN during the process of structural transformation from Sc→ScN. Hence, O K-edge spectra confirms the incorporation of bonded O on the sample surface which can be present either in the form of Sc_xO_y (for Sc)/ScO_xN_y (for 1.6% ScN), but, certainly not Sc_2O_3 ($10Dq_{[Sc-O]} = 3.3$ eV), as both samples were of metallic grey in color (as opposed to transparent Sc_2O_3).

With further increase in R_{N_2} from 2.5 - 100%, as can be seen from Figure 2(a) and 2(b), two new pronounced features labelled as 'A' and 'B' arises which can be ascribed to ligand field splitting in the presence of N octahedral environment and noticeable reduction in the intensity of features '1' and '3' can be detected which can be better discerned from the O K-edge spectra. Here, instead of a doublet, a new feature 'C' can be noted. The appearance of similar experimental spectra have also been reported by Kumar et al. in the O K-edge of TiN (T_s = 1023 K) and through a combined study of DFT and abinitio full potential multiscattering (FMS) theory, they concluded that such feature arises due to the presence of substitutional (O_N) and interstitial (O_i) oxygen in a defect complex state ($4O_N+O_i$) [20]. Hence, the contribution of defects in early TMNs are comparable for both ambient and high T_s depositions. As mentioned earlier, since the N K-edge and Sc L-edge appears very close to each other, Nayak et al. have performed theoretical simulations based on the FMS theory and reported a value of 10Dq = 2.1 eV [27], which is in well agreement with the experimentally observed value of $\approx 2.3(\pm 0.3)$ eV obtained in the present work. Even though, it appears from Figure 2(a) that the energy features of '2' and '4' remain almost unaltered with an increase in R_{N_2} flow (2.5% and above), but first order derivatives of the absorption spectra divulged a diverse profile where a new feature (around '2' and less pronounced for feature '4') towards the lower energy side for these samples is clearly visible as evidenced from Figure 2(b), which can be attributed as L_{III} (e_g) and L_{II} (e_g) corresponding to Sc-N bonds. Considering the new features of ScN, the spin-orbit splitting comes out to be $4.8(\pm 0.3)$ eV, well in agreement as reported for ScN [28].

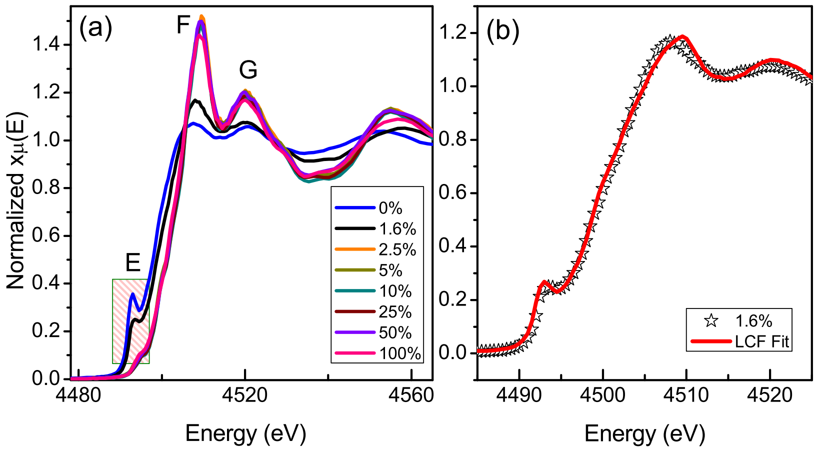


FIG. 3. Normalized Sc K-edge XAFS spectra of metallic Sc and ScN thin film samples deposited at various R_{N_2}= 1.6, 2.5, 5, 10, 25, 50 and 100% (a) and linear combination fit (LCF) of 1.6% sample.

C. X-Ray Absorption Fine Structure

So as to achieve complementary information about the electronic structure, Figure 3(a) depicts the normalized Sc K-edge XANES spectra of Sc and ScN samples deposited at various R_{N_2} flow and Figure 3(b) shows the linear combination fit (LCF) of 1.6% sample. The spectra of Sc and 1.6% ScN sample shows distinct features in comparison to samples deposited at relatively high R_{N_2} flow. In the pre-edge region, an intense feature 'E' can be seen for Sc sample and with R_{N_2} flow at 1.6%, it gets feeble. Such intense pre-edge feature has previously been reported for metallic hcp Ti (for both foil and film) with hexagonal symmetry and has been attributed to 1s $\rightarrow$ 3d electric quadrupole transition ($\Delta l = \pm 2$) [50, 51]. It is conventional that pre-edge features differ due to different geometrical parameters such as, inversion symmetry, co-ordinations and bonding configurations (e.g. bond length, bond angles) etc [52]. For 1.6% sample, the best fit was obtained considering the phase co-existance of both Sc (71.9(±0.9) %) and ScN (28.1(±0.9) %) phases.

On the contrary, with further incorporation of N, the intensity gradually reduces from R_{N_2} = 1.6% to 2.5%, and are alike for rest of the samples. Since, it is well established that Sc is bonded with nearest neighbor N atoms in an octahedral co-ordination sphere with inversion symmetry [28], the emergence of weak pre-edge feature can be ascribed to 1s core electron transition to 3d states of the absorber having a partial contribution from the 2p orbitals of N under the allowed electric dipole transition scheme ($\Delta l = \pm 1$), like in TiN [53].

In addition, nitridation of the samples consequences in continuous shift of the absorption edge (E_0 = taken at 50% of the absorption spectra) [54] towards the higher energy side at E_0 = 4496(±0.3) (Sc), 4497.4(±0.3) (R_{N_2} = 1.6%) and 4500.3(±0.3) (R_{N_2} = 2.5%) eV which can be interpreted in terms of higher core-hole screening due to increase in valence states of Sc from Sc $\rightarrow$ ScN. However, above the absorption edge in the XANES region, the feature 'F' and 'G' can be assigned to 1s $\rightarrow$ 4p electric dipole allowed transitions of a core electron as evidenced for other transition metal compounds e.g. TiC [55]. A diverse trend of feature 'F' can be attributed to the different stacking sequence of Sc (ABAB for hcp) and ScN (ABCABC for fcc) samples, where higher fcc phase fractions result in sharp intense peak (characteristic features of TMNs) in comparison to diffuse kind of feature for Sc thin film samples [56, 57], due to a possible intermixing between 3d quadrupole and 4p dipole states [29]. It is worth mentioning here that in the present study, Sc K-edge of pure Sc thin film sample does not replicate the XAFS spectra of Sc_2O_3 studied by Chassé et al [58], consistent with our XRD data analysis. Since, fluorescence detected XAFS is known to be a bulk sensitive technique with penetration depth ranging in micrometers [51] compared to surface sensitive SXAS where the depth scale ranges only in nanometer scale (≈10 nm) [29], the averaged out bulk information from the XAFS data further confirms the presence of higher surface oxidation in the samples as was evidenced from the Sc L-edge and O K-edge SXAS spectra.

Figure 4(a) and (b) shows the Fourier Transform (FT) moduli $|\chi(R)|$ and the real component [Re $\chi(R)$] of the Sc K-edge EXAFS spectra as a function of radial distance (R-ϕ) and the corresponding best fit, whereas, Figure 4(c) demonstrates the $\chi(k)\times k^3$ spectra. For fitting, hcp and cubic rocksalt type NaCl structure of Sc and ScN were considered having space groups of P63/mmc [59] and Fm$\bar{3}$m [15], respectively. The fitting was performed using LP obtained from the XRD data and the obtained metrical parameters are tabulated in Table II.

