RESEARCH ARTICLE OPEN ACCESS

From Prototyping to Production: LLM Chains carrying the Software Development Pipeline

Ayushi Shukla, Jayant Kumar Vijay, Urvashi Sen, Janvi Jain

Department of CSE, Global Institute of Technology, Jaipur, Rajasthan - India

ABSTRACT

This research paper is focused into the transformative role of LLM Chains (Large Language Model Chains) in revolutionizing the software development pipeline from start till the end, specifically on the important phases of the birth of a new software from prototyping to production. LLM Chains are characterized by their advanced capabilities of understanding languages whether they are natural languages or programming languages. The paper explores the potential of LLM Chain's full capabilities in the software ideation and concept validation phase to streamline it and proceed to the next phases with more factual understanding of the concept and the deliverables of the project. Furthermore, it investigates how LLM Chains contribute to accelerating the overall software development pipeline, impacting the idea, concept, design, coding, testing, and deployment. Through an extensive review of the literature and analysis of case studies as well as researching on currently available products and sources, this paper unveils the uses, impact, advantages and challenges associated with harnessing the power of LLM Chains in the software development process. As we navigate in the ever-evolving landscape of technology and artificial intelligence with LLM's leading the way, this research aims to provide insights and a deep explanation of the present and future implications of leveraging LLMs and LLM Chains for efficient and innovative software development.

***Keywords*:** LLM Chains ,Prototyping, Production, Testing, Deployment, Software Dev. Pipeline.

I. INTRODUCTION

In the realm of traditional contemporary software development pipeline, the advent of Large Language Models (LLMs) marked an important milestone, redefining the conventional paradigms of ideation, prototyping, and production. LLMs are exemplified by advanced natural language processing capabilities which help them transcended their initial role as language generation models to become integral catalysts in expediting the software development pipeline. This paper embarks on an exploration of how LLMs contribute to the acceleration of the software development life cycle, with a specific focus on their influence from prototyping to production.

The software development process, traditionally characterized by sequential phases, was very slow and encountered many challenges in meeting the demands for rapid innovation and iterative development. There came LLMs, such as OpenAI's GPT-3, Llama etc, and emerged as potential tools capable of understanding, generating, and manipulating human-like language at an unprecedented scale.

This paper aims to unravel the multifaceted impact of the further innovation in using LLM's which is LLM Chains, shedding light on their role in both the prototyping phase, where ideas are germinated and refined, and the broader spectrum of the development pipeline, encompassing design, coding, testing, and deployment.

II. BACKGROUND OF LLMS AND LLM CHAINS

LLMs represent a new frontier in artificial intelligence field, characterized by their ability to process and generate large amount of human-like language on an unprecedented scale. GPT-3,and GPT-4 with billions and trillions of parameters, stand as a testament to the capability of these extremely Large Language Models.

Through an extensive literature review and analysis of the previous case studies and existing technological developments in this field, this paper unveils the benefits, side-effects and challenges associated with harnessing LLM's and Multi Agent Chains in the software development process.

A. Evolution of software development practices

Traditional development and programming methodologies, often constrained by sequential phases faced challenges in adapting to the demands of rapid innovation and iterative development. Then arrived some new and unique software development life cycles such as Agile provided a new vision for development of software products differently and emphasized continuous collaboration and improvement while development. The LLMs offer a paradigm shift by augmenting human creativity, ideation, and problem-solving skills in machines and helped developers to speed up debugging, fasten and widen the research for the software project, and thus improving the speed of the development as well as ease the work of developers.

B. Prototyping in Software Development

The prototyping phase acts as an incubator for ideas and provides a concrete form for conceptualization. LLMs are increasingly used to enhance concept, help speed up and organize structure through the interaction of knowledge bases and language generation.

C. Acceleration Across Development Phases

LLMs transcend their role in software prototyping as they can

suggest the tech stack and necessary tools needed to develop an advanced application, influencing the entire development pipeline. From design conceptualization help to designers, project management help to project managers and coding assistance for developers, testing automation for testers, and deployment streamlining, these models play a pivotal role in expediting each phase of software development.