For Sc sample, the single shell of FT spectra corresponds to Sc co-ordinated to 7.2(±1.8) Sc atoms each having an atomic pair distance of 3.25(±0.02) Å. Hence, the local co-ordination environment is rather in a distorted hexagonal symmetry which might be responsible

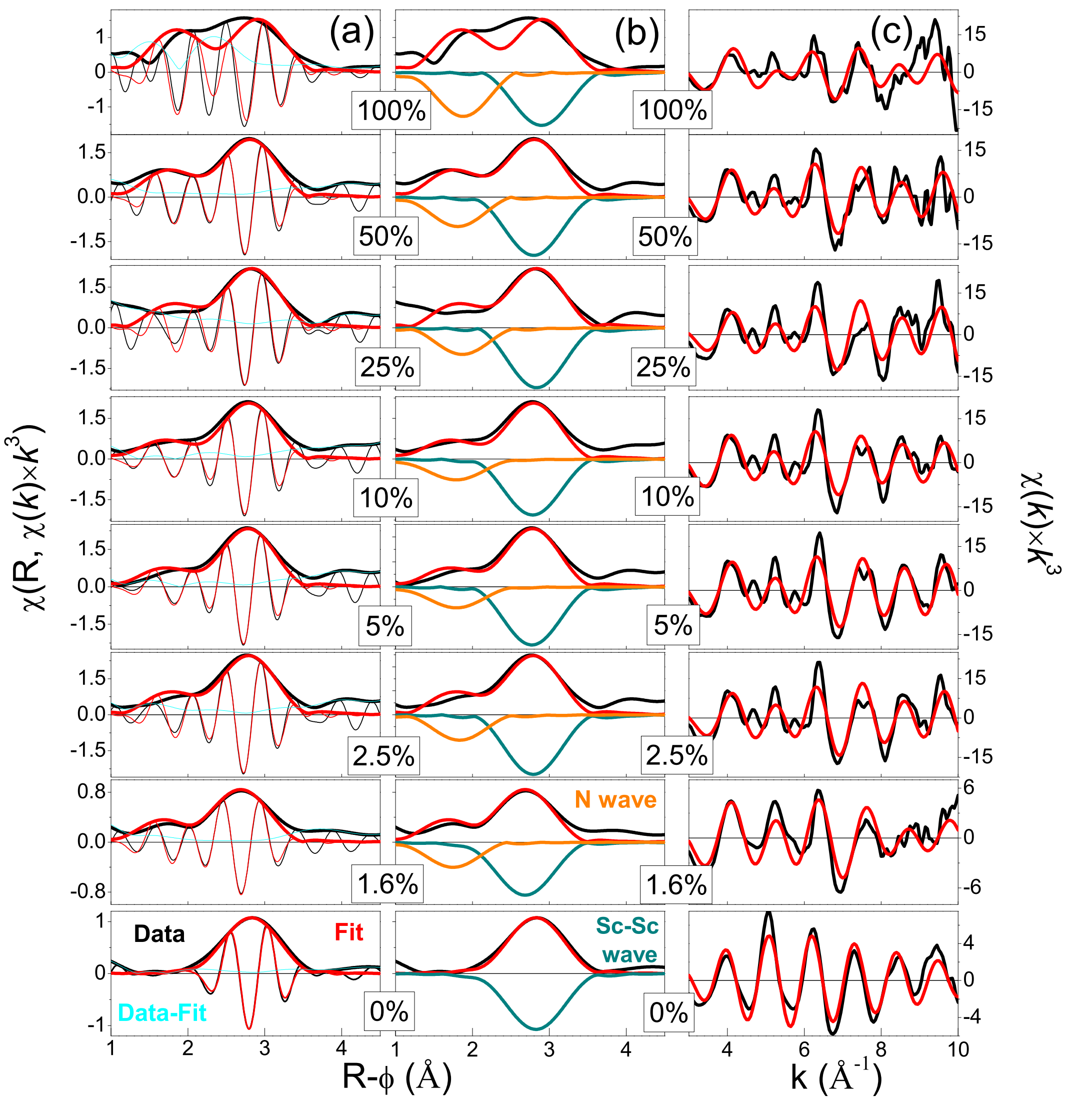


FIG. 4. Comparison of Fourier transform (FT) moduli $\chi(R)$ (a), Re $[\chi(R)]$ (b) in the R range and $\chi(k)\times k^3$ spectra in the k range (c) of Sc and ScN thin film samples deposited at various R_{N_2}= 2.5, 5, 10, 25, 50 and 100%.

for an intense pre-edge feature across the Sc K-edge absorption spectra (Figure 3). In addition, for ScN, considering a theoretical LP of a* = 4.501 Å [15, 30], the first and second nearest neighbor distances can be anticipated at $R^*_{Sc-N} = a/2 = 2.25$ Å and $R^*_{Sc-Sc} = a/\sqrt{2} = 3.18$ Å [60]. As can be seen from Figure 4(a) and (b), the two consecutive maxima distributed over R-ϕ = 1 - 3.9 Å range correspond to the first Sc-N and second Sc-Sc nearest neighbor bonds. For 1.6% sample, even though it was not possible to obtain the structural parameters from the XRD data due to appearance of a single broad peak, but the obtained EXAFS fitting parameters from Table II clearly states the interstitial nature of the sample with less content of both N and Sc atoms in the first and second shell co-ordinations to fully evolve to the fcc phase considering only the ScN phase during fitting. Apparently, at higher R_{N_2} flow (above 2.5%), all ScN samples typically exhibit the octahedral inversion symmetry as expected in NaCl structure and corroborates well with our XRD data within the experimental resolution, as expected. To take into account the optical and mechanical response of the ScN samples, UV-Vis measurements along with nanoindentation tests were performed and are discussed in section III D.

D. Optical & Mechanical Behavior

In order to delve the optical properties with electronic structure, Figure 5(a) shows the Tauc's plot curve of α as a function of incident photon energy for ScN thin film samples deposited at R_{N_2} = 2.5, 5, 10, 25, 50 and 100 % flows. The optical bandgap values were obtained from the intersection of the energy axis by extrapolating the lin-

TABLE II. The metrical parameters obtained from the fitting of the EXAFS data recorded at Sc K-edge. Considering the central atom as Sc, here, N and N$'$ = first and second nearest neighbor co-ordination, R_{Sc-N} and R_{Sc-Sc} = atomic pair distance of the first and second neighbors i.e. Sc-N and Sc-Sc, σ_{Sc-N} and σ_{Sc-Sc} = root mean square displacement obtained from fitting of the first and second shell.

R_{N_2} (%)	N	R_{Sc-N} (Å)	σ_{Sc-N} (Å)	N$'$	R_{Sc-Sc} (Å)	σ_{Sc-Sc} (Å)
0%	-	-	-	7.23	3.25	0.098
	-	-	-	(±1.82)	(±0.02)	(±0.01)
1.6%	2.252	2.189	0.064	5.78	3.136	0.098
	(±0.676)	(±0.025)	(±0.031)	(±1.59)	(±0.021)	(±0.017)
2.5%	4.48	2.25	3.358	9.25	3.20	0.06
	(±1.34)	(±0.02)	-	(±2.55)	(±0.02)	(±0.02)
5%	7.42	2.23	2.488	8.93	3.19	0.06
	(±2.22)	(±0.03)	(±0.03)	(±2.42)	(±0.02)	(±0.02)
10%	8.03	2.23	0.11	9.40	3.20	0.07
	(±2.4)	(±0.03)	(±0.03)	(±2.6)	(±0.02)	(±0.02)
25%	5.12	2.27	0.06	7.46	3.23	0.04
	(±1.5)	(±0.03)	(±0.03)	(±2.14)	(±0.02)	(±0.02)
50%	5.86	2.23	0.07	8.25	3.21	0.07
	(±1.75)	(±0.03)	-	(±2.3)	(±0.02)	(±0.02)
100%	7.61	2.28	0.07	5.39	3.27	0.05
	(±2.28)	(±0.02)		(±1.62)	(±0.02)	(±0.03)

ear least square fitting curve around the inflection point using the Tauc relation, $\alpha h\nu = A\ (h\nu - E_g)^{\frac{1}{2}}$ for direct transitions [61] where, $h\nu$ = photon energy, E_g = optical bandgap and A is proportionality constant. As mentioned earlier in section III B, both Sc and 1.6% ScN sample were metallic in nature and did not show any absorption in the whole energy spectrum. Beyond R_{N_2} = 1.6 %, all the samples showed semiconducting behavior with a well-defined optical bandgap. In this context, it is to be mentioned that ScN exhibits an indirect bandgap of 0.9 eV [31, 62], whereas, two direct bandgaps at 2.2 and 3.8 eV were reported [3, 30]. Generally, only the first direct bandgap has been reported (see Table I) and will also be considered in this work. The large variation in the reported bandgap values have been attributed either to the formation of defect states near the conduction band for n-type degenerate semiconductor termed as 'Burstein-Moss band filling effect' [63] or to the strain mediated effects which can modulate the energy band shifts to a certain extent [64]. The trend was non-linear in nature with a maximum value of 2.62 eV for 5% ScN sample and a minimum of 2.25 eV for sample deposited at R_{N_2} = 100%. Thus, the variations in the bandgap values in the present study can be inferred primarily to the contribution of defects as the stress-strain mediated changes might be negligible in the present case, as all the samples were deposited on amorphous quartz substrates. It is worth mentioning here that the bandgap values of ScO_xN_y and Sc_2O_3 are reported to be 3.25 eV and 5.6 eV, respectively [27], which are way higher than the obtained bandgaps in the present study.

Figure 5(b) demonstrates the measured indentation hardness (H) and modulus (E) of the ScN thin film samples as a function of R_{N_2} whereas, Figure 5(c) illustrates the ratio of (H^3/E^2) which corresponds to the resistance of ScN samples to plastic deformation as a function of H. As expected, Sc exhibits the lowest H and E values of 8(±1.3) and 79(±7) GPa, respectively. With nitridation, both H and E increases monotonically from 15 - 27 GPa and 152 - 268 GPa for the ScN samples, except for R_{N_2} = 2.5%, which is close to the calculated value of 25 GPa for ScN [65]. At R_{N_2} = 2.5%, the values maximizes at 34(±6.2) and 476(±157) GPa, respectively, which can be attributed to the highest density as estimated from the XRD data [66], smaller grain size and strong (111) and (222) texturing of the ScN sample [67]. With increase in R_{N_2} = 2.5 - 50%, from the XRD data (Figure 1), it is evident that the XRD peaks shift chronically along the lower diffraction angle demonstrating in-plane compressive residual stress. In addition, the texturing effect is well retained, which is known to be the highest density plane and the corresponding H is considered to be the highest along the (111) plane [68] and are the reasons behind the increase in H. Even though, the texturing effect is witnessed for R_{N_2} = 100% sample only along the (111) plane, but the decrease in grain size in turn elevates the grain boundaries which could lead to plausible rise in H value [67]. Furthermore, the gradual rise in E values from Sc→ScN can be attributed to the strong covalent bond formation of Sc-N than metallic Sc-Sc bonds in Sc [44], as evidenced from our XAFS study. Apart from this, the ScN thin film samples also show a propitious resistance to plastic deformation (H^3/E^2) following a linear trend with increase in H values [69]. However, it is to be mentioned here that typically the values of H and E are likely comparable in the range of R_{N_2} = 2.5 - 100% within the error bars.

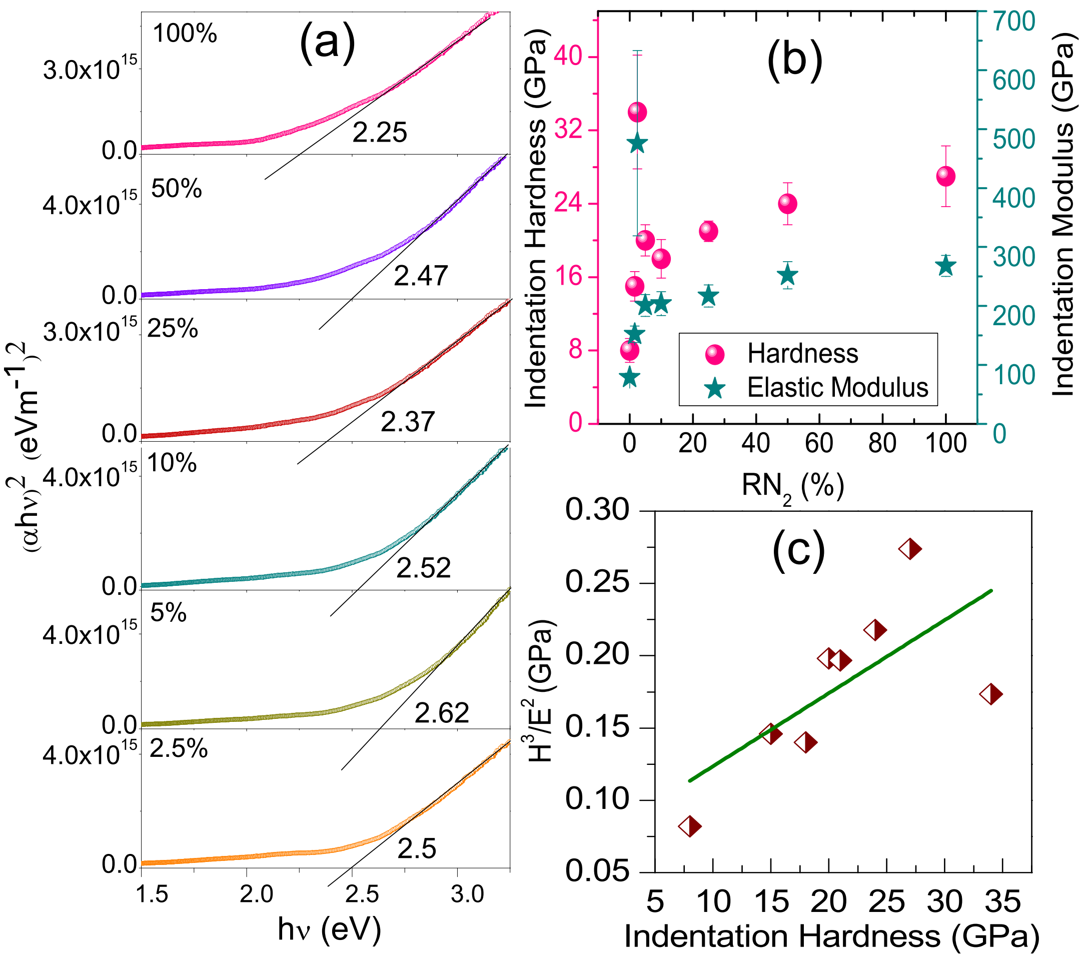


FIG. 5. Obtained direct bandgaps from the Tauc's plot of the absorption co-efficient as a function of incident photon energy (a), nano-indentation hardness and modulus with error bars (b) and the ratio of (H^3/E^2) depicting resistance to plastic deformation of ScN thin film samples as a function of hardness (c), deposited at room-temperature at various R_{N_2}= 1.6, 2.5, 5, 10, 25, 50 and 100% flow.

IV. DISCUSSION

At an early stage of subtle nitridation (R_{N_2} = 1.6%), combined XRD and XAFS study reveal formation of an interstitial compound in the intermediate stage during evolution from hcp Sc to fcc ScN and with higher content of N, at R_{N_2} = 2.5% and above, adaptation of fcc phase with octahedral symmetry were observed. Here, it is interesting to note that likewise in early TMNs, ScN does not possess bimetallic phases (e.g. Ti_2N, V_2N and Cr_2N) at ambient temperature and pressure, and rather manifests a large homogeneity range (retaining NaCl type rocksalt crystal structure from as low as R_{N_2} = 2.5% to as high as 100%). In contrary, with increase in N_2 atomic %, the overall crystal lattice withstands a minimal lattice expansion of only ≈1.7% at highest R_{N_2} = 100% for ScN. It could be well discerned in terms of higher interstitial lattice volume of ScN compared to other early TMNs in the series (TiN, VN and CrN). Since, the early TMNs exhibit NaCl type rocksalt crystal structure, the corresponding octahedral interstial site occupancy of nitrogen atoms would be at edge centers of the unit cell i.e. at ($\frac{1}{2}$,0,0), (0,$\frac{1}{2}$,0), (0,0,$\frac{1}{2}$) and at the center i.e. ($\frac{1}{2}$,$\frac{1}{2}$,$\frac{1}{2}$) position of the respective unit cell. Considering the (100) lattice plane of the unit cell, from the simple pictorial overview as shown in Figure 6 for ScN sample, the radius of the interstitial site (R_{int}) can be similarly evaluated for early TMNs by solving two basic equations along the edge and diagonal as,

$$R_{TM}+2R_{int}+R_{TM} = a, ...(i)$$

and,

$$R_{TM}+2R_{TM}+R_{TM} = (a^2 + a^{\ 2})^{\frac{1}{2}}, ...(ii)$$

where, R_{TM} = radius of TM atom, and a = LP of TMN. Hence, the corresponding structural parameters of the early TMNs are enlisted in Table III.