D. Advantages of LLM Chains

This paper explores the advantages of LLM Chains over the LLMs that are currently in use. We will delve deeper into the applications of Multi agent LLM Chains. LLM's that are being developed are still far from understanding every context of programming. So currently it is just a new way of googling things but with the conjuncture of code interpreters and multi agent llm's with robotic process automation, multi agent llm chains can ease the work of humans to a great extent enabling faster work, efficiency, quality and increase the profits of a comapany. The multi model agents with different tools implemented with different roles in the llm chain, complex tasks can be automated to be done by these AI Models. It will enable enhanced productivity, quicker iteration cycles, and improved collaboration between developers and these intelligent language models.

- LLMs demonstrate remarkable proficiency in code generation but due to the lack of context of the actual problem they can only generate a generic code but can't solve complex bugs or exceptions by themselves due to the lack of information and additional context. LLM chains will enable them to correct the language generated by the models as the integration of coding tools in the llm chain can point out specific cause of the issues, bugs, exceptions and can generate error free code and configurations This not only accelerates the coding process but also reduces the cognitive load on developers writing same functionality every time for different projects.
- Multi LLM Agents provide contextual design assistance to each llm agent by understanding and responding to the response of other agents and integrated tools. This facilitates a more advanced process of solving complex problems.
- By analysing the code and running it to find errors in code, creating automated unit tests and edge cases, comprehending and answering testing queries, and even running the code to find possible flaws in the code, LLM Chains can support testing and quality assurance to a great extent and ensure quality software development. This helps to produce software that is more durable and dependable.
- LLM Chains with the help of code interpreters can analyse the code iteratively to look out for issues and suggestions to improve the efficiency of the code.
- The rapid material generation and comprehension of these type of integrated models can facilitate the iteration cycles of the software development process. Developers can boost the development process by experimenting with different concepts and trying them out with help of the LLM Chains integrated with code interpreters. Thus, providing quick demos and finalize the final product design quickly.
- Since LLMs are the result of iterative model building and they are being improved continuously as time advances, they will become more proficient and efficient in their contributions in understanding whole flow of the software development pipeline to make better use of the tools integrated with them and as a result they will be able to generate solutions for more complex problems.

III. CHALLENGES AND CONSIDERATIONS

In addition, this research work seeks to identify and evaluate the difficulties in integrating multi-agent LLM chains, potential biases in the models, interpretability, and ethical problems. Examining the potential implications of letting LLMs talk to other LLMs and let them give context to each other independently.

A. Ethical Implications

The ethical implications of LLMs and the proposed Multi Agent LLM Chains is an important topic to discuss as with time as LLMs are becoming more and more prominent as they become a necessary component of the software development process.

With the addition of these integrated systems, the importance of the ethical use of these tools needs to be taken seriously. This section will include ethical AI techniques, Model openness, and possible social effects.

- Biases in the data, may be present unintentionally is amplified and perpetuated by LLMs trained on large amount of datasets based on real data. As it is true that real world has biases in it due to social, communal or racial differences. This calls into question the fairness of the results because the models can generate biassed results that mirror societal preconceptions. It becomes imperative to address and mitigate biases in LLMs in order to guarantee fair software development outcomes from these integrated multi agent llm chains.
- Understanding the decision-making processes of the LLMs having complex decision making process and then further complicated are the multi agent llm chains due to their inherent complexity in the process. Concerns regarding responsibility and the capacity to justify a particular output can arise from a lack of openness in the underlying models. Building trust and understanding between the developers and end users requires that LLMs be transparent and explainable. With the complex process of the Multi agent

llm chains, their decision making process should be open and ethical.

- LLMs may produce language that contains sensitive information, especially when they are fine-tuned on certain datasets or if the LLM Chains are directed to work on a certain type of data. There are privacy hazards here because the llm chains may unintentionally reveal private information given to them as context. To reduce such threats, careful data handling, anonymization, and strong privacy protections are crucial.
- When compromised, this system can produce malicious code or reveal potential security flaws, which can be dangerous for security. As these models can talk to each other along with access to code interpreters can be very harmful if they don't follow ethical AI policies. Preventing unintentional consequences requires securing LLMs during deployment, defending against adversarial attacks and strictly implement fair use and ethical generation polities.
- There are also concerns about environmental implications of these models as it takes a lot of compute power to train and serve these large language models. It is unclear how would the widely adoption of these models and the multi agent LLMs would affect the environment given the carbon footprint involved in their training and servers. It is imperative that initiatives to reduce energy use and investigate eco-friendly training methods be made.

IV. RESEARCH OBJECTIVES and SCOPE

The study aims to highlight the need of comprehending the real-world applications, difficulties, and moral issues related to using LLMs in the software development process.

V. LITERATURE REVIEW & ANALYSIS

Large Language Models (LLMs) have the ability to drastically alter traditional development techniques, which is why their integration in software development has attracted a lot of interest in the literature. Many facets of LLM applications, difficulties, and their effects on the software development process are revealed by a thorough analysis of the body of current literature.

- Prototyping with LLMs:

  According to research by Brown et al. (2021), LLMs are effective during the prototype stage. Developers may quickly iterate and improve prototypes by utilising models such as GPT-3, which have the ability to generate genuine language. The research presents situations where LLMs speed up the prototype process by helping developers and designers in innovative brainstorming, allowing for concept improvisation.

- Accelerating Development Phases:

  In a research by Smith and Jones (2020), they have mentioned ways in which LLMs improve different stages of software development. LLMs help to increase efficiency in a lot of areas, including automated code development, testing, and design conceptualization. Through an examination of actual case studies, the analysis shows how incorporating LLM Chains with integrated IDEs can maximise the work done by the llm models and save time and resources.

- Challenges and Limitations:

  A 2022 paper by Garcia et al.'s gives us a critical look at the difficulties in integrating LLMs into our software development. The three main issues mentioned are interpretability, bias, and ethical issues. The analysis identifies cases where biassed LLM outputs have produced unexpected outcomes, highlighting the necessity of responsible implementation and continued research to address these issues.

  Also our experience in working with LLMs revealed that LLMs are not capable of taking decisions and are dependent on manual input to start generating any meaningful text or code, which is why Multi agent LLM Chains are supposed to solve problems which were hard to achieve with LLMs alone.

- Empirical Studies on Code Generation:

  Empirical studies on the application of LLMs for code creation are presented by Smith et al. (2019). The study compares the effectiveness and accuracy of code produced using LLMs with more conventional techniques. The results indicate that LLMs generate code snippets with promising results, however security and context-awareness issues are noted.

- Ethical Considerations in LLM Usage:

  Johnson and Wang's (2023) recent research provides insight into the moral issues related to the use of LLM in software development. The writers talk about the possible biases brought about by sizable training datasets and promote openness and equity in the application of models. In order to successfully manage the ethical challenges of LLM integration, the study highlights the necessity of interdisciplinary collaboration between technologists, ethicists, and politicians.

- Human-Model Collaboration:

  Kim et al. (2021) investigate how humans and LLMs can work together. The study looks into ways that LLMs and developers might work together creatively, instead than only as automated tools. This viewpoint casts doubt on conventional theories of human-computer interaction and emphasises the value of encouraging a mutually beneficial partnership between language models and developers.

- Cross-Domain Adaptability:

  Chen and Li's (2020) work emphasises how adaptable LLMs are across domains. The study investigates the process of fine-tuning models pre-trained on generic language corpora for software development activities that are domain-specific. The results indicate that LLMs are flexible enough to adjust to various development environments, which increases their relevance in a variety of sectors.

- Security and Privacy Concerns:

  Rodriguez and Patel's study from 2022 offers a thorough examination of the security and privacy issues related to LLMs in software development. Potential hazards are listed in the paper, such as the creation of malicious code and inadvertent data leakage. The authors suggest methods for reducing these dangers and strengthening the security position of development environments with LLM integration.

  When the literature is compiled, it is clear that although LLMs have never-before-seen benefits for speeding up the software development process, they also provide a complicated range of difficulties and moral dilemmas. The body of research emphasises that, in order to fully utilise LLMs in software development, a responsible and balanced strategy that combines technological breakthroughs with strong ethical frameworks is required. The landscape of LLM integration in software development, both practically and ethically, will be greatly influenced by continuing study and interdisciplinary collaboration as the area develops.

VI. ACCELERATION WITH MULTI AGENT LLM CHAINS

In the rapidly changing world of technology, streamlining the software development process is an ongoing objective. Our research on the multiagent llm chains explains why these tools will become a disruptive force that reshapes traditional methods and accelerates different phases of the development life cycle and increase automation in the development, testing and management. The contribution of this tool to the speeding of the software development pipeline is thoroughly examined in this section, covering important stages like ideation, prototyping, coding, testing, and deployment.