TABLE III. Early transition metals (TM) and their corresponding crystal structures (CS) and lattice parameters (LP_{TM}). In comparison to metal counterparts, lattice parameters (LP_{TMN}) of their nitrides with calculated radius of interstitial octahedral site (R_{int}) have been tabulated below.

TM	CS	LP_{TM} (Å)	TMN	LP_{TMN} (Å)	R_{int} (Å)	Ref.
Sc	hcp	a = b = 3.309, c = 5.273	ScN	4.501	0.659	[70]
Ti	hcp	a = b = 2.951, c = 4.684	TiN	4.24	0.621	[71]
V	bcc	a = 3.03	VN	4.139	0.606	[66]
Cr	bcc	a = 2.885	CrN	4.14	0.606	[72]

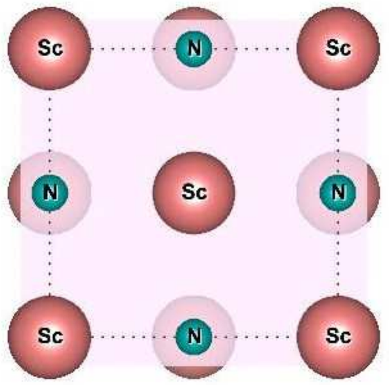


FIG. 6. Representative unit cell of ScN (100) crystal plane

From Table III, it is evident that ScN exhibits the largest unit cell with higher fraction of interstitial volume among the early TMNs, leading to a large homogeneity range for retainig the fcc rocksalt phase accompanied with minuscule lattice expansion. In this context, it is worth mentioning that Al et al. have reported that ScN can withstand upto ≈ 20% of N vacancies within the crystal lattice [73]. Such a large homogeneity range has also been witnessed for other early TMNs like TiN_x ($0.67 \leq x \leq 1.3$) [74], VN_x ($0.79 \leq x \leq 0.96$) [75] etc. in the octahedral symmetry. In addition, combined SXAS study at Sc L-edge, N K-edge and O K-edge reveal that the effect of oxidation is less pronounced for nitrided samples compared to their metal/interstitial counterparts. This is due to formation of strong Sc-N covalent bonds which results in relatively high Gibb's energy for oxide formation (-6.48 eV) of ScN than metallic Sc-Sc bonds in pure Sc (-9.43 eV) at ≈298 K [13]. Furthermore, our Sc K-edge XANES spectra confirms distinct evolution from hexagonal to octahedral symmetry complemented with a clear rise in the valence state for ScN samples consistent with our XRD results. The pre-edge features of neither Sc nor ScN resembles the Sc K-edge oxide spectra [58] emphasizing on the higher surface oxidation effect in all these samples. Nonetheless, the local electronic structure as recorded from EXAFS replicates the XRD data with further insight on the presence of local defects in the vicinity of Sc atoms but the visible changes are however marginal for all the samples. Even so, the non-monotonic variations in the optical bandgaps across the whole R_{N_2} range have no clear trend and in this scenario, it is difficult to correlate them in terms of defects as in case of polycrystalline thin film samples, the role of defects are always expected to be higher than epitaxial ScN thin film samples studied so far. Howbeit, the bandgap values in the present study lie well within the energy regime of ScN thin film samples as observed for high T_s depositions on single crystal substrates. Additionally, the hardness and indentation modulus agrees well during the evolution from Sc→ScN with a linearly increasing trend of resistance to plastic deformation and also matches well with those reported in the literature [76, 77] within the experimental error bars.

CONCLUSION

In lieu of adoptation of high tempearture depositions hitherto, as-deposited ScN thin film samples exhibited highly textured orientation along the (111) and (222) reflection planes grown on amorphous quartz substrates due to preferable highest surface energy configuration at low temperature regime (here 300 K). SXAS study reveals pronounced incorporation of oxygen in metallic Sc and interstial ScN sample deposited at R_{N_2} = 1.6% flow, although reduction of oxygen content can be witnessed with nitridation of the samples (R_{N_2} = 2.5 - 100%). Complementary XAFS study shows distinct evolution from Sc→ScN, where an intense pre-edge feature stems from non-centrosymmetric distortion for Sc and interstitial ScN (R_{N_2} = 1.6%) sample, whereas, octahedral symmetry was retained by rest of the ScN samples deposited at higher R_{N_2} flow. From UV-Vis measurement, the obtained direct optical bandgaps were found to vary between 2.25 - 2.62 eV for R_{N_2} = 2.5 - 100%, well in agreement with the values routinely reported in literatures for epitaxial ScN thin film samples. Even so, the nano-indentation measurements validates the high hardness of highly elastic ScN thin film samples ranging between 15 - 34 GPa with a monotonically increasing trend in the value of resistance to plastic deformations. In turn, the large homogenity range of Sc-N system has been compared with elemental early 3d transition metal nitride series viz. TiN, VN and CrN to comprehend the phase stability of cubic NaCl rocksalt type structure of ScN thin film samples over large variation in R_{N_2} flow. Hence, cumulative inferences drawn from this work can be epitomized as high vacuum deposition is imperative for high quality ScN samples but an alternate room temperature deposition can be adopted as opposed high T_s, to minimize the unintentional diffusion mechanisms at elevated temperatures finding applications in electronics viz. CMOS integrated circuits, free standing films on plastic substrates.

ACKNOWLEDGMENTS

Authors (SC and RG) are grateful to UGC-DAE CSR, Indore for providing financial support through CSR-IC-BL-62/CSR179-2016-17/843 project. Thanks are due to V. R. Reddy and Anil Gome for XRR measurements, Rakesh Sah for SXAS measurement at BL-01, Indus 2, RRCAT and Layanta Behera for various experiments. SC is thankful to Sanjay Nayak for fruitful discussions and Yogesh Kumar for his extended help in XRD and XAFS measurements. We also thank S. Tokekar, A. J. Pal, A. K. Sinha, D. M. Phase, V. Ganesan and V. Sathe for their kind support and constant encouragements.

[1] B. Saha, A. Shakouri, T. D. Sands, Rocksalt nitride metal/semiconductor superlattices: A new class of artificially structured materials, Applied Physics Reviews 5 (2018) 021101.

[2] P. Patsalas, N. Kalfagiannis, S. Kassavetis, G. Abadias, D. Bellas, C. Lekka, E. Lidorikis, Conductive nitrides: Growth principles, optical and electronic properties, and their perspectives in photonics and plasmonics, Materials Science and Engineering: R: Reports 123 (2018) 1–55.

[3] B. Biswas, B. Saha, Development of semiconducting ScN, Physical Review Materials 3 (2019) 020301.

[4] P. V. Burmistrova, J. Maassen, T. Favaloro, B. Saha, S. Salamat, Y. Rui Koh, M. S. Lundstrom, A. Shakouri, T. D. Sands, Thermoelectric properties of epitaxial ScN films deposited by reactive magnetron sputtering onto MgO (001) substrates, Journal of Applied Physics 113 (2013) 153704.

[5] F. Scholz, Semipolar GaN grown on foreign substrates: a review, Semiconductor Science and technology 27 (2012) 024002.

[6] M. Little, M. Kordesch, Band-gap engineering in sputter-deposited $Sc_xGa_{1-x}N$, Applied Physics Letters 78 (2001) 2891–2892.

[7] J. Su, F. Niekiel, S. Fichtner, L. Thormaehlen, C. Kirchhof, D. Meyners, E. Quandt, B. Wagner, F. Lofink, AlScN-based MEMS magnetoelectric sensor, Applied Physics Letters 117 (2020) 132903.

[8] F. Tasnádi, B. Alling, C. Höglund, G. Wingqvist, J. Birch, L. Hultman, I. A. Abrikosov, Origin of the anomalous piezoelectric response in wurtzite $Sc_xAl_{1-x}N$ alloys, Physical review letters 104 (2010) 137601.

[9] V. Rawat, Y. K. Koh, D. G. Cahill, T. D. Sands, Thermal conductivity of (Zr, W) N/ScN metal/semiconductor multilayers and superlattices, Journal of Applied Physics 105 (2009) 024909.