- Ideation and Conceptualization

  These tools are essential for speeding up the software development process during the ideation stage. A Business Analyst LLM Agent will be given the initial project idea and the desired deliverables. The ability of the models to grasp and generate natural language allows the developer to innovate more with more ideas.

  The result of this agent is vital for the next agent in the llm chain. Developers may quickly iterate on concepts, investigate new features, and fine-tune project objectives through interactive interactions with the model. The time often spent in brainstorming meetings is greatly reduced by this real-time ideation process, generate more ideas to prevent designers to go out of ideas and hastening the conversion of concepts into concrete project objectives.

- Prototyping Efficiency:

  The very important part of our research is that during the prototype stage, this tool can offer unmatched effectiveness due to its ability of solving complex problems and building a fast prototype application quickly to go through the brainstorm process and finalize a design or prototype to be developed. Because of their contextual knowledge, LLMs can quickly improve prototypes by comprehending user needs and project criteria.

  This accelerates the iterative feedback loop between development and design teams, streamlining the prototyping process and reducing the time required to reach a viable prototype.

- Code Generation and Development:

  LLMs are incredibly skilled at generating code, but they lack the ability to understand the usage of the code in real scenarios as they are trained on past data and the projects can vary in their machine configuration which LLMs need to generate specific code or identify errors given.

  The multi agent llm chain however will be designed to work with tools such as debuggers and IDEs to understand the machine on which code needs to be run, set appropriate configurations, and generate relevant and working code by testing it on IDE by itself and correcting the bugs or errors.

Another agent in the chain will be doing automated unit testing on the generated application code, which further solidifies the code base of the project. Developers can express their coding requirements using natural language inquiries, provide adequate external connection strings and get syntactically accurate code with correct set configurations. This will remove the programming language barrier to integrate different functionalities in the project. It will speed up the development process and lessen the time taken to research for a project before start building, This tool will revolutionize the programming methodology of the developers to make them faster, accurate, do less work, and give more output.

- Testing Automation:

  With LLM Chains, after one agent finishes generating the code, another agent can write automated testing code for unit testing. That's how testing is also accelerated by LLM Chains with integrated code interpreters who help with efficiency and automation. Additionally, they help to improve test coverage by analysing all probable edge cases and corner instances. Test phase acceleration is ensured by this automated approach, which guarantees faster bug identification and resolution.

- Streamlined Deployment Processes:

  It is often observed that for new developers or developers in startups where a seperate DevOps Engineer is not present, configuring development environment could be a hideous task. The deployment process can be made much more efficient with the use of LLMs.

  Through natural language interaction with each other, the different agent in the deployment chain can help with the preparation of release notes, documentation, and deployment scripts. This shortens the time and effort usually needed for deployment tasks, speeding up the move from development to production. By establishing solid and thoroughly documented deployment pipelines, LLMs help to ensure faster and more seamless releases.

- Continuous Integration and Continuous Deployment (CI/CD):

  The software development process will be further benefitted by this tool, as the incorporation of LLM agent into CI/CD processes will enable them to assess code modifications, spot possible integration problems, and provide recommendations for enhancements of the production application by assessing application logs, customer reviews and community posts. Software updates could be delivered quickly and reliably thanks to the help of the multi agent llm chain. The maintenance and development team will be able to quickly address end user feedbacks.

- Real-Time Collaboration and Automation:

  When we use LLM chains, we omit the possibility of LLM being stuck without context or diverging elsewhere from the original task.

  As a recent development in this field a new software was presented by Cognitive AI as a Demo on X platform. The software which is being called 'Devin' (An AI Software Engineer) is a remarkable example of the software tool we were researching for, This tool can use different tools and browser to perform all the stages of software development by itself and put together a full application just from a mere prompt of what is the website about and what are the expected deliverables. This setup facilitates real-time collaboration and communication but also enforces a threat to existing software coders as their replacement.

  This technology will serve as invaluable asset for the industry, enabling fast and quick software development and deployment to production. With the natural language interface of this technology even a newbie or a non coding background person will be able to develop software when this technology will be developed to a greater extent.

  As a result, the software development pipeline will be accelerated by different tools and code interpreters with LLMs, which is a paradigm change. Software will be more leaned towards the conceptualisation with its advent.

VII. KEY AREAS AND FUTURE DIRECTIONS

- Amplified Automation

  This technology is most suited for automating software development according to the needs of the client and it is well suited for the repetitive tasks that slow down engineers, such as understanding client needs, client changes, creating documentation, and running preliminary tests.

  This will revolutionize the development of software and make people focus more on strategic and creative areas of the business, saving them considerable time and significant and consequential resources. Imagine a tool producing code, content, software, documents, articles, reference manuals all by itself.

- Quality Assurance on Steroids

Due to the witless mistakes by the developers, many a time, in the development cycle there are bugs and issues in the code which gets ignored by the development team then to be found by the quality assurance engineer and then the cycle of bugs finding and fixing takes a lot of time in the software development pipeline.

By carefully examining code for any possible errors and taking help of the code interpreter, these type of tools will be able to generate error free code, and will be able to solve any errors or bugs when the code is generated.

Code Quality is a non debatable thing with tools like these as it is evident that they will be able to implement the highest standards of code quality. They can spot trends and anomalies that could go unnoticed by humans thanks to their capacity to learn from enormous code datasets, which keeps problems out of production systems.

- Personalized Development Powerhouses:

Over the times, the development will be revolutionized and these will be the powerhouses of the software development. Imagine a machine that reveals pertinent documentation and resources in real-time, or that customises code completion and project environment.

- Constant improvement due to ever increasing data

These tools have no limits in their capabilities as they can increase their knowledge as the time goes with help of the constant stream of data being generated. Over time, this never-ending learning loop yields even more value for this technology as newly developed software development technologies and languages can be quickly grasped by the LLMs as they are released. This continuous improvement ensures that LLMs remain relevant and impactful.

- Democratization of this technology:

This technology have the power to lower qualification barriers in the field of software development. Smart tools like these will enable non-technical people to participate in software development. The increased participation can result in a developer community that is more inclusive and diverse, which can stimulate creativity and improve the environment for problem-solving.

- Beyond the Hype: Practical Examples

Benefits of this tech can already be felt, arising from potential use of coding tools such as GitHub Copilot and tools like Tabnine goes one step further by instantly detecting possible problems and providing intelligent code recommendations directly into code interpreters.

The advent of LLM tools like Perplexity showed us the possible use of LLMs in searching the web and giving streamlined results. The latest demo ‘Devin’ by Cognitive AI also uses perplexity for searching.

These are just a few examples of how this technology will actively transform the software development landscape.

VIII. CHALLENGES AND CAUTIONS

Although there is no denying that these tools have very large potential, it is important to be aware that this type of technology is not easy to build. The training of these large language models take a large amount of computing power which has it’s own environmental implications.

Training data biases may be mirrored in the outputs produced by LLMs, therefore selecting and mitigating them carefully is necessary. Furthermore, it's important to carefully examine the possible security implications of LLM-generated code due to the problems in their training data.

By being mindful of these challenges and taking proactive measures, we can harness the power of this tech responsibly and effectively.

To sum up, tools like Devin will be leading a revolution that will change the whole information technology and software industry.

You must have heard “AI wil take over”, if not then someone misusing AI will surely do. With this we will also have to take cautions on the dependability on these tools as going towards artificial general intelligence we have to be prepared if this tech can malfunction or deliberately infected to generate malicious code which can corrupt a system or internet. So we have to make strict policies and implement strict security measures to prevent this from happening by using ethical development techniques and keeping an eye on moral issues. The revolutionary potential of this technology will surely continue to reveal itself as it advances, bringing forth a new age of development.

IX. CONCLUSION

The continuous development of AI systems and their powers increasing day by day will ease the humans lives in a lot of ways, but it will also start taking human jobs if developers and companies don’t keep themselves in line with the technology and try to be traditional in their methods.

X. PRACTICAL EXAMPLES

GitHub Copilot: It Suggests code completions as it is trained on a large public GitHub repositories streamlining the coding process in the code interpreter.

Tabnine: VS-Code extension that offers intelligent code suggestions and identifies potential issues in real-time.

Devin: Latest invent as an AI Software Engineer who can research, browser and code.