[10] A. Belosludtsev, K. Juškevičius, L. Ceizaris, R. Samuilovas, S. Stanionytė, V. Jasulaitienė, S. Kičas, Correlation between stoichiometry and properties of scandium oxide films prepared by reactive magnetron sputtering, Applied Surface Science 427 (2018) 312–318.

[11] B. Saha, M. Garbrecht, J. A. Perez-Taborda, M. H. Fawey, Y. R. Koh, A. Shakouri, M. Martin-Gonzalez, L. Hultman, T. D. Sands, Compensation of native donor doping in ScN: Carrier concentration control and p-type ScN, Applied Physics Letters 110 (2017) 252104.

[12] P. Eklund, S. Kerdsongpanya, B. Alling, Transition-metal-nitride-based thin films as novel energy harvesting materials, Journal of Materials Chemistry C 4 (2016) 3905–3914.

[13] J. More-Chevalier, S. Cichoň, L. Horák, J. Bulíř, P. Hubík, Z. Gedeonová, L. Fekete, M. Poupon, J. Lančok, Correlation between crystallization and oxidation process of ScN films exposed to air, Applied Surface Science 515 (2020) 145968.

[14] D. Rao, B. Biswas, E. Flores, A. Chatterjee, M. Garbrecht, Y. R. Koh, V. Bhatia, A. I. K. Pillai, P. E. Hopkins, M. Martin-Gonzalez, et al., High mobility and high thermoelectric power factor in epitaxial ScN thin films deposited with plasma-assisted molecular beam epitaxy, Applied Physics Letters 116 (2020) 152103.

[15] J. Casamento, J. Wright, R. Chaudhuri, H. Xing, D. Jena, Molecular beam epitaxial growth of scandium nitride on hexagonal SiC, GaN, and AlN, Applied Physics Letters 115 (2019) 172101.

[16] A. Le Febvrier, N. Tureson, N. Stilkerich, G. Greczynski, P. Eklund, Effect of impurities on morphology, growth mode, and thermoelectric properties of (1 1 1) and (0 0 1) epitaxial-like ScN films, Journal of Physics D: Applied Physics 52 (2018) 035302.

[17] D. Rao, B. Biswas, S. Acharya, V. Bhatia, A. I. K. Pillai, M. Garbrecht, B. Saha, Effects of adatom mobility and Ehrlich–Schwoebel barrier on heteroepitaxial growth of scandium nitride (ScN) thin films, Applied Physics Letters 117 (2020) 212101.

[18] Y. Kumagai, N. Tsunoda, F. Oba, Point defects and p-type doping in ScN from first principles, Physical Review Applied 9 (2018) 034019.

[19] S. Acharya, A. Chatterjee, V. Bhatia, A. I. K. Pillai, M. Garbrecht, B. Saha, Twinned growth of ScN thin films on lattice-matched GaN substrates, Materials Research Bulletin (2021) 111443.

[20] R. Kumar, S. Nayak, M. Garbrecht, V. Bhatia, A. Indiradevi Kamalasanan Pillai, M. Gupta, S. Shivaprasad, B. Saha, Clustering of oxygen point defects in transition metal nitrides, Journal of Applied Physics 129 (2021) 055305.

[21] M. S. Haseman, B. A. Noesges, S. Shields, J. S. Cetnar, A. N. Reed, H. A. Al-Atabi, J. H. Edgar, L. J. Brillson, Cathodoluminescence and x-ray photoelectron spectroscopy of ScN: Dopant, defects, and band structure,

APL Materials 8 (2020) 081103.

[22] M. Moram, Z. Barber, C. Humphreys, The effect of oxygen incorporation in sputtered scandium nitride films, Thin Solid Films 516 (2008) 8569–8572.

[23] T. Ohgaki, K. Watanabe, Y. Adachi, I. Sakaguchi, S. Hishita, N. Ohashi, H. Haneda, Electrical properties of scandium nitride epitaxial films grown on (100) magnesium oxide substrates by molecular beam epitaxy, Journal of Applied Physics 114 (2013) 093704.

[24] S.-L. Tsai, T. Hoshii, H. Wakabayashi, K. Tsutsui, T.-K. Chung, E. Y. Chang, K. Kakushima, Room-temperature deposition of a poling-free ferroelectric AlScN film by reactive sputtering, Applied Physics Letters 118 (2021) 082902.

[25] J. S. Cetnar, A. N. Reed, S. C. Badescu, S. Vangala, H. A. Smith, D. C. Look, Electronic transport in degenerate (100) scandium nitride thin films on magnesium oxide substrates, Applied Physics Letters 113 (2018) 192104.

[26] J. More-Chevalier, S. Cichoň, J. Bulíř, M. Poupon, P. Hubík, L. Fekete, J. Lančok, Electrical and optical properties of scandium nitride nanolayers on MgO (100) substrate, AIP Advances 9 (2019) 015317.

[27] S. Nayak, M. Baral, M. Gupta, J. Singh, M. Garbrecht, T. Ganguli, S. Shivaprasad, B. Saha, Rigid-band electronic structure of scandium nitride across the n-type to p-type carrier transition regime, Physical Review B 99 (2019) 161117.

[28] B. Biswas, S. Nayak, V. Bhatia, A. I. K. Pillai, M. Garbrecht, M. H. Modi, M. Gupta, B. Saha, Interfacial chemistry and electronic structure of epitaxial lattice-matched TiN/$Al_{0.72}Sc_{0.28}N$ metal/semiconductor superlattices determined with soft x-ray scattering, Journal of Vacuum Science & Technology A: Vacuum, Surfaces, and Films 38 (2020) 053201.

[29] G. S. Henderson, F. M. De Groot, B. J. Moulton, X-ray absorption near-edge structure (xanes) spectroscopy, Reviews in Mineralogy and Geochemistry 78 (2014) 75–138.

[30] R. Deng, B. Ozsdolay, P. Zheng, S. Khare, D. Gall, Optical and transport measurement and first-principles determination of the ScN band gap, Physical Review B 91 (2015) 045104.

[31] H. A. Al-Brithen, A. R. Smith, D. Gall, Surface and bulk electronic structure of ScN (001) investigated by scanning tunneling microscopy/spectroscopy and optical absorption spectroscopy, Physical Review B 70 (2004) 045303.

[32] C. Stampfl, W. Mannstadt, R. Asahi, A. J. Freeman, Electronic structure and physical properties of early transition metal mononitrides: Density-functional theory LDA, GGA, and screened-exchange LDA FLAPW calculations, Physical Review B 63 (2001) 155106.

[33] D. Gall, M. Stoehr, J. Greene, Vibrational modes in epitaxial $Ti_{1-x}Sc_xN$ (001) layers: an ab initio calculation and Raman spectroscopy study, Physical Review B64 (2001) 174302.

[34] B. Saha, J. Acharya, T. D. Sands, U. V. Waghmare, Electronic structure, phonons, and thermal properties of ScN, ZrN, and HfN: A first-principles study, Journal of Applied Physics 107 (2010) 033715.

[35] C. Braun, Parratt 32 Program for Reflectivity Fitting, Hahn-Meitner Institute, Berlin (1999).

[36] L. G. Parratt, Surface studies of solids by total reflection of X-rays, Physical Review B 95 (1954) 359–369.

[37] D. M. Phase, M. Gupta, S. Potdar, L. Behera, R. Sah, A. Gupta, Development of soft X-ray polarized light beamline on Indus-2 synchrotron radiation source, AIP Conference Proceedings 1591 (2014) 685–686.

[38] W. A. Caliebe, V. Murzin, A. Kalinko, M. Görlitz, High-flux XAFS-beamline P64 at PETRA III, in: AIP conference proceedings, volume 2054, AIP Publishing LLC, 2019, p. 060031.

[39] B. Ravel, M. Newville, ATHENA, ARTEMIS, HEPHAESTUS: data analysis for X-ray absorption spectroscopy using IFEFFIT, Journal of Synchrotron Radiation 12 (2005) 537–541.

[40] B. K. Teo, Extended x-ray absorption fine structure (EXAFS) spectroscopy: techniques and applications, in: EXAFS Spectroscopy, Springer, 1981, pp. 13–58.

[41] S. D. Conradson, T. Durakiewicz, F. J. Espinosa-Faller, Y. Q. An, D. A. Andersson, A. R. Bishop, K. S. Boland, J. A. Bradley, D. D. Byler, D. L. Clark, et al., Possible Bose-condensate behavior in a quantum phase originating in a collective excitation in the chemically and optically doped Mott-Hubbard system UO_{2+x}, Physical Review B 88 (2013) 115135.

[42] A. Murphy, Band-gap determination from diffuse reflectance measurements of semiconductor films, and application to photoelectrochemical water-splitting, Solar Energy Materials and Solar Cells 91 (2007) 1326–1337.

[43] R. Ramaseshan, F. Jose, S. Rajagopalan, S. Dash, Preferentially oriented electron beam deposited TiN thin films using focused jet of nitrogen gas, Surface Engineering 32 (2016) 834–839.

[44] P. Panda, R. Ramaseshan, Effects of Cr doping on the mechanical properties of AlN films grown by the co-sputtering technique, Ceramics International 45 (2019) 1755–1760.

[45] F. H. Spedding, A. Daane, K. Herrmann, The crystal structures and lattice parameters of high-purity scandium, yttrium and the rare earth metals, Acta Crystallographica 9 (1956) 559–563.

[46] X. Bai, M. Kordesch, Structure and optical properties of ScN thin films, Applied surface science 175 (2001) 499–504.

[47] D. Gall, I. Petrov, L. Madsen, J.-E. Sundgren, J. Greene, Microstructure and electronic properties of the refractory semiconductor ScN grown on MgO (001) by ultra-high-vacuum reactive magnetron sputter deposition, Journal of Vacuum Science & Technology A: Vacuum, Surfaces, and Films 16 (1998) 2411–2417.

[48] J. G. Chen, NEXAFS investigations of transition metal oxides, nitrides, carbides, sulfides and other interstitial compounds, Surface Science Reports 30 (1997) 1–152.

[49] F. De Groot, J. Fuggle, B. Thole, G. Sawatzky, $L_{2,3}$ x-ray-absorption edges of d_0 compounds: K^+, Ca^{2+}, Sc^{3+}, and Ti^{4+} in O h (octahedral) symmetry, Physical Review B 41 (1990) 928.

[50] Z. Yong, T. Liu, T. Uruga, H. Tanida, D. Qi, A. Rusydi, A. T. Wee, Ti-doped ZnO thin films prepared at different ambient conditions: electronic structures and magnetic properties, Materials 3 (2010) 3642–3653.

[51] D. Mardare, A. Yildiz, R. Apetrei, P. Rambu, D. Florea, N. G. Gheorghe, D. Macovei, C. M. Teodorescu, D. Luca, The Meyer-Neldel rule in amorphous TiO_2 films with different Fe content, Journal of Materials Research 27 (2012) 2271.

[52] N. Jiang, D. Su, J. Spence, Determination of Ti coordination from pre-edge peaks in Ti K-edge XANES, Physical Review B 76 (2007) 214117.

[53] M.-H. Tuilier, M.-J. Pac, M. Gîrleanu, G. Covarel, G. Arnold, P. Louis, C. Rousselot, A.-M. Flank, Electronic and atomic structures of $Ti_{1-x}Al_xN$ thin films related to their damage behavior, Journal of Applied Physics 103 (2008) 083524.

[54] Y. Kumar, A. Tayal, W. Caliebe, M. Gupta, Study of carbon doped cobalt mononitride thin films, Applied Surface Science (2021) 150443.

[55] V. Moisy-Maurice, C. De Novion, An application of Ti-K X-ray absorption edges and fine structures to the study of substoichiometric titanium carbide TiC_{1-x}, Journal de Physique 49 (1988) 1737–1751.

[56] A. Longo, L. Sciortino, F. Giannici, A. Martorana, Crossing the boundary between face-centred cubic and hexagonal close packed: the structure of nanosized cobalt is unraveled by a model accounting for shape, size distribution and stacking faults, allowing simulation of XRD, XANES and EXAFS, Journal of Applied Crystallography 47 (2014) 1562–1568.

[57] Seema, A. Tayal, S. Amir, S. Pütter, S. Mattauch, M. Gupta, et al., Structural, electronic, and magnetic properties of Co4N thin films deposited using HiPIMS, Journal of Alloys and Compounds (2020) 158052.

[58] M. Chassé, A. Juhin, D. Cabaret, S. Delhommaye, D. Vantelon, G. Calas, Influence of crystallographic environment on scandium K-edge X-ray absorption near-edge structure spectra, Physical Chemistry Chemical Physics 20 (2018) 23903–23912.

[59] S. Chowdhury, R. Gupta, S. Prakash, L. Behera, D. Phase, M. Gupta, Study of scandium nitride thin films deposited using ion beam sputtering, in: AIP Conference Proceedings, volume 2265, AIP Publishing LLC, 2020, p. 030312.

[60] M.-H. Tuilier, M.-J. Pac, G. Covarel, C. Rousselot, L. Khouchaf, Structural investigation of thin films of Ti1- xAlxN ternary nitrides using Ti K-edge X-ray absorption fine structure, Surface and Coatings Technology 201 (2007) 4536–4541.

[61] K. Kuriyama, Y. Takahashi, F. Sunohara, Optical band gap of Zn_3N_2 films, Physical Review B 48 (1993) 2781.

[62] A. Qteish, P. Rinke, M. Scheffler, J. Neugebauer, Exact-exchange-based quasiparticle energy calculations for the band gap, effective masses, and deformation potentials of ScN, Physical Review B 74 (2006) 245208.

[63] E. Burstein, Anomalous optical absorption limit in InSb, Physical review 93 (1954) 632.

[64] S. Tamleh, G. Rezaei, J. Jalilian, Stress and strain effects on the electronic structure and optical properties of ScN monolayer, Physics Letters A 382 (2018) 339–345.

[65] M. A. Aslam, Z. Ding, Prediction of thermodynamically stable compounds of the sc–n system under high pressure, ACS omega 3 (2018) 11477–11485.

[66] J. Caicedo, G. Zambrano, W. Aperador, L. Escobar-Alarcon, E. Camps, Mechanical and electrochemical characterization of vanadium nitride (VN) thin films, Applied Surface Science 258 (2011) 312–320.

[67] Q. Wu, J. Liang, J. Liu, W. Bing, X. Long, S. Luo, Characteristics of microstructure and mechanical properties of Sc films as a function of substrate temperature, Applied surface science 258 (2012) 7421–7424.

[68] B. V. Sarada, C. L. Pavithra, M. Ramakrishna, T. N. Rao, G. Sundararajan, Highly (111) textured copper foils with high hardness and high electrical conductivity by pulse reverse electrodeposition, Electrochemical and Solid State Letters 13 (2010) D40.

[69] P. Mayrhofer, C. Mitterer, J. Musil, Structure–property relationships in single-and dual-phase nanocrystalline hard coatings, Surface and Coatings Technology 174 (2003) 725–731.

[70] H. Al-Brithen, A. R. Smith, Molecular beam epitaxial growth of atomically smooth scandium nitride films, Applied Physics Letters 77 (2000) 2485–2487.

[71] P. Patsalas, C. Charitidis, S. Logothetidis, The effect of substrate temperature and biasing on the mechanical properties and structure of sputtered titanium nitride thin films, Surface and Coatings Technology 125 (2000) 335–340.

[72] J. Olaya, S. Rodil, S. Muhl, L. Huerta, Influence of the energy parameter on the microstructure of chromium nitride coatings, Surface and Coatings technology 200 (2006) 5743–5750.

[73] H. A. Al-Brithen, E. M. Trifan, D. C. Ingram, A. R. Smith, D. Gall, Phase stability, nitrogen vacancies, growth mode, and surface structure of ScN (001) under Sc-rich conditions, Journal of crystal growth 242 (2002) 345–354.

[74] I. Schramm, M. J. Jõesaar, J. Jensen, F. Mücklich, M. Odén, Impact of nitrogen vacancies on the high temperature behavior of $(Ti_{1-x}Al_x)N_y$ alloys, Acta Materialia 119 (2016) 218–228.

[75] R. Pompe, Some thermochemical properties of the system vanadium—nitrogen and vanadium—carbon—nitrogen in the temperature range 1000–1550 C, Thermochimica Acta 57 (1982) 273–281.