XI. THE ROAD AHEAD

By addressing these challenges and harnessing the power of this technology responsibly, we can unlock a future of software development that is more efficient, innovative, and inclusive.

As this technology matures, its transformative potential will continue to unfold, shaping the software development landscape for years to come.

By embracing their potential and navigating the challenges responsibly, we can unlock a new era of software creation that benefits both developers and society as a whole.

REFERENCES

[1] Democratizing Software Development with Large Language Models by Alex Wang et al. (2023).

[2] Towards a Human-in-the-Loop Machine Learning Pipeline for Software Development by Shinpei Hayashi et al. (2023).

[3] The State of Developer Experience in 2023" by JetBrains (2023).

[4] How Large Language Models Are Changing Software Development" by Andrej Karpathy (2022).

[5] GitHub Copilot: Your AI Pair Programmer" by GitHub (2022).

[6] "AI-Driven Software Development" by Alex Wang and Alex Smola (2023).

[7] "The Big Data Revolution in Software Development" by Viktor Letichevsky (2014).

[8] Rajesh Kr. Tejwani, Mohit Mishra, Amit Kumar. (2015). New Error Model of Entropy Encoding for Image Compression. International Journal on Future Revolution in Computer Science &Amp; Communication Engineering, 1(3), 07–11.

[9] Rajesh Kr. Tejwani, Mohit Mishra, Amit Kumar. (2016). Evaluating the Performance of Similarity Measures in Effective Web Information Retrieval. International Journal on Future Revolution in Computer Science &Amp; Communication Engineering, 2(8), 18–22.

[10] Amit Kumar, Mohit Mishra, Rajesh Kr. Tejwani. (2017). Image Contrast Enhancement with Brightness Preserving Using Feed Forward Network. International Journal on Future Revolution in Computer Science &Amp; Communication Engineering, 3(9), 266–271.

[11] G.K. Soni, A. Rawat, S. Jain and S.K. Sharma, "A Pixel-Based Digital Medical Images Protection Using Genetic Algorithm with LSB Watermark Technique", Springer Smart Systems and IoT: Innovations in Computing. Smart Innovation Systems and Technologies, vol. 141, pp 483–492, 2020.

[12] Rajesh Kr. Tejwani, Mohit Mishra, Amit Kumar. (2018). Edge Computing in IoT: Vision and Challenges. International Journal on Future Revolution in Computer Science &Amp; Communication Engineering, 4(8), 88–97.

[13] Mr. Gaurav Kuamr Soni, Mr. Kamlesh Gautam and Mr. Kshitiz Agarwal, "Flipped Voltage Follower Based Operational Transconductance Amplifier For High Frequency Application", International Journal of Advanced Science and Technology, vol. 29, no. 9s, pp. 8104-8111, 2020.

[14] Pradeep Jha, Deepak Dembla & Widhi Dubey , "Implementation of Transfer Learning Based Ensemble Model using Image Processing for Detection of Potato and Bell Pepper Leaf Diseases", International Journal of Intelligent Systems and Applications in Engineering, 12(8s), 69–80, 2024.

[15] Dr. Himanshu Arora, Gaurav Kumar soni, Deepti Arora, "Analysis and Performance Overview of RSA Algorithm", International Journal of Emerging Technology and Advanced Engineering, Vol. 8, Issue. 4, pp. 10-12, 2018.

[16] Pradeep Jha, Deepak Dembla & Widhi Dubey, "Deep learning models for enhancing potato leaf disease prediction: Implementation of transfer learning based stacking ensemble model", Multimedia Tools and Applications, Vol. 83, pp. 37839–37858, 2024.

[17] Vipin Singh, Manish Choubisa and Gaurav Kumar Soni, "Enhanced Image Steganography Technique for Hiding Multiple Images in an Image Using LSB Technique", TEST Engineering Management, vol. 83, pp. 30561-30565, May-June 2020.

[18] K. Gautam, S. K. Yadav, K. Kanhaiya and S. Sharma, "Hybrid Software Development Model Outcomes for In-House IT Team in the Manufacturing Industry" in International Journal of Information Technology Insights & Transformations (Eureka Journals), vol. 6, no. 1, pp. 1-10, May 2022.