[76] D. Gall, I. Petrov, N. Hellgren, L. Hultman, J. Sundgren, J. Greene, Journal of Applied Physics 84 (1998) 6034–6041.

[77] M. A. Moram, Z. H. Barber, C. J. Humphreys, T. Joyce, P. Chalker, Young's modulus, Poisson's ratio, and residual stress and strain in (111)-oriented scandium nitride thin films on silicon, Journal of Applied Physics 100 (2006) 023514.

PAPER • OPEN ACCESS

A Review on forecasting the photovoltaic power Using Machine Learning

To cite this article: Amit Kumar Mittal *et al* 2022 *J. Phys.: Conf. Ser.* **2286** 012010

View the article online for updates and enhancements.

A Review on forecasting the photovoltaic power Using Machine Learning

Amit Kumar Mittal[1, 4]**, Dr. Kirti Mathur**[2]**, Shivangi Mittal**[3]
[1] Assistant Professor Computer Engineering IET DAVV, Indore, Madhya Pradesh, India
[2] Associate Professor IIPS DAVV, Indore, Madhya Pradesh, India
3 Lecturer Electrical Engineering Govt. Polytechnic College Dewas Madhya Pradesh, India

[4]**Email**: amittal@ietdavv.edu.in

Abstract. In this review paper on different forecasting method of the solar power output for effective generation of the power grid and proper management of transfer rate of energy per unit area occurred into the solar PV system. Essential part in focusing the prediction of solar power is irradiance and temperature. The irradiance can be forecasted by many algorithm and method is applied in prediction of generation of Short-term photovoltaic power and long term solar power forecasting. And many papers describes on numerical weather forecasting and some algorithm like neural networks or support vector regression for two step approach for predicting the PV power. In this review shown that methods like Bagging Model , deep learning, genetic algorithm, random forest, gradient boosting and artificial neural network. We found that for enhancing the performance of predicting PV power many authors proposed the ensemble method that is the hybrid models of different algorithm. And I found that on this review process ensemble methods show that good results and improve the forecasting solar PV power.

Keyword
Photovoltaic Power, Irradiance, Machine learning, LSTM, Performance, forecasting, deep learning,.

1. Introduction

In the era of renewable energy, we have focused on energy produced by solar cell and with the large drop in prices of photovoltaic (PV) cell the use of solar energy has increased [1]. Today one of the renewable energy Source is PV power and in the future electricity generation it plays important role. A great part of net electricity capacity of the globe growth occur in year 2017. Between years 2019 to 2024 the net renewable energy is expected to grow 50 percent globally. The proportion of solar power in renewable energy is 60 percent [4].

The solar power forecasting has some benefit like efficient operation in the power grid, optimization of energy into the PV power system, power plant Schedule, Congestion management, Trade the produce energy in the energy market, Price reduction for power generation.

Solar energy forecasting can be done by many ways. Machine learning and deep learning is most useful method for it. Various ways to forecast the PV power have been researched and scientists have gained surprising output. PV power forecasting methods are statistical approaches, machine learning techniques, artificial neural network, and numerical weather prediction of physical models by satellite images and hybrid method that is combination of the different methods.

2. Background

Solar panels are designed with the help of PV cells which is created by semi-conductors. The sun's energy is absorbed by the cell and these cells convert irradiance into electrical energy. The photons strike on the cell and knock out electrons. These electrons with great kinetic energy flows in the form of electrical current [18]. The higher electricity will flow when the light intensity is more. This generated solar electricity produced by Panels stored in batteries to use power in the night. Solar panel generated electricity could be On-grid or Off-grid [19].
Machine learning may be a sub-field of computing and is assessed as man-made intelligence. The relationship between inputs and outputs features is Machine learning models. Supervised learning, unsupervised learning, and Reinforcement learning are three types of machine learning. Generally we use supervised learning and unsupervised learning for predicting the power.

3. Literature Work

Sunghyeon Choi and et.al [3] explain to predict solar energy output using Bagging model. Choice tree is used as a base learner in bagging model. Author explain bagging model and add the previous results as a new data and its result defined that model using previous results shows more accuracy as compared to model with single model learner. In this model simulation concludes that in the bagging predictor using an ensemble model as a base learner overall performance was improved than using a decision tree-based bagging predictor. The error rate reduced near fifty percent by adding previous results along weather conditions. The ensemble models were random forest, LightGBM and XGBoost. In these models performance was not changed very effectively. Main issue is that the high mean absolute error metric achieved. For optimum condition author could not match each model's hyper parameter. To improve the model's accuracy required optimization of the hyper parameter and data cleaning.

Peder Bacher et. al [4]describes about forecasting of PV power production in online manner. Author presented the PV power prediction of data on hourly basis for up to thirty six hour. The data is used to observe the solar energy from 21 rooftops PV panel system in Denmark. Method defined a two-stage process of statistic reorganization of the solar power from very clear sky. Forecast the standard solar power are evaluated with linear time series system. In this literature autoregressive (AR) and AR with exogenous input (ARX) models also evaluated then consider for predictions of numerical weather as input. Results defined forecasted two hours of solar power for predictions of numerical weather input.

Dongha Shin et. al [5] takes a data of Korea. Temperature, precipitation, humidity, wind direction, cloudiness and sunshine are the various parameters which takes in process and he also apply various learning algorithms for weather forecast on three days period. This paper defines weather forecast and the model like Recurrent Neural network, artificial neural network, Dynamic neural network and long short term memory including sunshine and solar radiation data of past two hours, just to achieve the best prediction results. Performance is evaluated using Root mean squared error, Correlation , Mean absolute Error and BIAS.

Farid Touati, Amith Khandakar et. al [7] describes a customized PV system to analyzing, monitoring and evaluation of the working of PV system using different weather attribute. To neglect the issue of Overfiting many techniques of feature selection were showed. For PV power forecasting various parameters of environment and electrical have taken consideration and compare between the different Machine Learning method and distinct feature selection methods. It support to resemble an Artificial Neural Network (ANN) model. Author also depicted opportunities of tuning the ANN by changing the hidden layers by altering the algorithm for training. Author showed that describing the calibration tools and developing an in house customized Data acquisition system can improve the performance that is feasible for observing Photovoltaic model in Qatar's weather datasets with the help of hardware and software.

Zhi-Feng Liu , Ling-Ling Li et. al [9] in this Literature Chicken swarm optimizer (CSO) has been changed to Improved Chicken swarm optimizer (ICSO). Author proposes a prediction model for short term PV power. This input model first determines the correlation coefficient and then chicken swarm optimizing is done. The ICSO-ELM model suggested prediction of the short-term PV output in 3 different climatic situations. Effective solar PV prediction can support the power grid to schedule the dispatch of power and support the evolution and application of clean energy.

Gangqiang Li et. al [10] combined deep learning convolutional neural network (CNN) with long-short term memory recurrent neural network (LSTM) in PV output power forecasting using historical dataset of past one year. This paper presents the continuous days PV output power and shows past power data having different dates, modifying performance of PV systems predictions, showing the forecast with scattered plot as per season. Using PV real dataset of Belgium, the numerical results demonstrated it to be very efficient . Performance evaluation for 15 min ahead and 45 min ahead forecast data through different metrics such as RMS Error and MA Error.

Dan A. Rosa de Jesus et. al [11] sponged a new idea for predicting solar photovoltaic power using Hybrid Deep Neural Network (HDNN) model. Also including, rainy and cloudy days for PV power prediction and integrating HDNN model with dispatching algorithms simulated, the transactive energy scenarios with high level of uncertainty. It takes a dataset of Ashland Oregon region for PV power. In this paper approach is improved in prior one hour forecast to next three days in a season.

Debasish Patnaik et. al [12] explored a different method for solar photovoltaic forecasting along with comparison. He gathered a real time data model from Odisha, India and predicted output of solar PV system. It was found that the Genetic algorithm based forecasting was better precise than statistical analysis. Solar PV Output Optimization took temperature as well as Solar Radiation as variables. It predicted about eighteen year Statistical analysis of temperature histogram and provided improved fitness value of PV system for month wise in a year .

Usman Munawar et. al [13] In this paper a method is proposed on quantitative evaluation using numerous models of machine learning like artificial neural network, random forest, extreme gradient boosting (XGBoost) and some feature selection techniques having the feature importance for principle component analysis (PCA). The best method for power prediction of solar in Hawaii, US stated results showing that the XGBoost method with features selected by the PCA method, was a better approach. The random forest and XGBoost models have a lesser use in short-term solar forecasting. Simulation results and case studies show, R^2 score on every repetition using feature as an important method.
Rachit Srivastava et. al [14] has defined, one to six day-ahead hourly radiation of solar forecasting using the Multi adaptive regression splines , M5, Classification and Regression tree and random forest model. The dataset needed for forecasting, was from Gorakhpur, India , a solar radiation resource setup. Results

determined on all four models and the random forest model gave improved results, whereas the CART model gave poor output. All models were provided by precise results for one-day to 6-day-ahead forecasting.

Javier López Gómez et. al [15] combined the dataset from a Numerical Weather Prediction model and machine learning tools for predicting accurate power generation, with 10% less error . A real dataset was used in Artificial Neural Network (ANN) model to forecast the power, which is located in Puglia using solar radiation and temperature data retrieved, the Global Data Assimilation System. Three different scenarios are designed for training and testing cases: first one input dataset from the GDAS was compared with that of the monitoring dataset, and found it best for performance of error metrics and second one is used for predicting the results, which was retrieved from the monitoring data. For the second dataset error values are high third one complete replace the weather data from the monitoring dataset. Author defined that in cloudy season forecasting error were high because the variation in solar insolation and error is low in winter and summer season.

Yi Zhou , Nanrun Zhou et. al [16] said , for prediction of hourly solar power output , the hybrid model which was invented for solar power output forecasting, hourly and correlation coefficient of Pearson was used to calculate the difference of distinct days on the basis of meteorological factors and the dataset alike those from the target forecast day, are selected as the training set of ELM. This operation effectively increased the amount of useful samples while reducing the time for training data. The optimal values of the hidden bias and thus the input weight were searched through GA , just to enhance the prediction accuracy in ELM. The performance of this proposed forecast model was evaluated using coefficient of determination (R2), mean absolute error (MAE) , normalized root mean square error (nRMSE) and the result showed that day-ahead PV power prediction, with the SDA-GA-ELM model had higher accuracy & stability.

Marcello Anderson F.B. Lima et. al [17] proposed , solar forecasting ,using Multilayer Perceptron, Support Vector Regression ,Radial Base Function and with ML combination approach towards improving solar forecast power. In PT, correlation coefficient between the assets and one of the assets defined prediction errors, are counter balanced by another asset present. Results show that the Errors for datasets of Spain and Brazil, deep learning is used as a solar resource predictor, had significant gain with respect to other individual forecasting methods. For better management of solar energy PrevPT tools show, Positive impact on different weather conditions, solar resource availability and different places.

4. Research Gaps:

Many research gaps were identified in the literature review and there is a strong need to improve the predicting results, which can be seen by considering different machine learning methods and input variable. In three weather conditions such as: fog, snow and rain. Researchers predicted that short-term PV power, which is not taken into consideration, must be considered. In future, we can study the PV power prediction under extreme weather conditions, for improving the stability of the prediction model.
For PV power forecasting an integration of the HDNN model and dispatching algorithms, can help to simulate transactive energy scenarios.
The effect of different weather attribute may be studied such as wind effects, on the cooling of the solar modules, removing possible depositions of fine dust in rainfall etc.

5. Conclusion

Solar photovoltaic arrangement generate the solar electrical power. It depends on the weather condition in which it operate and it depends on some parameters like the amount of solar radiation and temperature.

These parameters depend on weather data. Solar Photovoltaic either On grid or Off grid both issues based on the Forecasted result.

- Based on primary literature review following research objectives may be set for investigation.
- We will use machine learning approaches on annual data for next year short and long term power generation forecasting.
- For management of reduction of the forecasting error we design the forecasting model which can predict the solar power generation.
- We can find new methods for improving the model accuracy, such as optimization of the hyper parameter, data cleaning.
- In future study, researcher can predict PV power for extreme climate conditions for improvement of efficiency, in prediction model.

References:

[1] GTM Research/SEIA. U.S. *Solar Market Insight, Report Q2. In Executive Summary*; National Renewable Energy Lab. (NREL): Golden, CO, USA, 2015.
[2] IEA. Renewables 2018—*Market Analysis and Forecast from 2018 to 2023*. Available online: https://www.iea.org/renewables2018.
[3] Sunghyeon Choi and Jin Hur, Ensemble Learner-Based Bagging Model Using Past Output Data for Photovoltaic Forecasting, *MDPI Journal of Energy*, 2020.
[4] Paulescu, M.; Paulescu, E.; Gravila, P.; Badescu, V., Weather Modeling and Forecasting of PV Systems Operation; Springer: London, UK, 2013.
[5] Peder Bacher, Henrik Madsen, Henrik Aalborg Nielsen, Online short-term solar power forecasting, *Solar Energy*, July 2009.
[6] Dongha Shin · Eungyu Ha · Taeoh Kim · Changbok Kim, Short-term photovoltaic power generation predicting by input/output structure of weather forecast using deep learning, Springer-Verlag GmbH Germany, part of Springer Nature 2020.
[7] Farid Touati, Amith Khandakar, Muhammad E.H. Chowdhury, Antonio Jr. S.P. Gonzales, Christian Kim Sorino and Kamel Benhmed , Photo-Voltaic (PV) Monitoring System, Performance Analysis and Power Prediction Models in Doha, Qatar, *Renewable Energy - Technologies and Applications*, IntechOpen , 2020.
[8] Chayut Tubniyom,Watcharin Jaideaw, Rongrit Chatthaworn, Amnart Suksri, Tanakorn Wongwuttanasatian, Effect of partial shading patterns and degrees of shading on Total Cross-Tied (TCT) photovoltaic array configuration, Elsevier, *Energy Procedia* **153** (2018) 35–41.
[9] Zhi-Feng Liu , Ling-Ling Li , Ming-Lang Tseng , Ming K. Lim, Prediction short-term photovoltaic power using improved chicken swarm optimizer - Extreme learning machine model, *Journal of Cleaner Production*, Elsevier(2019).
[10] Gangqiang Li ,Sen Xie , Bozhong Wang , Jiantao Xin, Yunfeng Li , And Shengnan Du , "Photovoltaic Power Forecasting With a Hybrid Deep Learning Approach", IEEE access(2020), 175871-80.
[11] Dan A. Rosa de Jesus, Paras Mandal, Shantanu Chakraborty and Tomonobu Senjyu, Solar PV Power Prediction Using A New Approach Based on Hybrid Deep Neural Network,IEEE(2019).

[12] Debasish Pattanaik, Sanhita Mishra, Ganesh Prasad Khuntia, Ritesh Dash, and Sarat Chandra Swain, “An innovative learning approach for solar power forecasting using genetic algorithm and artificial neural network”, *De Gruyter,Open Engg*(2020),630-641.

[13] Usman Munawar, Zhanle Wang, A Framework of Using Machine Learning Approaches for Short Term Solar Power Forecasting”, Springer 13 Janurary 2020.

[14] Rachit Srivastava, A.N. Tiwari, V.K. Giri, Solar radiation forecasting using MARS, CART, M5, and random forest model: A case study for India, *Heliyon* (2019).

[15] Javier López Gómez Ana Ogando Martínez , Francisco Troncoso Pastoriza , Lara Febrero Garrido , Enrique Granada Álvarez and José Antonio Orosa García ,Photovoltaic Power Prediction Using Artificial Neural Networks and Numerical Weather Data , *Sustainability ,MDPI*, (2020).

[16] Yi Zhou, Nanrun Zhou, Lihua Gong, Minlin Jiang, Prediction of photovoltaic power output based on similar day analysis, genetic algorithm and extreme learning machine, *Energy* (2020).

[17] Marcello Anderson F.B. Lima , Paulo C.M. Carvalho , Luis M. Fernandez-Ramírez , Arthur P.S. Braga, “ Improving solar forecasting using Deep Learning and Portfolio Theory integration” Elsevier, *Energy*(2020).

[18] Washington State University Extension Energy Program, Solar Electric System Design, Operation and Installation, *An Overview for Builders in the Pacific Northwest*, October 2009.

[19] J Sreedevi , Ashwin N , M Naini Raju ,A Study on Grid Connected PV system, IEEE (2016).