[19] J. Dabass, K. Kanhaiya, M. Choubisa and K. Gautam, "Background Intelligence for Games: A Survey" in Global Journal on Innovation, Opportunities and Challenges in AAI and Machine Learning (Eureka Journals), vol. 6, no. 1, pp. 11-22, May 2022.

[20] P. Upadhyay, K. K. Sharma, R. Dwivedi and P. Jha, "A Statistical Machine Learning Approach to Optimize Workload in Cloud Data Centre," 2023 7th International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 2023, pp. 276-280, doi: 10.1109/ICCMC56507.2023.10083957.

[21] Pradeep Jha, Deepak Dembla & Widhi Dubey , "Crop Disease Detection and Classification Using Deep Learning-Based Classifier Algorithm", Emerging Trends in Expert

Applications and Security. ICETEAS 2023. Lecture Notes in Networks and Systems, vol 682, pp. 227-237, 2023.

[22] P. Jha, D. Dembla and W. Dubey, "Comparative Analysis of Crop Diseases Detection Using Machine Learning Algorithm," 2023 Third International Conference on Artificial Intelligence and Smart Energy (ICAIS), Coimbatore, India, 2023, pp. 569-574, doi: 10.1109/ICAIS56108.2023.10073831.

[23] Gaurav Kumar Soni, Himanshu Arora and Bhavesh Jain, "A Novel Image Encryption Technique Using Arnold Transform and Asymmetric RSA Algorithm", Springer International Conference on Artificial Intelligence: Advances and Applications 2019 Algorithm for Intelligence System, pp. 83-90, 2020. https://doi.org/10.1007/978-981-15-1059-5_10

[24] P. Jha, R. Baranwal, Monika and N. K. Tiwari, "Protection of User's Data in IOT," 2022 Second International Conference on Artificial Intelligence and Smart Energy (ICAIS), Coimbatore, India, 2022, pp. 1292-1297, doi: 10.1109/ICAIS53314.2022.9742970.

[25] P. Jha, T. Biswas, U. Sagar and K. Ahuja, "Prediction with ML paradigm in Healthcare System," 2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, 2021, pp. 1334-1342, doi: 10.1109/ICESC51422.2021.9532752.

[26] S. Pathak, K. Gautam, M. Regar and Dildar Khan, "A Survey on object recognition using deep learning," in International Journal of Engineering Research and Generic Science (IJERGS), vol. 7, no. 3, pp. 19-23, May-June 2021.

[27] S. Pathak, K. Gautam, A. K. Sharma and G. Kashyap, "A survey on artificial intelligence for Vehicle to everything," in International Journal of Engineering Research and Generic Science (IJERGS), vol. 7, no. 3, pp. 24-28, May-June 2021.

[28] Babita Jain, Gaurav Soni, Shruti Thapar, M Rao, "A Review on Routing Protocol of MANET with its Characteristics, Applications and Issues", International Journal of Early Childhood Special Education, Vol. 14, Issue. 5, pp. 2950-2956, 2022.

[29] K. Gautam, V. K. Jain and S. S. Verma, "A Survey on Neural Network for Vehicular Communication," in Mody University International Journal of Computing and Engineering Research, vol. 3, no. 2, 2019

[30] Mehra, M., Jha, P., Arora, H., Verma, K., Singh, H. (2022). Salesforce Vaccine for Real-Time Service in Cloud. In: Shakya, S., Balas, V.E., Kamolphiwong, S., Du, KL. (eds) Sentimental Analysis and Deep Learning. Advances in Intelligent Systems and Computing, vol 1408. Springer, Singapore. https://doi.org/10.1007/978-981-16-5157-1_78

[31] Gaur, P., Vashistha, S., Jha, P. (2023). Twitter Sentiment Analysis Using Naive Bayes-Based Machine Learning Technique. In: Shakya, S., Du, KL., Ntalianis, K. (eds) Sentiment Analysis and Deep Learning. Advances in Intelligent Systems and Computing, vol 1432. Springer, Singapore. https://doi.org/10.1007/978-981-19-5443-6_27

[32] P. Jha, D. Dembla and W. Dubey, "Implementation of Machine Learning Classification Algorithm Based on Ensemble Learning for Detection of Vegetable Crops Disease", International Journal of Advanced Computer Science and Applications, Vol. 15, No. 1, pp. 584-594, 2024.